English

Fujitsu Software

BS2000
Performance Handbook

User Guide

Valid for:
BS2000 V21.0B
X2000 V6.5

February 2025

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to: bs2000.info@fujitsu.com.

Certified documentation according to DIN EN ISO 9001:2015

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2015.

Copyright and Trademarks

Table of Contents

Preface

The aim of every IT department is to satisfy the requirements of the system users while at the same time keeping costs to a minimum.

The following aspects make the adjustment of the system environment (hardware and software) a dynamic process:

- The steady increase in real-time processing with all its options of directly handling customer service and office work.
- The constant increase in applications and functions (data networks, distributed processing).
- The interrelationships between these functions (because of their mutual dependence).
- The growth of existing applications as a result of increased business volume.
- The load fluctuations in daily business.

The following steps should be taken to ensure that the system is used economically:

1. Determine precise performance expectations.
2. Define the configuration.
3. After taking up production, **carry out monitoring runs** to determine whether the performance expectations are being met.
4. If not, concentrate on those bottlenecks whose removal promises the greatest improvement in performance.
5. Once the bottlenecks have been discovered and removed, **repeat the monitoring runs**, as this often reveals other bottlenecks which were previously concealed.
6. Even after performance expectations have been met, it is essential to carry out **periodical monitoring runs**, for only in this way is it possible to detect tell-tale signs of saturation in the main resources due to the increasing workload and to avoid critical situations in the system.

Objectives and target groups of this manual

The “Performance Handbook” is intended to assist the system user in assessing and improving the performance level of his/her data processing system. It is aimed in particular at personnel in data centers and the system support.
Information for the fine-tuning of both configuration and software makes it easier to use SE servers more cost-effectively.

This Performance Handbook is a component part of the basic documentation for BS2000 OS DX. It provides a straightforward summary of the performance aspects of IT operation for SE servers.

Summary of contents

This manual describes the potential for the fine adjustment of the system configuration and software for ensuring that the required performance is provided and for optimizing the use of resources. Where necessary a distinction is made between the different SE servers.

The manual consists of two parts:

- Part 1 contains a general overview of the monitored variables, objects and processes which are relevant to performance, and measurement tools. It comprises the following chapters:
  - Characteristic performance values
  - Objects relevant to performance
  - System-internal processes which are relevant to performance
  - Measurement tool openSM2
- Part 2 contains recommendations and measurements which relate to specific situations and particular hardware or software configurations. It comprises the following chapters:
  - Servers
  - Peripherals
  - Data backup
  - VM2000
  - System and application control
  - System and application analysis with openSM2

At the end of the manual you will find an appendix and a reference section that will make it easier for you to work with the manual.

Readme file

The product-specific Readme file contains details of functional changes to the current product version and additions to this manual.

Readme files are available to you online in the information on the product concerned in addition to the product manuals at https://bs2manuals.ts.fujitsu.com. You will also find the Readme files on the Softbook DVD.

Information under BS2000

When a Readme file exists for a product version, you will find the following file on the BS2000 system:

`SYSRME.<product>.<version>.<lang>`

This file contains brief information on the Readme file in English or German (<lang>=E/D). You can view this information on screen using the `/SHOW-FILE` command or an editor.
The `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` command shows the user ID under which the product's files are stored.

Additional product information

Current information, version and hardware dependencies and instructions for installing and using a product version are contained in the associated Release Notice. These Release Notices are available online at https://bs2manuals.ts.fujitsu.com.

Changes since the last edition of the manual

The following major changes have been made since the last edition of this manual:

- The manual has been updated to cover BS2000 OS DX, release BS2000 V21.0B.
- Updated measured networking performance values for BS2000 server SE710 and SE730 with HNC Software V6.5.

Notational conventions

On account of the frequency with which the names are used in this guide, the following abbreviations have been adopted for the sake of simplicity and clarity:

- **BS2000 servers** for the servers with /390 architecture and the x86 servers. These servers are operated with the corresponding BS2000 operating system.
- Servers with /390 architecture (**/390 servers** for short) for the Server Unit /390 of the Fujitsu server BS2000 SE series.
- Servers with x86 architecture (**x86 servers** for short) for the Server Unit x86 of the Fujitsu server BS2000 SE series (x86-64 architecture).
- **SE servers** for the Fujitsu server BS2000 SE series (Server Units /390 and x86).
- **SM2** for the openSM2 measurement system, provided **no** distinction need be made between the openSM2 measurement system and the SM2 monitor.

The following conventions are used in this Migration Guide:

¡	to indicate particularly important information

[ ] References to related documents are provided in abbreviated form in the text. The full title of each publication referenced by a number is provided in full after the corresponding number in the “Related publications” section.

Commands to which reference is made in this manual are described in the “Commands” manual [15 (Related publications)] and the macros mentioned are described in the “DMS Macros” manual [8 (Related publications)]. The metasyntax of the commands and macros is given in the corresponding manuals.

Note on the units of measurement used

This manual uses both German and English notation. The following definitions apply:

1 KB = 2^{10} bytes = 1,024 bytes
1 Kbyte = 10^3 bytes = 1,000 bytes

1 MB = 2^{20} bytes = 1,048,576 bytes
1 Mbyte = 10^6 bytes = 1,000,000 bytes

1 GB = 2^{30} bytes = 1,073,741,824 bytes
1 Gbyte = 10^9 bytes = 1,000,000,000 bytes

Characteristic performance values

Every IT system can be divided hierarchically into the following levels:

- Application programs

  The user's requirements (as represented by the application programs) are transformed into tasks at the system software level, and into instructions and I/O operations at the hardware level.

- System software

  The system software no longer views the application programs as its workload, but rather the number of concurrent tasks and the differing demands they make.

- Hardware

  At hardware level the workload appears as a set of commands, input/output operations and storage reservations. At user level many different workloads can result in the same hardware workload.

The performance of an IT system is measured and evaluated with the help of characteristic performance values. This chapter examines the following aspects:

- Characteristic performance values of an online application
- Characteristic performance values of a batch application
- Formulating a performance expectation
- Relative performance factor of the CPU (RPF)
- Characteristic main memory values
- Characteristic peripheral values

Characteristic performance values of an online application

In both transaction and interactive mode, the unit of IT work is the **transaction**. User requests in the form of inputs are represented in the system by means of transactions.

In **TP mode** (also known as transaction mode or inquiry and transaction mode) the terminal user can only communicate with programs that have been predefined by the application. Normally, a large number of data display terminal users work with a relatively small number of **application programs**.

In **interactive mode** (also known as dialog mode or timesharing mode), each terminal user formulates his/her own application, which is then used to solve the particular problem in dialog. Programs used to control the individual dialog applications are normally **system programs** for creating, testing and modifying files or programs.

The time span between the user's input and the end message from the system is referred to as the **transaction time**. It may be made up of a number of individual responses, each of which has its own response time.

Characteristic values in an online application

- Transaction rate:
  Sum total of successfully terminated transactions per unit of time.
- Response time:
  Time taken by **one** processing operation by the server.
- Number of terminals
  (equivalent to the number of active terminal users).
- Efficiency per terminal.

Diagram of time definitions

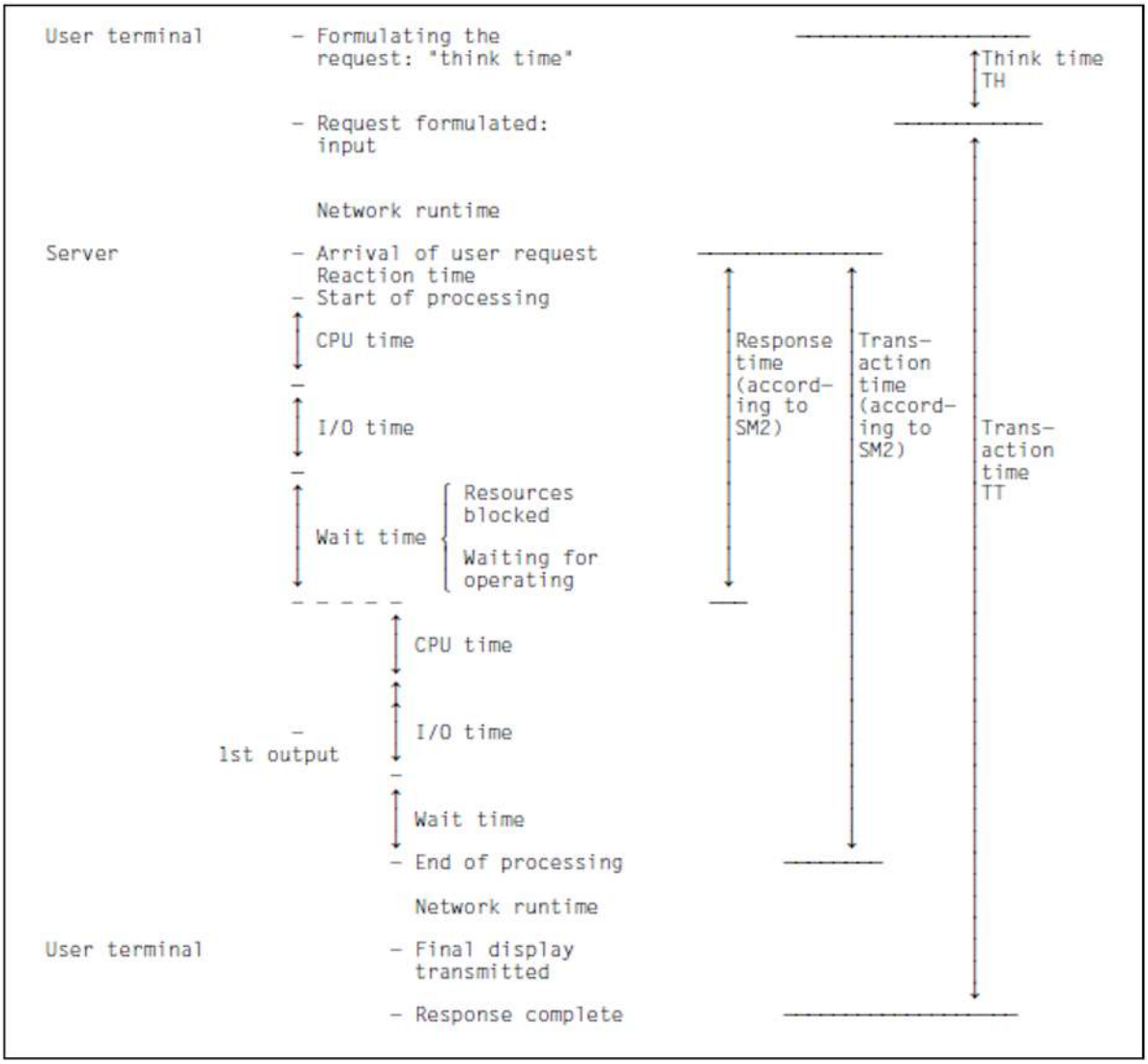


Figure 1: Time definitions in an online application

> ℹ If there is only **one** response per transaction (which is usually the case in TP mode), the response time is identical with the transaction time.

The time definitions shown (response time, transaction time, think time) and the transaction rate can be recorded by the SM2 monitoring program BCAM-CONNECTION, but only for applications that work with the BCAM transport system via the IDCAM or ITIAM program interface.

For applications that work with the BCAM transport system via the SOCKETS program interface, the SM2 monitoring program BCAM-CONNECTION returns connection-specific values via PORT numbers:

- INPROC:
  Time from the receipt of a message by BCAM until it is fetched by the application. The INPROC time comprises the INWAIT time (i.e. the time between BCAM indicating that the message has arrived and the application fetching the message).

- REACT:
  Time between the send call and the immediately preceding receive call of an application. In TP mode this time can be regarded as the response time.
- OUTPROC:
  Time from the send call until the message’s last byte is transferred to the network.

The TCP transport service works on a stream-oriented rather than a message-oriented basis. It is therefore not possible to obtain the transfer time through BCAM for transactions with several responses per input. Nor is the think time recorded.

If the input message length is short (< 500 bytes), the number of TSDUs received (Transport Service Data Units; jobs to BCAM) per second can be interpreted as a measurement of the transaction rate.

The measured values INPROC, INWAIT, REACT and OUTPROC are also obtained for applications that work with the BCAM transport system via the ICMX, IDCAM and ITIAM program interfaces.

Characteristic performance values of a batch application

With batch processing, the unit of IT work is the **job**.

Diagram of time definitions

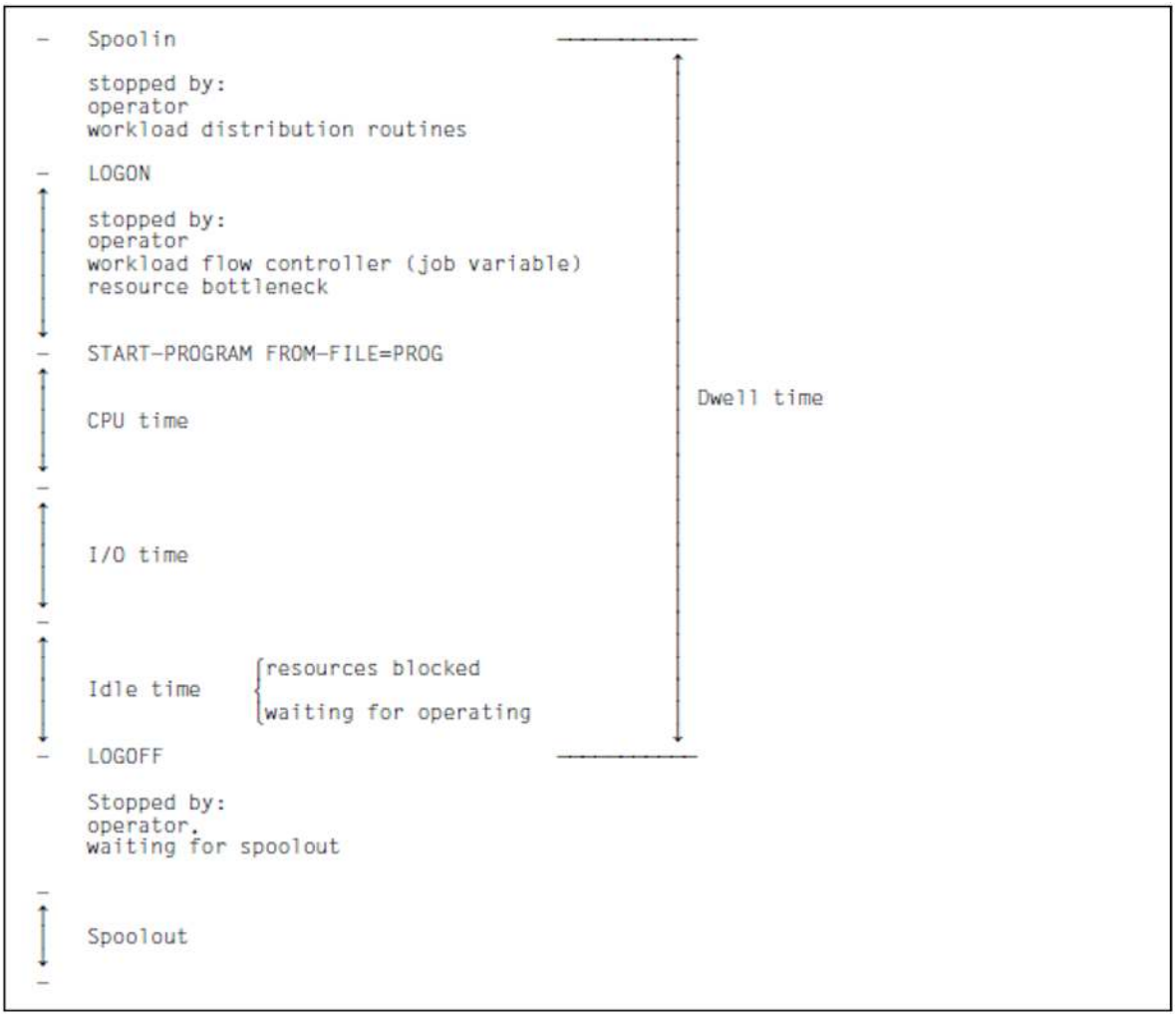


Figure 2: Time definitions in batch processing

Characteristic values in batch processing

- Throughput rate:
  Sum total of successfully executed jobs per unit of time
- Dwell time:
  Time required to process one job

Formulating a performance expectation

The following fundamental relation holds between the transaction time, the transaction rate and the number of terminals:

(Think time + Transaction time) * Transaction time = Number of terminals

The term “transaction time” summarizes the system's reaction to a wide variety of requirements. For this reason it is difficult to decide whether a transaction time is appropriate or not. Due to the costs involved in operating the terminal and the user's working hours, the following method is recommended:

The terminal cannot be used during the transaction time as it is waiting for the server. During the think time the terminal and user interwork by reading the output or typing in a new entry. The proportion of think times in the total busy time of the terminal is the efficiency value E for terminals. If the average value for all terminals drops below 75%, this is an indication of performance problems. Of course, pauses for a “breather” must not be taken into account when calculating think times.

The following values are used:

TH: Think Time
TT: Transaction Time
TR: Transaction Rate

According to the definition, the following is true for the **efficiency value (E) for terminals**:

E = TH / (TH + TT)

Let the number of transactions to be performed per second for an application be TR. Assuming the number of terminals required for this application is K, then

K = (TH + TT) * TR

K = TH / E * TR

If an efficiency value of 0.75 is assumed for terminals, the formula is:

K = 1,33 * TH * TR

The table below contains the values for 1/E for various efficiency values.

E [%]	**70**	**75**	**80**	**85**	**90**	**95**
1/E	1.43	1.33	1.25	1.18	1.11	1.05

The formula E = TH / (TH + TT) shows that relatively long transaction times (or response times in the case of only **one** response per transaction) can be accepted in the event of long think times, while in the event of short think times the transaction times have a lasting effect on the efficiency value and hence on the number of terminals required.

The number of terminals required is determined by the utilization (U) of the terminal as well as by the efficiency value. The utilization is the proportion of the total time spent by the machine either waiting for a response (transaction time) or preparing a new input (think time). Time taken for a “breather”, for instance, reduces the utilization without the device really being free. In the case of a data entry application, documents are generally processed from a stack. Workstations can therefore be utilized to a very high level.
It is possible to achieve terminal utilization of 90% under these conditions. A slight loss of 10% is allowed for to accommodate “breathers”. Utilization must be set lower for other applications. In the case of TP mode with customer traffic, it is desirable to eliminate inconvenient waiting times for the customers. These occur when there is no terminal free and hence the customer cannot be served immediately. The situation is similar in the case of program development. In cases such as this, terminal utilization (U) should not exceed 60%.

The **number of terminals (K)** required can then be calculated by:

$K = (TH * TR) / (E * U)$

Example

The following two applications exist:

The first application is a data entry application with a think time of 20 seconds and a transaction time of 1 second. On average 90% of all data entry stations are always in use. 7,200 documents must be processed per hour.

The second application is a seat booking application and involves dealing with the public. Think time is 120 seconds and transaction time is 3 seconds. In order to avoid noticeable delays for customers who wish to book, terminals are only utilized to 60% of their capacity. 360 bookings per hour must be processed.

$E_1 = 95\%$; $U_1 = 90\%$; $TR_1 = 2/s$

($E_1 = TH / (TH + TT) = 20s / (20s + 1s) = 0.95$)

$E_2 = 97\%$; $U_2 = 60\%$; $TR_2 = 0.1/s$

$K_1 = (20 * 2) / (0.95 * 0.90) = 47$ terminals

$K_2 = (120 * 0.1) / (0.97 * 0.60) = 21$ terminals

A total of 68 terminals are required for both applications. You must ensure that the values for the transaction time contained in the efficiency value for terminals are feasible for the server.
This constitutes the performance expectation which needs to be calculated.

It can only be met if the resources required by the application do not exceed the limits of the server and the server has a certain amount of reserve capacity. Here the load on those resources which are heavily used by the application concerned plays a particularly important part. If the application involves a large number of I/O operations to disk, you must ensure that utilization of the disk drives is kept at a low level. The situation is not improved by additional spare capacity in the server.
The opposite is true of applications which require a large number of calculations. Paging is unhelpful in both cases.

There is a natural lower limit for all transaction times. This is reached when a terminal performs its transaction using a server which is otherwise completely free of other operations and using data transmission paths which are equally free of other operations. It is, of course, not cost-effective to use an IT system like this. If other terminals and, where appropriate, other applications are added, the time required for execution is extended due to mutual obstruction by the transactions.

A **dilation factor (D)** is defined:

D = TT (current workload) / TT (empty server)

This dilation factor is the price is to be paid for the economic utilization of the system. To what extent a particular dilation factor is acceptable depends in turn on the think time and the optimal transaction time TT (unoccupied server). Dilation factors of 10 are still acceptable in the case of short transaction times and long think times. If, in the case of a think time of only 10 seconds, the optimal transaction time is even as much as 3 seconds, then a dilation factor of three is already too large. The **optimal transaction time** can either be measured on a server otherwise free of other operations or be estimated using the following formula:

TT (empty server) = CPU + n * IO + NL

where:

CPU: CPU time required for the transaction
IO: mean time for an I/O operation
n: number of I/Os (including paging) required for the transaction
NL: network runtime for the input **and** output messages

All these values can be altered by means of modification to the hardware or the software. In the case of the CPU and I/O times, several tasks generally have to be taken into account, in transaction mode, for example, UTM and DBH tasks.

> The dilation factor measured by SM2 does not take the network runtime into account. Only the internal server data is processed.

Example 1

Database query with the following values:

CPU=0.1s
IO=0.004s
NL=0.1s
n=60

TT (empty server) = 0.1s + 60 * 0.004s + 0.1s = 0.44s

Using a dilation factor of 3 for the server and the network results in a $TT_{current}$ of 3 * 0.44 = 1.32 (= ca. 1.3).
Using a think time of 40 seconds, the efficiency value for terminals can be calculated at 97%:

E = TH / (TH + $TT_{current}$) = 40s / (40s + 1.3s) = 0.97

If the think time is only 20 seconds, the efficiency value for terminals drops to 94%.
In this case you should check whether the I/O times could be reduced, for instance by reducing the number of inputs/outputs or quicker peripherals.

Example 2

Scrolling in the editor with the following values:

CPU=0.01s
IO=0.004s
NL=1s
n=2

TT (empty server) = 0.1s + 2 * 0.004s + 1s = 1.02s

The important factor here is the transmission time for the output screen. Even if the disks on the server are overloaded (e.g. 0.025 s per I/O operation), the transaction time is only marginally increased to 1.06 s. A think time of 3 seconds results in an efficiency value for terminals of 75%. This is unsatisfactory.

E = TH / (TH + TT) = 3s / (3s + 1.02s) = 0.746

A doubling of the line speed in this case results in an optimal transaction time of 0.53 seconds. If it is assumed that no dilations occur during the transmission of messages, then this transaction time remains valid even when the server has a realistic workload. In this case the efficiency value for terminals rises to 85%.

These two examples show how varied the causes for inadequate performance can be. Overloading can occur in the CPU or in the peripherals, as can combinations of bottlenecks. For help in locating and solving problems like this, see section “Basic procedure”.

The formulas and examples above demonstrate the way in which users should formulate their performance expectations for online operation. Their intention is always to minimize costs. If savings (e.g. in staff and terminals) on the one hand are set off against increased costs (e.g for faster lines, faster processors, faster disks or program alterations) on the other, then one always arrives at a sound performance expectation. Any decisions reached should always be examined at regular intervals in the light of fluctuations in the price of hardware.

Of course, subjective considerations such as the psychologically damaging effect of occasional long transaction times must not be ignored completely. However, performance expectations based upon these considerations are misleading. Eliminating individual long transaction times is generally a very complicated operation and does not normally lead to overall savings.

Relative performance factor of the CPU (RPF)

In the BS2000 world the performance of a server is expressed in terms of its RPF (Relative Performance Factor). The RPF value indicates the performance capability of the CPUs under a realistic load.

The RPF is determined by measuring with two benchmark loads. An OLTP load (TP benchmark with UTM and SESAM/SQL applications) and a batch load (SKBB benchmark with COBOL programs).

A performance value is obtained for batch mode (RPF_{SKBB} using jobs per CPU second as the unit of measurement) and TP mode (RPF_{TP} using transactions per CPU second as the unit of measurement).

The result for TP mode accounts for 75% and the result for batch mode for 25% of the overall RPF value.

Here are some examples of the calculation of the RPF value (the measured values are compared with the values measured for the reference machine C40-F (RPF=1)):

Model	**RPF_{SKBB}** (RPF value with SKBB benchmark)	**RPF_{TP}** (RPF value with TP benchmark)	**RPF_{total}** (weighted mean from RPF_{SKBB} and RPF_{TP})
C40-F	1	1	1
SE330-10F	320	390	300
SE330-20	595	495	520
SE730-10D	836	560	630
SE730-100	6,560	3,680	4,400

Table 1: Examples for the RPF value

> It is not ideal to express the performance behavior of a server in a single figure. As you can see from the table above, the performance depends to a certain extent on the application profile.
> Due to the particular architecture and the load-dependent efficiency of the CISC firmware (see "Special features for x86 servers"), the performance of x86 servers is more heavily dependent on the application profile, resulting in a wider performance bandwidth.

Multiprocessor systems

Multiprocessor systems consist of 2 to 16 central processing units and one or more I/O processor units. The advantages lie in increased availability and a corresponding improvement in performance.

Efficient use of multiprocessor systems requires multiprocessor-capable applications. This means that the application must be able to process the requirements not only in parallel, but also without any significant mutual impairment.

Increased availability

Given appropriate hardware redundancy, whenever one of the hardware components breaks down, the system performs automatic reconfiguration without interrupting the current session.

Improved performance

In multiprocessor mode, i.e. under the control of **one** BS2000 operating system, a number of tasks can be processed in parallel (depending on the number of CPUs). To ensure that this increased performance capacity can be utilized, there must always be enough tasks available for parallel processing.

It should be remembered that each task can only make use of the capacity of **one** CPU. This means that improvements in response time behavior are limited, the only possibility being the reduction by the other CPUs of the wait times for CPU allocation. The same applies to batch applications: the runtime of a batch application can only be significantly reduced by parallelizing the pure computing requirement.

Note that the amount of CPU time used for a task is (with the same work) generally somewhat higher than on the corresponding uniprocessor.

For details, see section “Scaling behavior of multiprocessor systems”.

Special features for x86 servers

The sections below deal with the x86 server components which have an effect on the CPU performance.

CPUs

The CPUs available are split logically into CPUs for applications under BS2000 (BS2000 CPUs), and CPUs on which BS2000 I/Os and administration tasks for the x86 server are handled by X2000 (I/O processors).

Depending on the model, up to 16 CPUs can be used as BS2000 CPUs. 25% of the available CPUs are used as I /O processors.

A special X2000 firmware layer supports execution of the BS2000 operating system and the compatible execution of /390 applications on the BS2000 CPUs, and also implementation of input/output operations on the I/O processors.

CISC firmware, JIT390

The CISC firmware (CISCFW) is the firmware component for mapping nonprivileged /390 code to x86-64 code. It complements x86-64 mode and allows existing /390 code to be run on an object-compatible basis (synonym in /390 mode: compatibility mode) on x86-64 hardware.

The CISC firmware contains the component JIT390, a Just-In-Time /390 code compiler which converts the /390 code to x86-64 code at execution time. A code block is compiled only when it is executed and stored in a task-local JIT buffer. When the code block is executed again the code which has already been compiled and optimized is executed directly from the JIT buffer. Reusing predefined and optimized code sections is considerably faster than renewed interpretation and at the same time places less of a load on the processor. The JIT buffer is created as resident memory for the purpose of further optimization.

Existing (nonprivileged) customer applications still run on an object-compatible basis in /390 code using the CISC firmware. This applies both for applications which were generated with ASSEMBH and for programs generated with the BS2000 compilers for higher-level programming languages.

The effective performance of a server with x86 architecture depends on the number of system calls which occur within a /390 application, and on how greatly the /390 application profits from the efficiency of the CISC firmware.

System calls are processed in the server's TPR or SIH processor state (x86-64 mode). The applications run in the TU processor state (/390 mode).

The efficiency of the CISC firmware thus depends on how the JIT buffer is used by the load concerned: on the one hand on the repetition rate of specific /390 code parts (low CPU requirement), and on the other hand on the size of the program parts of a /390 application translated into x86-64 code (higher load on main memory and main memory cache).

CPU performance

The nominal RPF values (see “Standard values for x86 servers”) do not always allow reliable conclusions to be drawn with regard to runtimes, transaction rates or CPU time consumption. Depending on the application, these values can be higher or lower.

The RPF values measured for an x86 server apply for applications with a TU percentage of 40 - 50%. In the case of applications with a TU percentage outside this range, the performance of an application can deviate from the nominal RPF values.

See also the section “Migration from /390 servers to x86 servers”.

Characteristic main memory values

In particular for TP operation, a paging rate that is too high is critical (see the maximum recommended number of paging I/O operations in the section "Standard values for BS2000 servers"). Too high a paging rate always results in unsatisfactory response times. If the critical number of paging I/O operations is repeatedly reached or even exceed, the main memory must be expanded.

In VM2000 operation the main memory should be distributed as efficiently as possible over the guest systems before a general upgrade of the main memory configuration.

Monitoring main memory utilization

The utilization of the main memory should be continuously monitored by systems support using SM2 in order to detect an impending main memory bottleneck in good time.

The following key figures are of significance for calculating and evaluating the memory utilization (see also section "Control functions within the framework of main memory management" (Activating/deactivating tasks)):

- NPP (Number of Pageable Pages): Corresponds to the size of main memory which is available minus the number of resident pages.
- SWS (System Working Set): Number of used and accessible pageable pages (corresponds to the NPP minus the number of freely usable pages in the so-called free pool).

> The closer you are to a memory bottleneck or the critical paging rate, the more accurate is the result of the evaluation method recommended here.

The utilization level of main memory (in percent) is obtained using the following formula:

N = SWS * 100 / NPP %

Measurements for N up to 75% are regarded as uncritical.
When the values for N exceed 90% a main memory upgrade should be considered. This is particularly the case when the new version of a software product is to be introduced and this software is used by several tasks.

Special features for x86 servers

The main memory is distributed between BS2000 and domain 0 (Dom0) or X2000 (I/O processors).
A recommendation for the main memory configuration of x86 servers which is appropriate for most applications is shown in section "Standard values for x86 servers". The x86 server standard models offered are delivered with this recommended memory configuration.

In most cases the default settings for main memory management do not need to be changed. Expansion of the main memory needs to be considered in the following cases:

- Expansion of the entire main memory

  When multiple guest systems are used, the size of the entire main memory must be selected to ensure that the total main memory requirement of all guest systems which operate in parallel does not exceed the total main memory which is available. The main proportion of memory for Dom0/X2000 must also be taken into account here, see also the example in section "Standard values for x86 servers".

- Main memory size of BS2000

  For each VM with a BS2000 guest system at least 512 MB of main memory is required for the guest system and micro kernel together. The main memory required by the applications is not included in this figure. The total main memory required for a guest system depends largely on the applications and the performance required, Use of an extremely large number of or of extremely large tasks or processes always calls for a larger main memory configuration.

- JIT buffer

  The JIT buffer requirement for the JIT compiler must be borne in mind. By default, 40% of the BS2000 main memory is reserved for this purpose. This memory is not directly available to the operating system and applications.
  Further information is provided in section "Controlling buffer sizes of the x86 hardware (CISC)".

Characteristic peripheral values

In the case of interactive applications, the focus is on I/O operations on disks (more precisely: logical volumes).

The following aspects are of importance for measuring and evaluating the peripheral performance:

- Time slices for I/Os to disk
- Hardware service time
- Software service time

Time slices for I/Os to disk

At the **logical** level, the user formulates his/her I/O requests in the application program using macros (GET, PUT, GETKY, STORE, PAM, etc.). When higher programming languages are used, the READ or WRITE statements are converted accordingly by the relevant runtime system.

The expansion of the macro contains an equivalent SVC call which, however, is not executed with every macro.

In the case of sequential input operations (GET), the access method (SAM, ISAM) provides the next virtual record from the input buffer into which the data block was physically read. The SVC call is executed only when there are no further records in the input buffer.

In the case of sequential output operations (PUT), the access method (SAM, ISAM) performs the SVC call only when there is no space left in the output buffer.

Before an update operation PUTX (ISAM) can be performed, the corresponding records must have been read (GET, GETKY). Only when the record length increases in the course of the update will the SVC call be executed.

If the called DMS access method discovers that a physical I/O operation is to be performed, the SVC call $EXCP (Execute Channel Program) or $XCPW (Execute Channel Program with Wait) is used to trigger the I/O operation. These calls are not immediately available to the user.

The time relations on the channel or disk side are identical for both calls; the only difference is the reaction of the calling program routine.

With the $XCPW call, the program run is interrupted (placed in the wait state from Q0 to Q4 by the system-internal PEND routine) and will not resume until the I/O operation has terminated.

> ¡ The Q4 queue is simply a “collector” for tasks waiting for execution or for termination of I/O operations (and bourse events).In general, the length of Q4 does not necessarily reflect an I/O bottleneck. The important queues in this respect are those in front of the individual I/O devices (device queues, see also the report group DEVICE of openSM2).

With the $EXCP call, the program continues asynchronously to the processing of the I/O operation (performs other functions unrelated to the I/O operation). At a given time, it issues an explicit wait call. For the sake of simplicity, only the $XCPW call is presented in more detail below.

If the logical volume is occupied at the time of the $XCPW call (busy processing I/O operations for other users), a wait time is incurred in front of the volume. A higher volume workload causes a longer wait time (see section "Software service time").

When the PAV function is used, there are several device addresses (alias devices) available for each logical volume (base device). Concurrent I/O jobs can be processed via these, which prevents excessive delays.

As soon as the volume is free (or at least a free alias device is available), the operating system sends the instruction START SUBCHANNEL (SSCH) together with the “Operating-Request Block” (ORB), which contains a description of the possible access paths in the “Logical Path Mask”, to DCS.

If all access paths are occupied (which is very rare with multiple access paths), a corresponding wait time is incurred (function pending time: cf. the report group SERVICETIME of openSM2) before the I/O operation can be initiated. As a rule, the wait time for a free access path is so small as to be negligible.

ℹ On x86 servers, actual handling of the input/output (triggered by the command that is equivalent to SSCH) is performed by the I/O drivers of X2000 on a separate CPU.

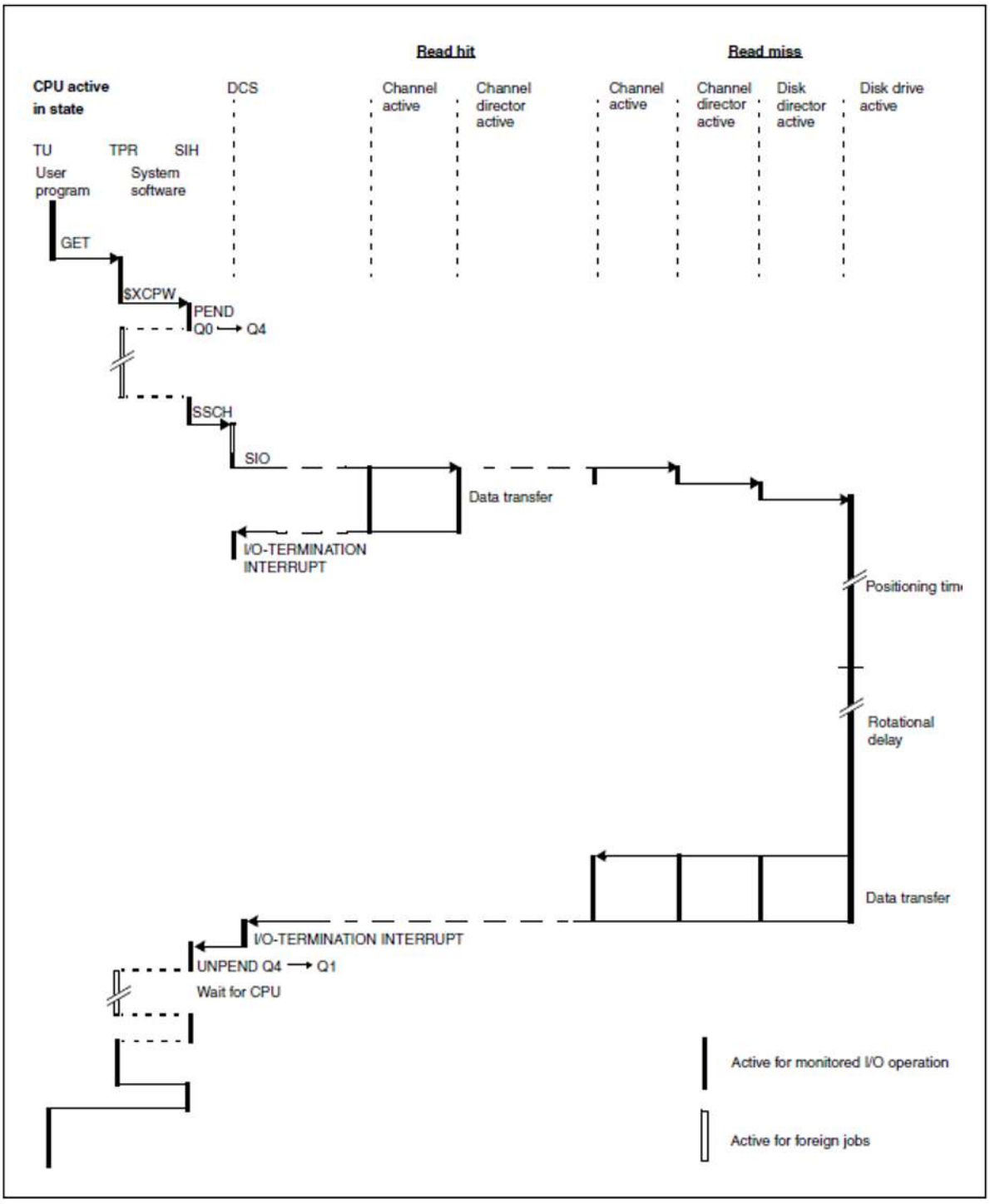


Figure 3: Flowchart for a disk I/O operation: Read hit/read miss

Hardware service time

The hardware service time refers to the time from the START SUBCHANNEL (SSCH) instruction through to the I/O termination interrupt.

In the case of storage systems, the hardware service time varies greatly depending on whether a cache hit or a cache miss is involved.

In the case of inputs/outputs with a block size of 4 KB, a typical value for a cache hit is below 1 ms. In this case the value for a cache miss is up to 10 ms.

Software service time

The software service time per I/O operation which the user sees mainly comprises:

- the wait time ahead of the volume concerned and
- the hardware service time for performing the physical I/O operation (see DISK report group of openSM2: SOFTWARE-DURATION and HARDWARE-DURATION).

The wait time W ahead of the volume is derived according to the rules for queue formation depending on:

- the distribution of the times between the I/O requests (interim arrival times),
- the distribution of hardware service times and
- the volume utilization.

Without delving further into queue theory, the following two formulas can be used to estimate the average wait time W:

1. When assuming the prerequisite "M/M/1" (worst-case scenario):

   M: Exponential distribution of interim arrival times
   M: Exponential distribution of hardware service times
   1: one console

2. W = S * U / (1-U)

   S: Average hardware service time
   U: Volume utilization

   Example

   S = 6 ms; U = 30%
   W = 6 ms * 0,3 / (1 - 0,3) = 2,6 ms

   When the volume utilization is 30%, the I/O requests must wait an average of 2.6 ms before the volume is released.

3. When assuming the prerequisite "M/D/1" (best-case scenario):

   M: Exponential distribution of interim arrival times
   D...: Constant hardware service time
   1: 1 console

   W = S * U / (2 * (1-U))

A comparison of the two formulas shows that, given a constant hardware service time (M/D/1), the average wait time W is only half as long as with exponential distribution of the hardware service time (M/M/1).
In practice neither a strictly exponential distribution of hardware service times nor a constant service time occurs. The wait times which occur as a result of the volume utilization are around the average of the results of the two formulas.

The table below illustrates the factor by which the hardware service time is dilated irrespective of the volume utilization.

Volume utilization (%)	dilation factor M/M/1	dilation factor M/D/1
10	1.11	1.06
20	1.25	1.13
30	1.43	1.21
40	1.67	1.33
50	2.00	1.50
60	2.50	1.75
70	3.33	2.17
80	5.00	3.00
90	10.00	5.50

The wait time ahead of the volume constitutes a major part of the I/O time. The utilization of volumes which are shared by multiple users should not exceed 30%.

Objects relevant to performance

This chapter describes the following objects which are relevant to performance:

- Storage systems
- Disk organization and accesses from the BS2000 viewpoint
- Network connection
- Net-Storage
- Virtual machines

Storage systems

This section describes the following aspects of storage systems which are relevant to the performance:

- Connection to the server
- Properties of storage systems
- Load types
- Cache behavior with different load types
- RAID configuration

Connection to the server

This section describes the connection of storage systems for the various server types.

Properties of the type FC channel

Data transfer is serial via a fiber-optic cable.

The essential features of the type FC channel are:

- full-duplex operation
- a maximum of **eight** I/O operations (of different control units) can be executed simultaneously
- connection of FC devices possible via FC switch (HNC and SKP can also be connected directly
- maximum distance of an FC switch: 550 m
- distances of up to 100 km are bridged without any major performance losses
- only disks with FBA formatting are supported.

The following storage systems and devices are supported via the type FC channel:

- ETERNUS DX and AF
- ETERNUS CS8000
- MTC devices LTO-x in the Scalar MTC archive systems
- High-Speed Net Connect HNC

- Service/console processor SKP

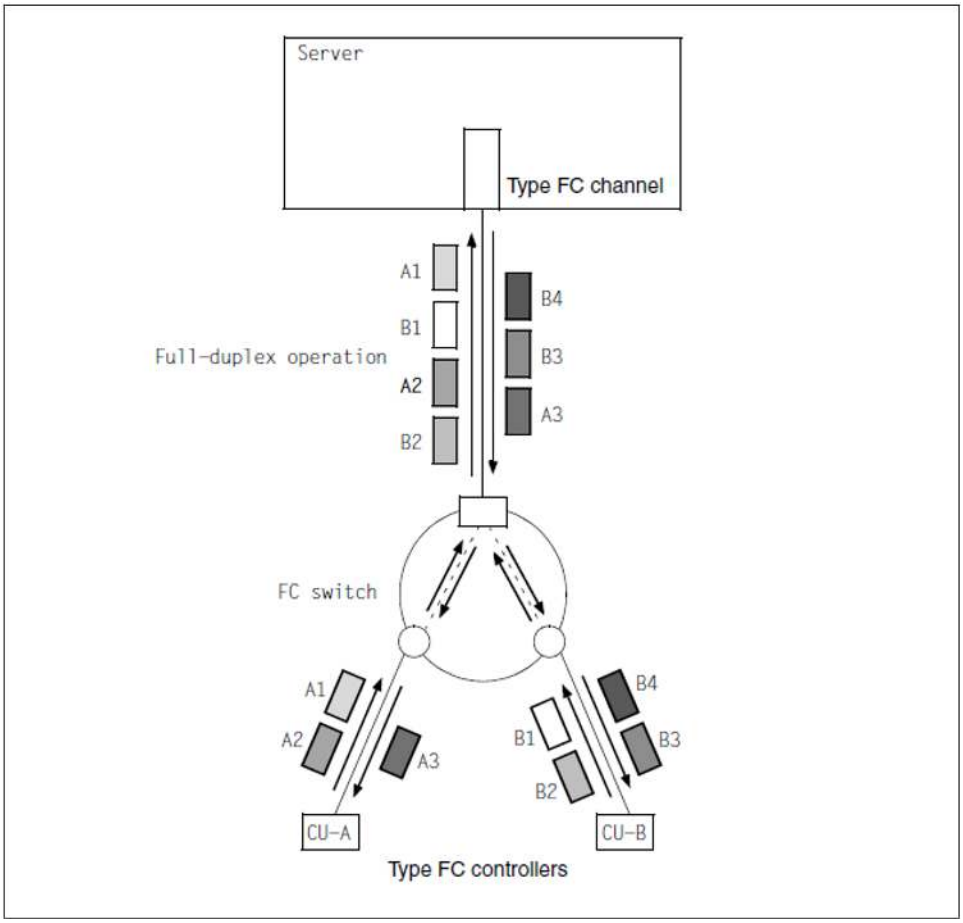


Figure 4: Type FC channel

Storage connection to /390-servers

SE-servers with SU /390 are connected to the storage via a FC-Switch and 16 Gbit/s or 64 Gbit/s channels.

Storage connection to x86-servers

There are different ways to connect x86-servers to storage:

- local SAS-connection:
  SAS-connection of the internal system-volumes (not approved for use with BS2000) and/or a rack-mounted storage-system ETERNUS JX.
- Fibre Channel (FC-peripherals, e.g. ETERNUS DX/AF)
- Ethernet (Net-Storage)

All inputs/outputs to any peripherals are routed via the I/O-processors (X2000). The I/O-processor performance is very good and more than sufficient even for x86-servers under extreme loads.

The corresponding interfaces are managed within the X2000. The peripherals are connected via PCIe-Controllers.

SAS-connection

Direct Attached Storage (DAS) like the ETERNUS JX-series and its fast SAS-connection extends the storage-capacity of x86-servers.

When making use of the optional SAS-RAID connection with Backup Battery Unit (BBU) of the ETERNUS-JX system, the write-cache is going through a BBU relearn Cycle every 30 days. The BBU relearn cycle can take up to 15 hours and fully charges and discharges the BBU several times. During this time the write-cache is unavailable and without effect. For some workloads this can lead to extended run and response times. The BBU relearn cycle can be configured individually to for example only run during weekends.

FC-connection

X86-servers can use FC PCIe-Controllers with connection speeds of up to 64 Gbit/s.

Properties of storage systems

This section describes the following topics:

- Components of the storage systems
- Replication: volume-based mirroring for storage systems
- Thin Provisioning

Components of the storage systems

The most important components of the storage systems which are relevant to performance are:

- Cache
- Physical disks
- Controllers

Cache

The component of a storage system which most influences the performance is the cache. Each input/output request is subject to caching. The hardware service time (I/O time) depends largely on whether the data is located in the cache or must first be read from or transferred to the disk.

If the data block which is being searched for is not contained in the cache, it must be read from the disks (read miss). Such a read miss takes considerably more time than a read hit, in the case of which the data is already in the cache. When sequential processing is recognized, the data is read ahead in order to prevent a read miss.

Since the cache of the storage systems is protected against a power failure by batteries, write requests are regarded as completed when they have been written to the cache and verified (write hit rate = 100 %). A write I/O operation therefore takes almost as long as a read hit. Writing to disk takes place asynchronously to the I/O operation. If not enough space is available in the cache, so-called “delayed fast writes” occur, i.e. the cache contents must first be saved before new data can be read into the cache (write hit rate < 100 %). This typically occurs when large quantities of data are reconstructed.

The hit rate is the share of cache hits in the total number of inputs/outputs to/from the storage system. Sometimes a distinction is also made between the read hit rate and the total hit rate. The total hit rate contains the cache write hits. Only the read hit rates are examined in this manual.
Vendors of external storage systems often specify the “total hit rate”, which also contains the cache write hits and is therefore higher than the read hit rate examined here.

> The performance data for external storage systems sometimes only specifies the total hit rate, which also includes the cache write hits and is therefore higher than the read hit rate examined here.
> However, the read hit rate can be determined using the ratio of write and read IOs, provided there is no bottleneck and a 100% write hit rate can be assumed. The write IO times provide the starting point for this calculation.

A high hit rate (> 90 %) can also be achieved for read requests by using large caches. A read hit rate of 80 % or higher is normally regarded as good performance for a storage system.

Large cache configurations in the ETERNUS DX/AF storage systems are urgently recommended for performance-critical applications.

Physical disks

The following are known for physical disks:

- storage type (HDD/SSD)
- storage capacity (in Gbyte, e.g. 300 Gbyte)
- rotational speed (in “rotations per minute”, e.g. 15,000 rpm).

The following are also decisive for the performance of a disk:

- internal connection with connection type (e.g. Fibre Channel, SAS, Serial ATA (SATA)), nearline SAS (NL-SAS)) and data transfer rate (e.g. 8 Gbit/s)
- cache on the drive (drive cache)

Generally the user only sees the performance of the physical disks in the case of a read miss.

From the BS2000 viewpoint no restrictions exist for using the disk sizes offered today.

> ¡ The data of performance-critical applications should **not** be placed on SATA or NL-SAS disks.

Controllers

The controllers of the storage systems process the server’s I/O requests, control cache access and handle access to the physical disk drives.

The number of controllers and the number of paths to the servers which are controlled and to the internal disks depend on the model.

Remote Replication for storage systems

Storage systems offer functions for remote replication between geographically separated storage systems.

The software product SHC-OSD is the BS2000 host component for storage systems. It provides information services and commands for controlling the replication functions of the storage systems. A detailed description of the replication functions is contained in the “SHC-OSD” manual [29 (Related publications)].

Performance behavior in the case of synchronous remote replication

If a copy of the data is kept in a remote second storage system, each write access is entered first in the local cache and then in the "remote cache".

This means that using synchronous remote replication at least doubles the hardware service time for write accesses. An increased protocol time for processing the function also has to be taken into account.

Another major factor is the distance between the locations. Double the runtime of light in glass (outward and return directions) must be added in each case.

The write hit rate is usually 100% (provided there are sufficient “breathers” for saving the cache contents), i.e. each write access is performed quickly. With small blocks, the doubling of the hardware service time is normally of no consequence, but with large blocks (e.g. copy operations) there is a clear increase in the hardware service time.

An additional increase in the hardware service time can occur if the performance capability of the remote data links is insufficient.

> ¡ In the case of asynchronous remote replication, only the “protocol time” is accrued. The input/output is therefore delayed considerably less.

Thin Provisioning

The thin provisioning function permits the capacity of storage systems to be used efficiently. Devices (LUNs) with preconfigured virtual capacity are offered to the application, while internally the storage system provides the necessary physical capacity. Thin provisioning requires a corresponding configuration in the storage system.

SHC-OSD supports thin provisioning for the storage systems ETERNUS DX S2 and higher. SHC-OSD has monitoring and information functions for thin provisioning. A detailed description of the thin provisioning function is contained in the “SHC-OSD” manual [29].

Load types

The configuration of the disk peripherals is influenced both by both the load profile and by the time requirements of the application. In practice there are essentially two load types, which can be characterized as follows:

	OLTP processing (Database applications)	**Batch processing, data backup or other applications with large IO size**
Typical IO size	"small": 2 KB or 4 KB	"large": 128 KB or greater
Performance indicator	IO rate (IO/s)	Throughput (MB/s)
Typical IO pattern	Random access	Sequential access

In general, the applications in BS2000 can use IOs with block sizes of up to 160 KB for each I/O operation. In the case of NK formating, certain disk models even permit block sizes of up to 480 KB. This enables the throughput to be increased significantly.

Depending on the application type, different performance targets are focused on.

- High throughput is aimed for in the case of data backup and in batch mode.
- In OLTP mode, good response times are expected.

The I/O operations play an important part in the throughput rate/dwell time and the response time/transaction time (see section Time slices for I/Os to disk).

Standardized load types

In this manual the performance observations relate to the following standard load cases:

- "Sequential Write", "Sequential Read". This corresponds to throughput-oriented loads in the case of data backups or batch processing.
- "Random Read", "Random Write". This corresponds to IO-oriented OLTP loads, where response time is critical.
- "Random25" random accesses with a proportion of 25% write and 75% read I/Os. With small blocks this case reflects OLTP mode.

While "Random25" is a load which is close to practical application and is interesting above all for measurements, the theoretical information in this chapter is restricted to pure reads or writes.

Cache behavior with different load types

The section below is based on measurements and data from current Fujitsu ETERNUS DX systems. Other systems can display different behavior with respect to their cache management and algorithms. However, most statements are of a fundamental nature and can also be applied to other systems.

An area in the cache of an ETERNUS DX system is reserved for write jobs. The remaining cache is available for caching read requests. The size of these two areas is managed dynamically and can be between 0% and 100% of the total cache, depending on the situation.

While write IOs can always be handled as cache hits (except in the case of an overload), it is more critical to achieve a high bit rate when reading. The behavior and efficiency of the cache and disk utilization resulting from this for various loads are described in terms of quality.

Sequential Read

When an ETERNUS DX system recognizes a Sequential Read, the read-ahead mechanism comes into effect. The following data which has not yet been requested by the client is read and stored in the cache. Provided no parallel loads cause a significant disturbance, this achieves a 100-percent hit rate. This also applies when a number of read operations are executed in parallel. It is consequently generally no problem to reduce the time slot for backup runs by parallelizing the operations (provided the RAID groups can supply the total throughput and a heavy productive load is not already imposed on the cache).

When a suitable RAID configuration is provided, the highest throughputs can be achieved with this load. In terms of cache (and also disk) utilization, Sequential Read is the optimal application.

Sequential Write

In the case of Sequential Write the data is practically written sequentially immediately from the cache to the disk drives. Here, too, a number of operations can run in parallel without any problem - without having a negative influence on cache utilization. Provided the performance of the RAID groups concerned is sufficient, the same throughput is achieved as with a Sequential Read. However, depending on the RAID level, the load on the disk drives is greater than when reading, which means that bottlenecks can occur earlier in the event of heavy loads. Details of this are provided in section RAID levels and their performance.

If the disk drives can no longer cope with the load, the write area of the cache will become full, and Delayed Fast Writes will occur (see section Components of the storage systems). This can be recognized from a write hit rate of below 100%. As this will probably occur with Random Write, the effects are described in the associated section.

Random Read

In the case of Random Read with large volumes of data, the performance is influenced negatively by cache misses. Here the read hit rate is determined by the characteristics of the load, size of the cache, and utilization of the cache by competing loads.

In ETERNUS DX systems, read and write hit rates can be determined separately at RAID group level. If only the total hit rate is available, take into account the relationship between read and write IOs when you evaluate it as normally cache misses are restricted to read IOs.

Parallel random loads compete more strongly for the cache (in the case of parallel loads in the same RAID group also for disk accesses) than sequential loads and potentially have a greater influence on each other when the throughput is the same. When the disk drives are approaching high utilization, the negative effects on the IO times are considerably higher than with sequential access. Observing the disk utilization enables the threat of bottlenecks to be recognized before they become apparent in appreciable increases in the IO times.

Random Write

In the case of Random Write (in contract to Sequential Write), the data is not written immediately to the disk drives, but initially only stored in the cache. When the load is low, the data is retained as long as possible in the cache to prevent duplicated writing if the data is modified again. Above all in the case of small IOs it makes sense to wait until a number of consecutive blocks have been modified in order to write them jointly to the disks and thus relieve the load on the disk drives. However, too much cache may not be reserved for the read IOs. The actual disk utilization is therefore largely dependent on the cache size and cache utilization. Management of the write and read percentages in the cache and the decision regarding whether and when data is written from the cache to the disk drives is the responsibility of the storage system's algorithms and as a rule cannot be influenced.

When data is finally written from the cache to the disks, depending on the circumstances disk utilization of 100% can occur. This is uncritical when:

- the performance of the RAID group is sufficient to write the data from the cache quicker than new requests are received by the cache. The disks are heavily utilized in the background in order to process the accumulated write operations promptly. Provided this status only exists for a short time, this is not a certain indication of a bottleneck.
- parallel read requests, also to other volumes of the RAID group, are not disturbed. This is ensured as far as possible by the storage system. In the case of high disk utilization it is recommendable to consider the read IO times to determine whether these actually impair the performance.

Only when the incoming write requests exceed the capability of the RAID groups concerned is continuously more write cache required. If a cache bottleneck then occurs, the performance is impaired in various respects:

- Delayed Fast Writes occur, t.e. cache contents must first be saved before it is possible to store new data in the cache. This results in increased write IO times. The write hit rate decreases. A write hit rate below 100% is always a sign for an acute overload.
- A high requirement for write cache automatically leads to a smaller available read cache. The read hit rate decreases, and increased IO times also occur there - also in RAID groups which were initially not affected.
- In such cases, writing the data onto the relevant volumes has highest priority in order to relieve the cache situation. Competing read IOs (also to different volumes of the same RAID group) are significantly impaired by this and, in extreme cases, can require times which are increased by a factor of 10-100.

In extreme cases the effects can also continue for a few minutes beyond the high load situation until the write jobs accumulated in the cache have been processed.

You are certainly recommended to avoid cache bottlenecks and to provide sufficient write performance for any volume on which a heavy load is imposed by Random Writes. Relevant details and recommendations are provided in section "RAID levels and their performance".

RAID configuration

This section describes the aspects of the RAID configuration of a storage system which are relevant to performance insofar as they are important from the BS2000 viewpoint.

RAID groups LUNs, and volumes

RAID (Redundant Arrays of Independent Disks) is the name of an architecture for providing fault-tolerant and high-performance volumes and LUNs. Here a number of individual disk drives are combined to form a transparent network, the RAID group. So-called Logical Unit Numbers (LUNs) are defined for these RAID groups. It is typical that multiple LUNs (logical volumes) are configured on the same RAID group. For information on the effects of logical volumes on the performance, see “Logical volumes and physical disks”.

All ETERNUS DX/AF storage systems are high-availability systems which not only comply with the RAID standards originally defined by Berkeley University, but also meet all the availability levels specified by the RAID Advisory Board. They differ only in their configuration options. The following RAID levels are usual in the BS2000 environment:

- RAID 1 (full mirror)
- RAID 1/0 (striping + full mirror)
- RAID 5 (striping + parity mirror)
- RAID 6 (striping + dual parity mirror)

RAID levels and their performance

The utilization of the disk drives is influenced by various factors. The number of physical IOs and the effort required for each IO are decisive. While the former can be reduced by an adequately dimensioned cache, the load caused by the accumulated IOs depends largely on the RAID level.

RAID 1

Two identical disk drives in each case, original and mirror disk, form a RAID 1 group (referred to below as “RAID 1 (1+1)” for short).

The data these contain is identical. Consequently only 50% of the disk capacity is available to the user.

Mirroring does not discernibly affect the write performance. In the case of read jobs, the storage system attempts to optimize and read from the more convenient disk, and the load is thus distributed to both disks. While the performance does not profit from this in the case of a single user task with synchronous read, under some circumstances a significant benefit can be achieved in the case of parallel loads.

RAID 0

With RAID 0 the data is distributed over multiple disks by means of "data striping". In this case the maximum throughputs can be scaled almost with the number of disks. In contrast to RAID levels with parity information, a high write performance is also achieved.

> **CAUTION!**
>
> RAID 0 offers no failsafe performance, and is not recommended outside test scenarios. If high write performance is needed, RAID 1/0 should definitely be used instead.

RAID 1/0

RAID 1/0 combines the better performance through striping (RAID 0) with the redundancy of a mirror (RAID 1). A RAID 1/0 group therefore consists of multiple (generally 2 to 4) disks to which the original data is distributed by means of “data striping”. Plus the same number of disks containing the mirrored data (referred to below, for example, as “RAID 1/0(4+4)” for short). As with RAID 1, only 50% of the installed disk capacity is available to the user.

RAID 5

RAID 5 offers a good compromise between performance and capacity. Depending on the application scenario, however, a RAID 5 can offer sufficient performance provided all write IOs can be handled via the cache. RAID 1/0 should be preferred for write-intensive loads or data with high availability requirements.

RAID 5 implements common parity checking for a number of disk drives. As with RAID 1/0 the data is distributed in blocks over the disk drives of the RAID 5 group using “data striping” and saved with the parity information, which is also distributed over all the drives (“rotating parity”). Parity checking reduces the capacity of the network by the size of a disk drive. In the case of the frequently used configuration with 4 drives, consequently 75% of the installed disk capacity can be used (referred to below as "RAID 5(3+1)" for short).

But as with RAID 1/0, thanks to “data striping” RAID 5 offers advantages in the event of parallel access. When one disk fails and in the subsequent rebuild operation, however, only downgraded performance can be reckoned on.

When writing, the parity information must also be calculated and written. For this purpose the previous value of the block to be modified is read out before the actual write, i.e. this causes addititon read IOs. Consequently RAID 5 is somewhat more complicated and has longer write IO times than RAID 1/0.

RAID 6

As with RAID 5, the data stream is divided into stripes. However, not just one but two error correction values are calculated. Data and parity information is thus distributed over the disks in such a way that both sets of parity information are contained on different disks (“dual rotating parity”).

In RAID 6 up to two disks can fail. The effort for resynchronization, in particular when two disks have failed, is, however, considerably higher than with RAID 5, as is the effort for write IOs, too.

Compared to RAID 5, RAID 6 offers higher failsafe performance with lower write performance.

Examples

The examples below are designed to permit a comparison of the effects of the various RAID levels on the load of the single disk drives with the same disk capacity or number of disks.

RAID level	Capacity	HDDs	Operations for each physical disk (statistical) with 40 read IOs	Operations for each physical disk (statistical) with write IOs
Single disk	K	1	40x Read	40x Write
RAID 1	K	2	20x Read	40x Write
RAID 1/0	2*K	4	10x Read	20x Write
RAID 1/0	3*K	6	7x Read	13x Write
RAID 1/0	4*K	8	5x Read	10x Write
RAID 5	3*K	4	10x Read	10x Read 20x Write
RAID 5	4*K	5	8x Read	8x Read 16x Write
RAID 6	3*K	5	8x Read	16x Read 24x Write
RAID 6	4*K	6	7x Read	13x Read 20x Write

Disk organization and access from the BS2000 viewpoint

This section describes the following topics:

- Logical volumes
- Pubsets
- Access methods
- Processing mode

Logical volumes

On current storage systems, individual disks are only visible to a certain extent. BS2000 sees “logical volumes” which are assigned to the LUNs (Logical Unit Numbers). The assignment of LUNs to physical disks is largely hidden from BS2000. The storage systems ensure efficient operation by means of RAID levels.

“Large” logical volumes can also be used. These are volumes which are more than 32 GB in size, see the manual “Files and Volumes greater than 32 GB” [3 (Related publications)].

Logical volumes and physical disks

From the BS2000 viewpoint logical volumes are independent units on which I/O operations can be initiated in parallel. When the current high-capacity disks are used, multiple logical volumes are normally located on a disk drive.
When I/O operations on logical volumes located on the same disk drive are initiated in parallel, the hardware service time of magnetic disks is inevitably extended. Whether the extension of the hardware service time leads to an appreciable performance loss depends on the cache hit rate and the utilization of the disk drive – possibly also by users outside BS2000. SSDs are significantly better in handling parallel IOs. A significant extension of IO times is only expected when the overall load exceeds a certain limit.

Nowadays, storage space is usually provided via thin provisioning. The distribution of the logical volumes for BS2000 to the physical disks within a configured thin provisioning pool is managed by the storage system.

Requirements for configuring storage systems

From the BS2000 viewpoint, the disk peripherals are subject to different requirements:

- capacity
- throughput
- reaction times
- parallel use

Although the need for manual optimization is reduced when using thin provisioning and the algorithms of the storage systems ensure an optimal distribution of logical volumes onto physicasl disks, there are still performance relevant factors to be considered when selecting and configuring a certain thin provisioning pool to select a suitable

- disk type
- and RAID level
-
-

Data format

It is recommended to use NK2 data format. Compared to K data format, in which the key information is also included, NK data format makes better use of the disk capacity and the cache memory, thus achieving shorter hardware service times.

Use of SHC-OSD

You can ascertain the following with SHC-OSD:

- the RAID group in which a BS2000 volume resides
- which other volumes reside in the same RAID group

The drive workload on ETERNUS DX/AF cannot be determined using BS2000 resources and can therefore not be measured using openSM2. It consists of the sum of the I/O loads of the relevant logical volumes.

Pubsets

Use of SF pubsets

When running applications with certain input/output requirements, it makes sense to group similar public volumes together in pubsets.

One pubset should comprise 4 or less public volumes. The advantage of this is that, in the event of a failure, only the pubset concerned is blocked, and operation can continue to a limited degree with the remaining pubsets. Creation of a standby pubset (a copy of the home pubset) for the purpose of a quick restart is recommended (for a description of handling see the “Introduction to System Administration” [10 (Related publications)]).

Using a number of SF pubsets improves performance by increasing the range of distribution options for files.

Notes on the distribution of the TSOSCAT system files, paging areas and SYSEAM are given in section “Creating system files”.

In order to distribute the input/output requirements of users who work under different user IDs by means of system specifications, the following parameters must be taken into account when /ADD-USER or /MODIFY-USER-ATTRIBUTES is used:

```
/ADD-USER USER-IDENTIFICATION=xx, DEFAULT-PUBSET = *HOME / <catid> 1.
    PUBSET = *HOME / <catid> 2.
    PUBLIC-SPACE-LIMIT = <max>/<integer> 3.
```

1. <catid> identifies the pubset in which the user can access files without the specification of a catalog ID, e.g. in /CREATE-FILE, /ADD-FILE-LINK, /MODIFY-FILE-ATTRIBUTES.
2. <catid> defines the pubset which can be accessed by the user. It is only possible for a user to access a particular pubset if an entry exists for him/her in the user catalog of the relevant pubset.
3. <max> specifies the maximum public space available to this user ID for the pubset specified in 2.

One exception is when the pubspace value for a pubset is specified as zero for a given user ID. This user then has access to all the files whose access rights are set appropriately. He/she cannot, however, create any files in this pubset.

Use of SM pubsets

Whereas the formatting attributes of a volume set, which are defined at initialization, remain unchanged for as long as it exists, the availability and performance attributes can be changed in the ongoing operation. Only the performance attributes are dealt with below.

SM pubsets from the point of view of the user

So called storage services can be requested for a file using /CREATE-FILE or /MODIFY-FILE-ATTRIBUTES (STORAGE-CLASS operand) or the FILE macro (STOCLAS operand), i.e., the file attributes which are relevant for the storage location:

Operand	Meaning
`STORAGE-CLASS = *STD`	Assigns the default storage class from the user catalog of the SM pubset.
`STORAGE-CLASS = <name>`	Assigns a specific storage class.
`STORAGE-CLASS = *NONE`	There is no explicit assignment of a storage class. The user defines the performance attributes (*STD / *HIGH / *VERY-HIGH).

The user can obtain information on the names and attributes of the available storage classes (i.e. access must be permitted when GUARDS protection is set) with /SHOW-STORAGE-CLASS.

The user can get information on the storage services of the existing volume sets with /SHOW-PUBSET-FILE-SERVICES and thus check whether there is a volume set with the desired attributes for his/her file.

The storage services can also be requested by means of direct attributing.

SM pubsets from the point of view of systems support

For further information please refer to the chapter "Defining of the performance profile" within the manual "System Administration" [10].

Access methods

This section describes the access methods which are used for storage systems:

- UPAM and BTAM
- FASTPAM
- SAM
- ISAM
- PAM
- Databases
- DIV

UPAM and BTAM

UPAM and BTAM are the basic access methods for disks and tapes respectively. Because they are of minor importance for small applications, they have become less important. They should always be taken into account when a processing operation is being planned. They include certain functions not available with other access methods. It is often possible to improve performance considerably by using them in preference to other access methods at no detriment to user-friendliness. UPAM, for example, permits random access using the half-page number.

The following advantages apply to both UPAM and BTAM:

- a simple interface for the application program
  There is only one action macro for DMS for each method. The action required is formulated using current parameters. Every read/write request from the program causes a physical I/O operation.
  UPAM and BTAM use shorter path lengths compared with SAM and ISAM, as the blocking and unblocking operations on the records are not necessary.
- a choice between synchronous and asynchronous I/O operations
  Asynchronous I/O operations can reduce the runtime of a program considerably
- unrestricted access to the characteristics of the storage medium; optimum utilization of these characteristics is possible
- chaining of jobs to save SVC calls and processing of files created using other access methods
- chaining of I/O operations to reduce system overhead (chained I/O).

Additional advantages with UPAM:

- random access (similar to keyed access) using the half-page number
- possibility of processing files whose contents are damaged
- shareability even when updating
- possibility of combining asynchronous I/O calls with eventing

Additional advantages with BTAM:

- the block length is freely selectable within the range allowed by the hardware
- data carriers from foreign operating systems can be processed.

FASTPAM

Essentially everything that was said about UPAM also applies to the access method FASTPAM.

However you should bear in mind the following restrictions:

- The file format must be NK4
  (BLKCTRL=NO, block size=multiple of 4 KB).
- All accesses are made via absolute block numbers.
- There are no synchronization mechanisms when two or more users access the same file.
- FASTPAM supports multisystem networks by means of shared pubsets, but not remote file access (RFA).

FASTPAM provides better performance than UPAM because FASTPAM does not run the initialization and validation routines required for I/O operations prior to each I/O, but only once, before the file is opened.

Before the OPEN, the user defines a system environment consisting of ENVIRONMENT and IO-AREA-POOL. These are system and user storage areas used again and again in file accesses.

The maximum improvement in performance is achieved by making the system areas resident. In order to do this, the user must have the appropriate authorization in the user catalog (DMS-TUNING-RESOURCES=*EXCLUSIVE-USE) and make a suitable entry in the RESIDENT-PAGES operand in the user catalog and when starting the task.

SAM

The SAM access method is designed for the sequential processing of records. It should always be selected if files are not to be shareable and random access is not necessary. This record structure is simple and yet flexible and allows both fixed- and variable-length records. Depending on the application, both types of record length can help to save memory capacity.
The fact that keyed access is not used means that path lengths are considerably shorter than is the case with ISAM. If required, it is a simple operation to convert the file into a file with a key (e.g. using an editor).

The advantages of SAM are:

- reading and writing are performed record by record
- limited updating facilities and high performance (PUTX)
- short program runtimes achieved by reading the next block in advance
- files can also be processed using PAM
- high performance achieved by eliminating shareability during updating

ISAM

ISAM is the most highly developed access method in BS2000. It offers a large number of convenient features which are achieved at the cost of increases in the memory overhead and path lengths. For reasons of compatibility and protection against obsolescence, it often happens that files are kept as ISAM files even though their type and application make them typical SAM files (e.g. source programs). Since it is very easy to convert SAM to ISAM, this is not necessary.

However, ISAM is not a pure keyed access method. ISAM functions best with a set of accesses where a keyword is followed by a series of sequential accesses (indexed sequential). The distribution and frequency of the keys and in particular whether they are changed during the life of a file, all influence the number of physical I/O operations required to read a particular record. ISAM should only be used if one or more of the following characteristics are required:

The advantages of ISAM are:

- keyed accesses which are largely independent of the distribution of the keys and of any changes which may have been made
- efficient sequential reading following keyed access
- shareability even when updating
- an acceptable deterioration in access performance even after a very high number of single-page extensions

Due to the introduction of ISAM pools, the index areas (partly also data areas) are held in the virtual memory. This makes possible a sharp reduction in the physical I/O operations with indexed-sequential accesses, and leads to marked runtime reductions.

BS2000 automates and optimizes the creation of buffer areas (NK-ISAM pools) for NK-ISAM files which are opened with SHARUPD=YES. NK-ISAM pools then no longer need to be configured.

NK-ISAM permits the use of secondary keys. If, besides the primary key, additional selection criteria per logical record are maintained in the form of secondary keys, a given set of data can be searched and processed in a number of different ways.

Compared to an application without secondary keys, the resource requirements are slightly higher as long as the “index tree” is not changed; if it is changed (e.g. GET-ELIM, GETKY-ELIM, STORE), resource requirements increase sharply.

PAM

In addition to ISAM, it is possible to imagine further keyed access methods which can better fulfill more specialized requirements. With the aid of PAM, BS2000 offers users the facility of implementing their own access methods. The following example of a hash access method shows that this is possible with a minimum of overhead.

The following characteristics are either specified or required:

- 90% of accesses should be possible with a single I/O operation
- certain frequently used blocks should be able to be read without an I/O operation even if DAB is not used
- the frequency with which changes are performed is high, but it is only very rare that records are deleted, created or significantly extended
- the distribution of the keys is known and follows simply formulated rules
- the file is **not** read sequentially
- sufficient memory is available.

Given these conditions, the number of the block containing the record can be determined from the key with the aid of a simple algorithm. The extent to which the blocks are filled is so limited that the vast majority of updates do not cause it to overflow. In the unlikely event that a block should overflow, the block contains a pointer to the next block, which in turn contains a pointer to the next block should it also overflow. Buffering frequently used blocks does not cause any problems because the user implements the algorithm himself.

Although a special conversion algorithm for converting the key to the block number must be implemented every time that a key is allocated, this process can, compared with the global ISAM procedure, produce considerable increases in performance with large files.

Databases

Databases can be seen as an important extension of the range of access methods. Both the overhead involved in installation and maintenance of the data resources as well as the access path lengths are greater with databases than with other access methods.

They do, however, offer the following additional benefits.

- the way in which data and access paths are organized can be defined by the user and adapted to suit his/her data resources
- unfinished transactions can be reset
- with the aid of logging, the state of the data resources can always be reconstructed to the state applicable shortly before a crash.
- access rights can be defined and monitored
- the amount of redundant data can be reduced as a result of the possibility of creating complex relationships between records
- Queries for multiple keys with the option of logical links permit savings on application program parts.

> The software product LEASY is an enhancement of the BS2000 access methods for TP mode. Based on ISAM, LEASY permits transactions to be reset and all the collected data to be fully restored. In addition, access according to secondary keys and a CALL interface are offered. The use of LEASY must always be considered when no database features are required outside of TP mode.

DIV

DIV (Data in Virtual) is an access method which differs from the traditional access methods such as ISAM, SAM or UPAM in that it functions without structuring the file in records or blocks, without I/O buffer and without special I/O macros (e.g. GET, PUT).

DIV processes only PAM files of the type NK4 (BLKCTRL=NO, the block size being a multiple of 4 KB).

DIV regards the file as a linear sequence of bytes. The DIV function MAP enables a file or a file area to be assigned to an area in the virtual address space. The virtual address space assigned to a file area then forms a "window" in which the pages of the file appear automatically when the customary CPU commands are used to access the corresponding pages in the virtual address space. Data modified in one window can be written to the file using the DIV function SAVE.

In the steady state the file or file area resides in the user's address space and is thus subject to paging. When the file areas are large and insufficient main memory is available, the paging rate increases correspondingly.

Performance is increased when data in the window which was read into the window by preceding accesses (via paging) is accessed repeatedly.

DIV reading

> The maximum performance enhancement occurs when data in the window is accessed; no disk access is required.
> If data access needs to be handled using page transfer, the path length per access is approx. 20% shorter than with read access using UPAM.

DIV writing

> Secure writing is implemented by writing the modified page back to the file after every write access (DIV-SAVE). In this case the path length per access is approx. 35% longer than with a UPAM write access. The extra effort can only be reduced by decreasing the number of SAVE-IOs (e.g. saving the file once only, when CLOSE is issued). If page transfer is required for write access, the path length is approx. 10% longer than with UPAM write access.

The access method has a great influence on the performance of the entire system and should consequently be selected with care.

Processing mode

The processing modes for a file are very closely related to the access methods. Although they can be newly defined each time a processing operation is carried out, they are almost invariably embedded in the processing program and hence fixed. Their influence on system performance is similar to that of the access methods and they should therefore be chosen carefully. In all three phases of file processing (OPEN, access, CLOSE), decisions are made either explicitly or implicitly concerning the choice of mode.

OPEN

Which access options are available for other file users is defined by the type of OPEN call used when the program is designed. Consequently it should be obvious that processing modes exceeding the requirements of the intended application should not be chosen. A person who only wishes to read and does not open the file with INPUT prevents other users from gaining read access. In the case of files which are of general interest, the effort of making a decision concerning reading/updating at the time of the OPEN command is worthwhile.

Opening a file is a complicated process. Because of the overhead involved, it should be done as rarely as possible. Even the requirement that a file should be locked for as short a time as possible is subordinate to this requirement. If necessary, one must resort to the access method PAM or ISAM using SHARUPD mode.

If it can be accepted that a file is not 100% up-to-date, then it is worth installing a copy for read accesses. This should be write-protected in order to prevent users from opening the copy file for writing purposes.
This solution is more efficient because using SHARUPD mode causes an increase in the access overhead. If the probability of a collision is small, then the additional effort involved here is not worthwhile.

Accesses

In this case also, the rule applies that files should only be locked if it is vital for correct operation. Performance is reduced if too much emphasis is placed on safety considerations. It is possible that a lock or even SHARUPD mode can be avoided by allowing overbooking and by separating the component parts of records.

The use of the RELSE call to force a state of consistency for an open file should be used with caution because of the physical I/O operations involved.

Keyed accesses in ISAM are more complex than sequential accesses. If necessary they can be replaced in part by sequential accesses after the program or the file has been reorganized.

If files have to be accessed in a multiprocessor system, then the pros and cons of transferring a file or parts of it in advance should be considered. This depends on the structure of the file and of the accesses. It is also of great importance whether a multiprocessor system access can be executed via the network or via disks operated in shared pubset mode and which can be addressed by several processors.

If files are being created or extended, then an improvement in performance can be achieved by avoiding the secondary space allocation.

The more compact file structure which results from this gives rise to a variety of long-term improvements in performance.

It should also be mentioned at this point that parts of a file can be supported and monitored selectively by DAB (report “Reads resp.Writes for internal area” in the report group DAB of openSM2).

CLOSE

Care should be taken that open files are always locked for certain groups of users. In addition to this, they occupy memory space and the copy on the disk is not consistent. Therefore, it is advisable to close the file as soon as it is no longer needed. This should not, however, result in otherwise superfluous OPEN calls. In the case of tape files, CLOSE involves rewinding the tape, which blocks further access to the drive concerned for some time. Consequently it is advisable to issue the call in time to prevent subsequent operations from being obstructed.

If the contents of the file have been reduced considerably and if they are likely to remain small for a long time, then its size should be reduced after the execution of CLOSE by means of /MODIFY-FILE-ATTRIBUTES (operand SPACE=*RELEASE(...) given) or FILE macro with a negative value for the SPACE parameter.

Network connection

This section describes the network connection for /390 servers and x86 servers.

Information on the performence including measurements can be found in section "Measurements for LAN connections on SE servers".

Network connection via HNC (/390 servers)

/390 servers are connected to a LAN (Local Area Network) via an HNC (High-speed Net Connect) channel adapter.

The HNC is a high-performance network connection for BS2000 systems on /390 servers, see the “HNC” manual [13 (Related publications)].

Network connection via LAN board (x86 servers)

x86 servers are connected to a LAN by means of an integrated controller (PCIe controller). Like all I/O operations, the I/O operations to the integrated controller are routed via the I/O processor.

Net-Storage

BS2000 enables access to Unix file systems via NFS. This enables BS2000 files to be saved and processed by file servers and Net-Storage in released directories.

Basic information on Net-Storage can be found in the “Introduction to System Administration” [10 (Related publications)].

Details of the configuration of Net-Storage are provided in the “Net-Storage Installation Guide”.

Information and notes on the performance of Net-Storage can be found in section “Net-Storage behavior”.

Virtual machines

VM2000 permits simultaneous execution of multiple, self-contained BS2000 guest systems on a real BS2000 server. The individual guest systems run on virtual machines (VMs) and, as far as their functions are concerned, operate in the same way as in native mode.

The resources CPU, main memory and devices are assigned to the virtual machines and thus to the guest systems by VM2000.

A hypervisor controls the execution of the guest systems on the VMs. In particular it virtualizes the global resources CPU and main memory and starts execution of the operational guest system’s CPUs on the real CPUs (scheduling).

- On /390 servers the VM2000 hypervisor is a separate load module of VM2000 which is loaded (automatically) when VM2000 operation is initialized.
- On x86 servers the Xen hypervisor performs this role. Some of the hypervisor tasks are performed by the carrier system X2000.

VM2000 on /390 servers distinguishes two processor statuses:

- Hypervisor mode (the VM2000 hypervisor runs in this status)
- VM mode (the guest systems run in this status)

A context (a set of registers containing the states of the CPU and information specific to VM) is associated with each of these statuses. This context is loaded when the status is activated.

In contrast to native mode VM2000 overhead arises:

- Hypervisor overhead (program execution in hypervisor mode)
- Indirect overhead (reduced server performance through changing VM contexts)

With careful configuration the VM2000 overhead is between 5% and 15% of the server performance. How high the VM2000 overhead on a server will actually be depends on the VM2000 configuration. Refer to the information in the section "Recommendations for an optimal configuration" to keep the VM2000 overhead to a minimum.

VM2000 scheduling (/390 servers)

When a VM is scheduled under VM2000 the VM2000 hypervisor starts a loadable virtual CPU of a VM on a free real CPU. VM2000 uses two different processes for scheduling at VM runtime:

- CPU assignment in time slices (see below)
- Fixed CPU assignment (dedicated CPUs, see "Fixed CPU assignment" below)

Here the CPU performance of the server is distributed ideally across the loadable virtual machines in accordance with the settings for the VMs.

> ¡ On x86 servers the Xen hypervisor performs scheduling. To do this it uses a time slicing procedure, but this is not influenced by VM2000. The VM-specific control factors of multiprocessor level, CPU quota and limiting of the CPU performance are also available here.

CPU assignment in time-slicing mode

Normally the number of attached real CPUs in a CPU pool is **less** than the sum of the attached virtual CPUs of all active VMs which are assigned to this CPU pool. VM2000 starts a virtual CPU on a real CPU from the pool using time slicing.

The maximum size of the time slice is defined dynamically for each virtual CPU of a VM in VM2000 in the range from 0.1 to 8.0 milliseconds. VMs with a “very small” CPU quota are then also assigned a smaller time slice. The maximum size of the time slice is only reached in few cases on the current BS2000 servers. Generally a time slice is terminated by the virtual CPU of the VM entering the interruptible idle status.

The VM2000 administrator can influence the distribution of the CPU performance using VM-specific control parameters, see "VM-specific control parameters" below. VM2000 also enhances the performance, e.g. by means of CPU affinity, see "CPU affinity" below.

The scheduling priority of the individual guest systems is derived from the specified CPU quotas and the CPU time used most recently.

VM-specific control parameters

The control parameters which are relevant to performance for a VM in time slicing are:

- Multiprocessor level of the VMs/guest systems
- CPU quota of the VM group, CPU quotas/member CPU quotas of the VMs (see "CPU share of a VM" below)
- Performance limit of the VM or VM group (MAX-CPU-UTILIZATION, see "Performance limit of a VM" below)
- VM privilege IO-PRIORITY (see "Peripherals")

The time period in which a guest system can accept the CPU quota assigned to it depends to a very large extent on the degree of utilization of the VM2000 time slice, both that of the home guest system and of the other guest systems. This degree of utilization is high when the workload is CPU-intensive and low when the workload is I/O-intensive.

If a guest system does not fully utilize the VM2000 time slice assigned to it, this improves the I/O behavior of the other guest systems.
In the case of a multiprocessor VM, VM2000 attempts to distribute the unused share of a virtual CPU’s CPU performance primarily to the other virtual CPUs of the VM.

However, if the time slice is utilized fully by a guest system, the I/O times can be considerably increased, above all for guest systems whose load consists almost exclusively of I/O-intensive tasks. The measures that can then be taken are:

- restriction of the CPU-intensive guest system by MAX-CPU-UTILIZATION (see "Performance limit of a VM" below) and/or
- assignment of the privilege IO-PRIORITY (see "Peripherals")

/SHOW-VM-STATUS INFORMATION=*SCHEDULE provides information on the mean value of a guest system's used time slice. Here the wait time up to activation of the virtual CPU is output as the measurement for the "increase of the VM".

CPU share of a VM

In the information commands VM2000 outputs various CPU quotas for a VM or VM group which enable the VM2000 administrator to observe the planned and current distribution of the CPU performance:

- CPU-Q (CPU quota, /SHOW-VM-ATTRIBUTES/-RESOURCES)
  is the (MEMBER-)CPU-QUOTA of the VM set with /CREATE-VM or /MODIFY-VM-ATTRIBUTES
- EFF-Q (effective CPU quota, /SHOW-VM-ATTRIBUTES/-RESOURCES INFORMATION=*CPU)
  is the standardized CPU quota of the VM taking into account the constraints multiprocessor level, performance limit of the VM and CPU pools
  (sum of the standardized CPU quotas of all **configured** VMs = 100%).
  EFF-Q shows the CPU share of the VM or VM group to be expected in the long term in the event of a CPU-intensive load in all guest systems and when all real and virtual CPUs are connected.
- CUR-Q (current CPU quota, /SHOW-VM-STATUS)
  is the standardized CPU quota of the VM taking into account the constraints multiprocessor level, performance limit of the VM and CPU pools
  (sum of the standardized CPU quotas of all **active** VMs = 100%).
  CUR-Q shows the CPU share of the VM or VM group to be expected at present in the event of a CPU-intensive load in all guest systems and in relation to the real and virtual CPUs currently connected.

Performance limit of a VM

If a VM or VM group is never to be assigned more than a particular percentage of the server's CPU performance (not if CPUs are free, either), this can be defined using the MAX-CPU-UTILISATION parameter. Restricted CPU performance can, for example, be required in the context of service levels. The parameter enables excess CPU utilization by a guest system with a CPU-intensive load to be restricted.

CPU affinity

In time slicing VM2000 attempts to ensure with scheduling that a virtual CPU starts on the **same** real CPU in the event of the next scheduling procedure. CPU caches and initialization data of the VM are retained and reduce the indirect overhead. However, the primary aim is to optimize the response time.

Fixed CPU assignment (dedicated CPUs)

To use dedicated CPUs, the number of attached real CPUs in a CPU pool must be at least as high as the sum of the attached virtual CPUs of all active VMs which are assigned to this CPU pool. When this is the case, VM2000 automatically assigns precisely one real CPU to each virtual CPU of a VM in this CPU pool.

In this case the VM2000 administrator can use the VM attribute VM-ACTIVE-IDLE to determine whether a VM still retains control over a real CPU if the VM’s virtual CPU which is running on it is inactive (interruptible wait state, "idle"), see "CPU pools".

VM groups on /390 servers

Multiple VMs can be combined in **VM groups**.

CPU utilization is then controlled by VM2000 in two stages. This strategy enables logically associated guest systems to be examined as a unit with regard to CPU utilization. For example, multiple VMs can be provided for a customer and be assigned a service level.

The following VM group parameters enable the VM2000 administrator to influence the distribution of the CPU performance:

- A VM group is prioritized vis-à-vis other VM groups or (individual) VMs in the same way as (individual) VMs, i.e. via the CPU-QUOTA and MAX-CPU-UTILIZATION parameters of the VM group.
- The MEMBER-CPU-QUOTA controls prioritization among the VMs which form the VM group. For the scheduling priority the MEMBER-CPU-QUOTA is qualified by the CPU-QUOTA of the VM group.
- In addition to the CPU utilization of the VM group the CPU utilization of each VM in a VM group can be limited using MAX-CPU-UTILIZATION.
- With VM groups VM2000 primarily attempts to distribute the unused share of the CPU performance of a VM to the other members of the VM group.

VM groups do not generate any additional VM2000 overhead.

CPU pools

Under VM2000, the real CPUs of a server which are available for VM2000 operation can be split into different, disjunctive **CPU pools**. This strategy enables greater hardware efficiency and CPU affinity to be achieved for servers with a large number of CPUs. A VM is assigned to precisely one CPU pool.

VM2000 attempts to assign the CPU performance of the real CPUs in a CPU pool to the individual guest systems assigned to a CPU pool in proportion to the CPU quotas. An upper limit results from the maximum CPU utilization of a VM.

The following characteristics of CPU pools have an influence on the server's performance:

- CPU pools reduce the VM2000 overhead especially with a larger number of real CPUs
- CPU pools permit a targeted but strictly limited performance (service level) for a set of VMs ("restricted VM group") with a further reduction in VM2000 overhead
- CPU pools on /390 servers enable a permanent CPU assignment (dedicated CPUs) to be created for critical productive systems.

Criteria for establishing CPU pools:

- The size of the CPU pools (the number of real CPUs assigned to a CPU pool) should be planned to ensure that the CPU performance made available corresponds to the sum of the planned share of performance for the VMs assigned to the CPU pool. As with the CPU utilization of the entire server, the CPU utilization of the individual CPU pools in OLTP mode should not be greater than 75%.
- The utilization of the server's real CPUs should be as even as possible. Consequently the utilization of the CPU pools should be as even as possible.
- In the case of CPU pools with just one real CPU it is to be expected that the dilation of the hardware service time (see "Peripherals") is greater than with pools containing multiple real CPUs.

If the performance requirement of the guest systems changes in the course of a session, /SWITCH-VM-CPU can be used to add/remove CPUs to/from the CPU pool dynamically.

Criteria for assigning VMs to the CPU pools:

- In the case of guest systems with applications whose response time behavior is heavily dependent on the duration of the I/O operations, the contending VMs or guest systems in the same pool should be selected carefully so that the increase in the hardware service time does not have an unfavorable effect. For example, you should prevent a large number of CPU-intensive guest systems from being assigned to one CPU pool.
- In the case of guest systems with particularly high performance requirements the CPU pool can be equipped with enough real CPUs to ensure that a real CPU is available for each active virtual CPU. With respect to scheduling, VM2000 then works with a permanent CPU assignment on /390 servers.
- The VM-ACTIVE-IDLE=*AT-DEDICATED-CPUS parameter (/390 server) enables a VM with fixed CPU assignment to still retain control over a real CPU if the VM's virtual CPU which is running on it is inactive (interruptible wait state, "idle").
  This also prevents the changes of context when the guest system is in the wait state (idle). On /390 servers this allows a performance to be achieved which is almost equal to that in native mode.

> **i** No evaluation of the VM-ACTIVE times (/SHOW-VM-STATUS or VM report of openSM2) is possible for VMs with VM-ACTIVE-IDLE.

When required, individual VMs or entire VM groups can be assigned dynamically to other CPU pools using /ASSIGN-VM(-GROUP)-TO-CPU-POOL.

Main memory

Each guest system has its own exclusive share of the server’s main memory. Access to the main memory is carried out as efficiently as in native mode. The access times are the same.

The main memory distribution should be carefully planned. The main memory size required for a VM is initially estimated or derived from the existing system environment. After the paging rate has been measured using SM2 it may be necessary to make corrections. The paging rate should be low, in accordance with the load profile (TP- or dialog-oriented). The current main memory size of a VM (and on /390 servers also the location) can be determined using /SHOW-VM-RESOURCES INFORMATION=*MEMORY.

Because of the code expansion and the (resident) memory requirements for JIT buffers, the main memory requirement for a VM on an x86 server is greater than for one on a /390 server. On an x86 server, 40% of the main memory is by default reserved for JIT buffers.

The main memory size for the VM2000 hypervisor (/390 server) is calculated by VM2000 in accordance with the size of the peripherals.

The minimum main memory size (MIN-MEMORY-SIZE) for a VM should only be defined when the functions for main memory configuration (/EXPAND- /REDUCE-VM-MEMORY) are to be used. The size selected must then be enough to satisfy at least the resident memory requirements in the guest system.

The resident memory requirement depends on whether the software product DAB is used.

The JIT memory of the x86 servers (approx. 40% of the main memory) also resides in the resident memory. The MIN-MEMORY-SIZE of a VM may nevertheless be defined with a lower value because a corresponding amount of the resident JIT memory is released dynamically with /REDUCE-VM-MEMORY in order to comply with the 40% rule for JIT. The process is reversed with /EXTEND-VM-MEMORY.

On x86 servers the maximum main memory size (MAX-MEMORY-SIZE) determines the upper limit to which the VM's main memory can be extended (/EXTEND-VM-MEMORY). The default value is double the VM's MEMORY-SIZE. If the main memory of a VM is not to be extended, the value MEMORY-SIZE should be selected for the maximum main memory size.

The current utilization of resident memory can be determined by SM2 from the values of the MEMORY report group: Resident Memory = TOTAL - Pageable Frames.

Peripherals

Channels

On /390 servers the channels are available to all VMs. They are shared via the firmware. The utilization of the channels can be observed with SM2 over multiple VMs.

On x86 servers, no channels in the original sense are available. X2000 emulates devices with a virtual I/O path which is available on all VMs.

Devices

The devices which exist are determined dynamically when the monitor system and guest systems are started up. VM2000, the monitor system and all guest systems therefore know and monitor the same devices.

Peripheral devices must be assigned to the guest systems individually. There are two options:

- **explicitly** by the VM2000 administrator with /ADD-VM-DEVICES or /SWITCH-VM-DEVICES or
- **implicitly** by a guest system with the privilege ASSIGN-BY-GUEST=*YES(...) with /ATTACH-DEVICE. For this purpose the VM2000 administrator must permit this type of assignment for the VMs and devices concerned.

> ℹ When disks are assigned implicitly, “Shared” is entered by default as the usage mode.

Peripheral devices can be assigned exclusively to a single guest system or be available to a number of guest systems as shared devices:

- **Exclusive assignment** = The device is then permanently assigned to one guest system

  I/O operations are then initiated directly as in native mode.

  For guest systems in the wait state ("idle"), the VM2000 hypervisor (/390 server) monitors the receipt of I/O termination interrupts and initializes the corresponding guest system in accordance with its scheduling priority. This hypervisor overhead is minimal.
- **Shared assignment**

  = The device is assigned to several different guest systems

  The type of I/O processing on /390 servers depends on the following condition:

  1. If the device is only assigned to one guest system, the I/O operations are executed directly (direct I/O) as in the case of exclusive assignment without the involvement of the VM2000 hypervisor.
     In the VM2000 information commands the device has the letters SH(D) against it.
  2. If the device is assigned to a number of guest systems, on /390 servers the VM2000 hypervisor interrupts all I/O operations of the guest system and serializes them (indirect I/O).
     In the VM2000 information commands the device has the letters SH(I) against it.

     In the case of shared devices the CPU requirements of the VM2000 hypervisor increase with the I/O rate. The VM2000 overhead increases by roughly 0.4% for each 100 I/O operations per second.

     On x86 servers disk I/O operations are executed as logical requests via the RSC interface. There is no hypervisor overhead.

Service times for devices (monitoring program SERVICETIME of SM2) can be ascertained by only one VM at any given time.

Privilege IO-PRIORITY (/390 servers)

Applications generally perform synchronous I/O operations. After it has started the task waits for the end of the I/O operation. If no further task is ready to execute, the wait state ("idle") is switched to and the real CPU is redistributed by the hypervisor. After the I/O operation has been completed the guest system must wait for a real CPU to be assigned again before it can process the end of the input/output. The delay between the actual end of the I/O operation and it being processed in the guest system is referred to as the **increase in the hardware service time** or **IO increase**.

CPU-intensive guest systems can as a result disturb the operation of I/O-intensive guest systems. Indications of this are provided by /SHOW-VM-STATUS INF=*SCHEDULE in the %RUNOUT and %WAIT output columns.

The quicker the CPU is reassigned to the guest systems, the lower the IO increase is. On /390 servers, the IO-PRIORITY=*YES privilege can be assigned to a VM for this purpose. If a virtual CPU of such a VM is in the wait state, it is activated immediately on a real CPU when an I/O termination interrupt arrives from the hypervisor in order to process the result of the input/output. The runtimes of I/O-critical jobs are considerably improved.

The positive effect is that the hardware service times obtained are as good as in native operation, but on the other hand the scheduling rate is higher. As a result, the VM2000 overhead also increases.

It is therefore advisable to set up the guest systems in the first place without the privilege (i.e. with IO-PRIORITY=*NO) and only to assign the privilege IO-PRIORITY=*YES where necessary.

The IO-PRIORITY=*YES privilege provides no benefits in the case of dedicated CPUs.

Support of the IORM utility routine (I/O Resource Manager)

The IORM utility routine (see the “Utility Routines” manual [6 (Related publications)]) is supported by VM2000. In VM2000 mode IORM should be running on the monitor system and on all guest systems.

The following IORM functions are implemented in conjunction with VM2000:

- Monitoring for alias devices and limiting the number of aliases for FastDPAV with DPAV
- Optimized device selection in ETERNUS CS operation under VM2000 with DDAL (Dynamic Device Allocation)

  As an alternative to using the devices in accordance with the way they are generated (as in the previous versions), the operating system can ensure even utilization of all the available ICPs (Integrated Channel Processors).

  Under VM2000 the IORM function DDAL extends optimized (local) device selection to cover all of a server’s BS2000 guest systems.
- I/O limit for VM2000 guest systems with IOLVM (I/O Limit for Virtual Machines, /390 servers)

  In VM2000 mode, less important guest systems which are I/O-intensive can hinder other, more important guest systems in I/O operations.

  The IOLVM function of the IORM utility routine can detect such conflict situations and specifically slows down I/O operations of the user’s own guest system if one of the I/O resources that are used jointly (channel, port, path, disk) exceeds the specific I/O limit.

  The I/O limit for IOLVM is defined as the maximum I/O utilization of the VM in the MAX-IO-UTILIZATION operand in the VM2000 command /CREATE-VM or /MODIFY-VM-ATTRIBUTES.

System-internal processes which are relevant to performance

This chapter describes the following system-internal processes which are relevant to performance:

- CPU states
- User tasks and system tasks
- Task management
- Job management
- Data management

CPU states

There are four CPU states in BS2000. Of these, the TU, TPR and SIH states are essential to normal operation. The MEH state is used solely to handle hardware errors.

The following definitions apply:

TU: Task Unprivileged

TPR: Task PRivileged

SIH: System Interrupt Handling

MEH: Machine Error Handling

The user task (application program) runs in CPU state TU.

System routines which may be assigned directly to the user task (e.g. SVC: RDATA) are processed in CPU state TPR (see figure 5).

CPU time taken up in the TU and TPR CPU states should be regarded as productive time from the user's standpoint.

The basic control mechanisms of BS2000 are run in CPU state SIH. The SIH CPU state cannot be interrupted except after MEH.

System tasks provide support for organizational tasks; they run in CPU state TPR (see figure 6).

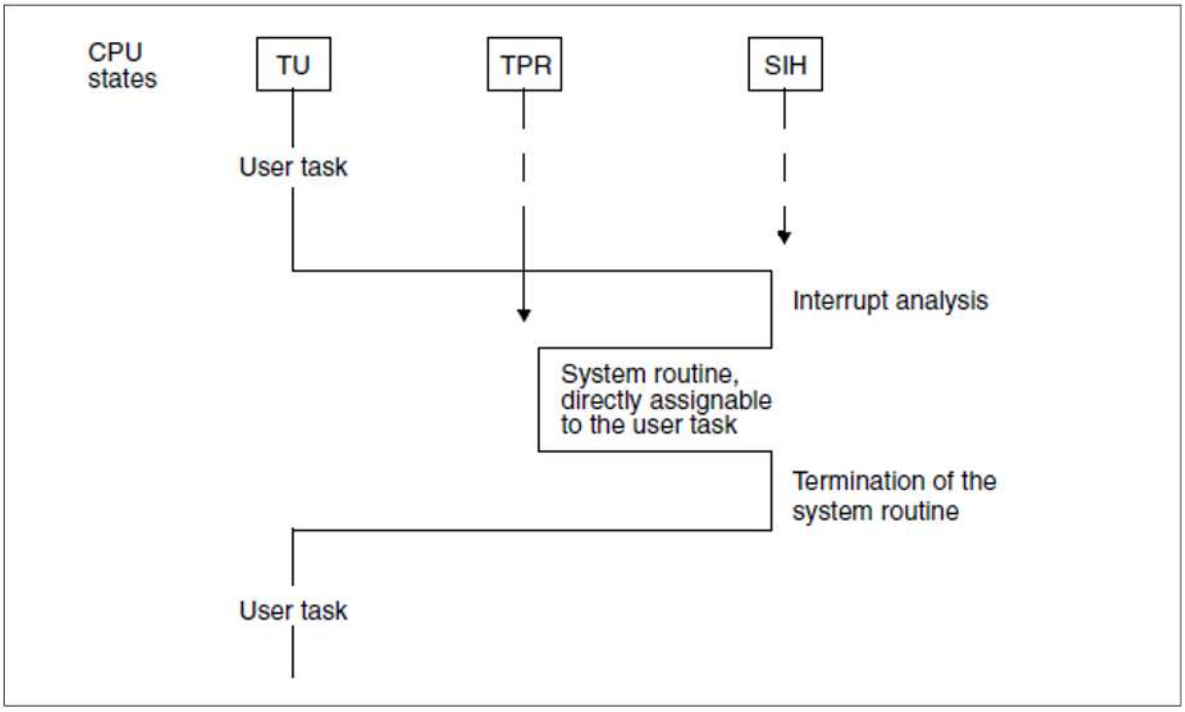


Figure 5: BS2000 CPU states (user task)

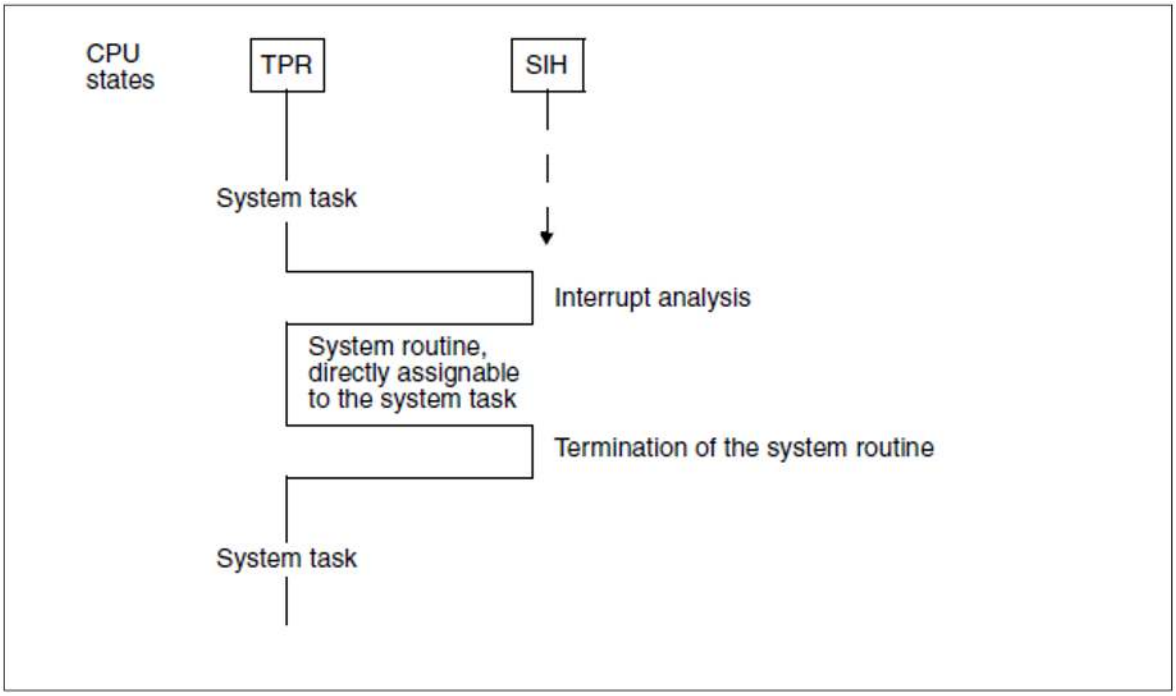


Figure 6: BS2000 CPU states (system task)

An interrupt is caused either by a user/system task which requires the support of the operating system or by an event which must be handled by the operating system.

There are five types of interrupt:

- SVC interrupt (= task-related interrupt)

  A user or system task requests certain system services. The calling task is charged not only for the time for the system routine, but also for the interrupt analysis and the termination of the system routine.

- Program interrupt (=task-related interrupt)

  This includes “genuine” errors (address error, data error, etc.) which cause the program to abort. These errors may be ignored when considering throughput. Also included is the interrupt cause "page not in memory" (event code **4C**).

  In the case of event code **4C** it is impossible to continue the program run for the time being since the addressed page is not immediately available. There are two possibilities:

  - The page is still in main memory, namely in the “free pool” (read-only queue or read/write queue):

    The page, which has not been addressed for some time, has been allocated to the “free pool” by paging management. The program can be resumed following a simple page chaining procedure (PAGE RECLAIM). As a result of the global working set strategies, page reclaiming is a very rare occurrence (see section "Activating/deactivating tasks"). In most cases, it is the result of the first access to a page (First Page Access).

  - The page is not in main memory and must be loaded there by means of a paging I/O operation (PAGE READ).

- Timer/external interrupt (= asynchronous interrupt)

  A signal from the interval timer or ETC (Elapsed Time Clock).

- I/O interrupt (= asynchronous interrupt)

  A signal that an event has occurred in connection with input/output. (In most cases, this involves the occurrence of an end condition when performing the I/O operation.)
- Hardware error interrupt (= asynchronous interrupt)

  A signal that a hardware error has occurred.

Once the interrupt has been treated (with the exception of the SVC interrupt), the dispatcher (selection) routine of the task initiator will select and pass control to the executable user or system task with the highest priority (task in control; see also section "Initiation/deinitiation of tasks").

User tasks and system tasks

BS2000 knows user tasks and system tasks.

User tasks

User tasks are created with the aid of system tasks at the request of the user. They are subdivided according to the type of task into

- Dialog tasks
  (task type X'40', generated in the case of /SET-LOGON-PARAMETERS by the system task DIAA)
- Batch tasks
  (task type X'20', generated in the case of /ENTER-JOB by the system task TSC)

The task attribute can be distinguished from the task type. The former influences capacity allocation in the current session. The following task attributes are available for the different applications:

- TP for transaction applications
- DIALOG for dialog (interactive) applications
- BATCH for batch applications

Dialog and batch tasks automatically receive the task attribute DIALOG or BATCH.

The tasks of transaction applications (DCAM, UTM, UDS, SESAM/SQL, etc.) are run under the same task type used when they were started (as a batch or interactive task).

If a job class is defined using the parameters TP-ALLOWED=*YES(CATEGORY=xy),START-ATTRIBUTE=*TP, these tasks are then given the task attribute TP as an additional identifier (see section "Job classes").

The TP task attribute can also be applied with the TINF macro while simultaneously switching to the category with the default name TP.

System tasks

System tasks (task type X'80') are automatically assigned the task attribute SYSTEM.

There exist preallocated system tasks and important, dynamically generated system tasks (see the "Introduction to System Administration" [10 (Related publications)]).

Task management

In BS2000, logically interrelated requests to the system and their execution are managed in the form of tasks.

If these requests to the system are UTM applications, the various transaction codes within UTM can be controlled by means of so-called TAC classes. Similar resources are available here as are used for tasks and categories under PRIOR or PCS.
An introductory summary on this subject is provided on "Optimizing the various phases".
Detailed information can be found in the openUTM manuals.

Basic principles of task management

Both user tasks and system tasks are monitored and controlled by the task management routines.

For a task to be able to run, task management must make the following decisions:

- allocation of authorization to use main memory --> activation
- allocation of the resource CPU --> initiation

Initiation is possible for already activated tasks only.

Criteria for managing **main memory**:

- multiprogramming level (MPL) per category
- Priority
- resource workload (CPU, main memory, paging activity)
- system services rendered (CPU time, number of I/O operations).

Criterion for managing the resource **CPU** is only the priority.

> i If multiprocessors are in use and the TANGRAM function has been activated, the assignment algorithm can result in a different initiation sequence (see also section“TANGRAM concept”).

Internal category name

Both task categories and priorities are given a system-internal representation which differs from that at the user interface.

The default categories have fixed identifiers:

Category	Category identifier
SYS	0
DIALOG	1
BATCH	2
TP	3
xy	4
. . .	. . .

Internal priority

The specified variable priorities are weighted according to the importance of the category concerned (represented by the WEIGHT-CODE parameter). The variable internal priority is a function of:

- the external priority
- the WEIGHT-CODE (W) of the category to which the task belongs
- the sum S of the weights of all categories

$S = W_{SYS} + W_{TP} + W_{DIAL} + W_{BATCH} + W_{xy} + ...$

Variable internal priority = 1 + (256 - external priority) * 0,5 * (1 + W / S)

Fixed external priorities assigned by the user are converted as follows:

Fixed internal priority = 256 - external priority

The internal priority serves as a basis for calculating the activation and initiation priorities.

Introduction to task management

The following aspects of task management are explained in this section:

- Activating/deactivating tasks
- Initiation/deinitiation of tasks
- Controlling tasks via queues

Activating/deactivating tasks

Activating and deactivating tasks comprises the following specific functions:

- Activation
  Allocation of the authorization to use main memory to an executable (inactive, ready) task.
- Deactivation
  Withdrawal of the right to use main memory when expecting long wait periods or after providing certain system services.
- Forced deactivation
  Automatic deactivation of a task due to a resource overload.
- Preemption
  Activation of one task at the cost of deactivating another task.
- Control functions within the framework of main memory management
  - Activation Control Function (ACF)
  - Preemption Control Function (PCF)
  - Paging Load Control Function (PLC)
  - System Service Slice Runout Control
  - Waiting Time Runout Control

All the functions mentioned above are only of significance in the rare case of a main memory bottleneck.

Activation

Activation is possible only when permitted by the Activation Control Function (ACF) on the basis of the CPU and main memory workload measurement and the paging activity (see "Control functions within the framework of main memory management").

The primary goal is **to attain the MIN MPL value** (MINIMUM-ACTIVE-TASKS) specified by the user **in each category.**

The task to be activated is defined by this process:

1. Filtering out all task categories in which no tasks are ready for processing
2. Assigning priorities to the remaining categories:
   - Priority 1: categories that have not yet reached MIN MPL
   - Priority 2: categories that have reached or exceeded MIN MPL, but not yet reached MAX MPL
   - Priority 2: categories that have reached or exceeded MAX MPL
3. Only the highest priority categories are considered. That means tasks in categories with priority 2 can only be activated, if no priority 1 categories exist.
4. If more than one category is left, the one with the lowest category index (see below) is selected.
5. Of this category, the task with the highest activation priority (see below) is selected.

> *Exception* If the MIN MPL value has already been reached for the calculated category and there is an activation request for a fixed-priority task from another category, the latter task will be activated, provided this is permitted by the ACF control function; if not, preemption will take place.

Tasks with system priorities are always activated immediately. Here, too, the ACF control function decides whether this is possible within an activation operation or whether preemption should take place.

Category index

Calculation of the category index:
Index = (NAT + 1 - N) / WEIGHT

where:

NAT = Number Active Tasks
WEIGHT = Parameter WEIGHT-CODE
N = 0, if MPL < MIN MPL
N = MIN MPL, if MIN MPL <= MPL < MAX MPL
N = MAX MPL, if MAX MPL <= MPL

Activation priority

For variable external priorities, activation priority is dependent on:

- the variable internal priority
- the CPU state (TU or TPR)
- the point in time t_A when the most recent activation occurred (in each case the difference D is determined) and
- a configuration factor C

Variable activation priority = (Variable internal priority + P) * Dt_A(Elapsed time) / (Dt_A(CPU time) * C)

Tasks waiting for authorization to use main memory and in CPU state TPR receive a higher activation priority; this is expressed by the constant P.

P = 5 If in TPR

P = 0 If in TU

The configuration factor C depends on the server type used and it is increased dynamically as the paging rate increases.

Fixed activation priority = Fixed internal priority

Deactivation

Deactivation of active tasks makes main memory available for inactive ready tasks.

Criteria for deactivating active tasks:

- further processing is impossible,
  e.g. due to wait calls in the program (PASS, VPASS) or waiting for terminal input in interactive mode (WRTRD, RDATA)

- wait periods were exceeded while waiting for events,
  e.g. waiting for terminal input in TP mode

  DCAM: YRECEIVE

  UTM: KDCWAIT

  Task communication:

  - ITC: REVNT
  - TU eventing: SOLSIG
  - Forward eventing: RSOFEI

  (See “Control functions within the framework of main memory management”: "WaitingTime Runout Control".)
- certain system services in the form of CPU time and the number of I/O operations have been rendered (see “Control functions within the framework of main memory management”: "System Service Slice Runout Control").

Deactivation is suppressed when the function “No Deactivation” (TINF macro: DEACT parameter) is switched on. Exception: the wait call VPASS n always causes deactivation.

Forced deactivation

Forced deactivation of tasks occurs when the Activation Control Function (ACF) detects an excessive workload on the resources CPU and main memory or an excessively high paging rate in relation to the CPU type (matrix decision).

Within an ACF measurement interval, only one task is deactivated at a time.

The following category attributes are used to determine the category from which a task is to be forcibly deactivated:

- MIN MPL
- MAX MPL
- WEIGHT

> The user should ensure that only tasks from “less important” categories are deactivated by specifying correspondingly high MIN MPL and WEIGHT values for categories with important applications.

For the most part, forced deactivation applies to low-priority tasks which are in the process of building up their working set (see below). As a result, non-productive system overhead is kept to a minimum.

Forced deactivation will not occur for:

- tasks with a fixed priority
- tasks in CPU state TPR
- tasks which have switched on the “No deactivation” function.

> A working set consists of virtual pages in main memory which can currently be accessed but in principle can be swapped out to the paging memory.

Preemption

Preemption can take place whenever an activation request exists but the ACF control function forbids further activation in light of the CPU and main memory workload and the size of the paging rate.

Preemptions are set out in the report group PRIOR-ACF of openSM2. They should be avoided at all costs. They may be caused when:

- The MIN MPL values are set too high.
- There are two many tasks with a fixed priority.

There are two types of preemption:

1. preemption between **different** categories

   Active task A is preempted by inactive task B belonging to another category. This occurs when:

   - for the category with inactive task B has MPL < MIN MPL
   - at the same time that the category with active task A has MPL >= MIN MPL

   Preemption of tasks from different categories:

   a. Activation of a task from a category with MPL < MIN MPL and the lowest category index number. The task with the highest priority from this category will be activated.

   b. Forced deactivation of a task from a category with MPL >= MIN MPL
   (see "Forced deactivation")

2. preemption within the **same** category

   Active task A and inactive task B belong to the same category. This is only possible when:

   - inactive task B has a fixed priority
   - active task A has a variable priority.

   Preemption of tasks within the same category:

   a. Forced deactivation of a variable-priority task from this category
   (see "Forced deactivation")

   b. Activation of a fixed-priority task.

Control functions within the framework of main memory management

- Activation Control Function (ACF)

  ACF measures the CPU and main memory workloads and the paging activity at periodic intervals (--> INVOCATION LONG = INVOCL, see report group PRIOR-ACF of openSM2).
  The polling interval lies between 0.25 and 2 seconds, depending on the CPU workload.

  CPU-UTILIZATION

  Value calculated on the basis of the CPU workload measured during the interval and the time spent waiting for CPU allocation.

  MEMORY-UTILIZATION

  Ratio of working set requirement estimated by the system for all active tasks (sum of PPC values) to the number of pages available for paging (NPP = Number of Pageable Pages).

  Calculation of the PPC value (planned page count) is therefore approximate: the UPG value (used pages) is compared with an assumed mean working set value, dependent on the category type.

  Since as many pages as possible are kept active, the total UPG value is practically always in the magnitude of the available paging main memory and has only limited relevance. The working set requirements are expressed exclusively by the PPC value. This value represents only a rough estimate. With todays large memory configurations advantages result from the fact that no outlay is necessary for estimating exact memory requirements, as they are superfluous.

  PAGING-UTILIZATION

  Paging rate.

  The measured values are divided into the following classes:

  - H (High)
  - M (Medium)
  - L (Low)

  The reference values for utilization are given in the report group PRIOR-ACF of openSM2. The limit values for these classes are defined at system startup on the basis of the server used (CPU speed, main memory configuration).

  Depending on the resource workload and the specified category attributes or task priorities, the system decides on the basis of a matrix whether further activation is permitted or whether forced deactivation, preemption, or even no action at all should take place.

  Before a task is actually activated, ACF makes an additional measurement regarding solely the main memory workload in order to ensure that the workload on main memory will permit further activation (INVOCATION SHORT = INVOCS, see report group PRIOR-ACF of openSM2).

- Preemption Control Function (PCF)

  Task preemption at the activation level is time-consuming since the working set of the preempted task must be reconstructed during reactivation. In order to keep the resultant overhead within reasonable limits, the system tries to keep the preemption rate as low as possible.

  PCF periodically monitors the flattened preemption rate per interval. Depending on the hardware used, the polling interval lies between 20 and 48 seconds.
  If at least one preemption occurs during this interval, the MAX MPL OPTION will be switched on. This means that from this point on the MAX MPL value must be regarded as an inviolable boundary beyond which activation will not take place. (The MAX MPL OPTION is switched off when no preemption occurs over several intervals).

  PCF takes countermeasures to keep the preemption rate low. The strategy behind these measures is to carry out preemption requests, wherever possible, as additional activations rather than as preemptions. To do this, PCF reserves main memory areas for preemption requests whenever a certain preemption rate per interval (preemption limit) is exceeded.

  The memory area reserved for preemption requests is enlarged step by step as the preemption rate increases (5% to 15% of the real main memory configuration depending on the size of main memory). This simulates a heavier main memory workload, with the result that tasks with no preemption request will not be activated, leaving sufficient main memory to activate tasks **with** a preemption request.

  The system distinguishes between 4 different levels (preemption levels 0 to 3), which are displayed with the message EXC0455 TASK PREEMPTION LEVEL=(&00):

  Level 0

  Normal state (preemption rate < preemption limit)

  Level 1, 2

  Short-term overloads
  The preemption level continues to climb if

  - preemption rate > preemption limit
  - preemption rate in interval n + 1 > preemption rate in interval n.

  Level 3

  Long-term overload. (This can arise when the MIN MPL parameters are set too high: tasks from categories which have not yet reached MIN MPL automatically issue a preemption request.)
  The operator is told that preemption level 3 has been reached and as of this point receives notification of every change made to the preemption level. If the system discovers by means of a lowered degree of multiprogramming that the MIN MPL parameter setting is too high, the operator will be asked to check this setting and to lower the MIN MPL value of suitable categories.

  As the preemption rate decreases, reserved main memory will be released step by step. Once the normal state has been reached, the operator receives the message:

  ```
  EXC0456   PREEMPTION CONTROL FUNCTION TERMINATED
  ```

  If preemption messages are output, steps should be taken to remedy the situation (see "Preemption").

- Paging Load Control Function (PLC)

  PLC carries out long-term measurements of paging activities (polling interval 20 to 48 seconds, depending on the speed of the CPU).It can be seen as a supplement to ACF. While ACF measures the current paging rate and compares it to the predefined limit values in order to make short-term decisions, PLC affects the limit values defined at system startup, depending on the workload combination.

  If the workload is dialog-oriented, higher paging activities are allowed as compared to workloads oriented towards TP or TP/batch tasks. This minimizes the negative effect of paging activities on TP-oriented workloads and optimizes throughput in the case of batch-oriented workloads.

  PLC keeps statistics regarding the frequency with which the states

  - underload
  - medium load
  - overload

  occur per polling interval with respect to paging activities. If the “overload” state occurs once per polling interval, the MAX MPL OPTION will be switched on. The MAX MPL OPTION will be switched off if the “paging overload” state fails to occur in the course of several intervals, the predominant preemption level is 0 and the preemption rate is less than 1.

- System Service Slice Runout Control

  The “system service slice” is used to add up the system services received by an active task in the form of CPU time (in CPU states TU and TPR) and of I/O operations performed.

  System service is calculated using the formula SERVICE = (A * B * C) + D, where:

  - A: Mean instruction execution time (depending on CPU speed).
  - B: Mean number of instructions per I/O request.
  - C: Number of I/O operations since last activation.
  - D: CPU time since last activation.

  The size of the “system service slice” depends on the category and the server.

  The polling interval for “system service slice runout control” is 2 seconds.

  For purposes of recovery action, a distinction is made between two types of system service slice runout, depending on whether the task is in the TU or TPR CPU state:

  - system service slice runout – CPU state TU

    Deactivation will occur only when an inactive ready task with higher priority is available in the **same** category.

  - system service slice runout – CPU state TPR

    Deactivation will take place when “system service slice runout” occurs for the 2. time if there is an inactive ready task with a higher priority in the same category. If the task switches to state TU after the first runout, it is immediately handled as a runout in TU.

- Waiting Time Runout Control

  The “waiting time limit” function is used to monitor tasks which are waiting for an event (bourse or inter-task communication events) and thereby occupy main memory.

  The “waiting time runout control” function is only called when more than 25% of pageable main memory is occupied by tasks waiting for bourse events (e.g. DCAM: YRECEIVE, UTM: KDCWAIT, TU eventing: SOLSIG, forward eventing: RSOFEI) or ITC events (inter-task communication: REVNT). The polling interval lasts from 1 to 7 seconds, depending on the main memory workload.

  If the “waiting time limit” is exceeded for one or more tasks, deactivation will take place in the following manner:

  Variable-priority tasks belonging to a category whose MPL value is greater than or equal to MIN MPL are deactivated first (provided deactivation is permitted in connection with “waiting time runout”: DWTR parameter in the TINF macro).
  The process is terminated as soon as the amount of pageable main memory bound by the waiting tasks falls beneath the 25% limit.

  If this action is not sufficient to fall beneath the 25% limit, the deactivation process will be extended to include waiting tasks with fixed priority as well as categories whose MPL value is less than MIN MPL (provided deactivation is allowed). Regardless of the 25% limit, all tasks which satisfy the specified conditions and have been waiting for longer than 30 seconds will be deactivated.

Initiation/deinitiation of tasks

The initiation and deinitiation of tasks comprises the following specific functions:

- Initiation

  Allocation of the resource CPU to an active, executable task (which is in the "active, ready" state).
- Deinitiation

  Withdrawal of the resource CPU when delays are expected (transition to the “active, not ready” state).
- Preemption

  Deinitiation of a task followed by initiation of another task with a higher priority.
- Micro Time Slice

  Micro Time Slice (MTS)

Initiation

From the available active ready tasks, the dispatcher routine of the task initiator selects and passes control to the task with the highest initiation priority (--> task in control).

With multiprocessors there are CPU-local queues (see also section "Controlling tasks viaqueues").These queues are populated cyclically when tasks are activated. For each CPU, the task with the highest initiation priority is selected from the queue assigned to that CPU. Higher priorities which may be assigned to tasks in queues which belong to other CPUs are of no significance. The strategy “foreign tasks take precedence over idle states” is also adopted, i.e. a task from a “foreign” queue is initiated before the CPU becomes idle as a result of underloading. In this case, those queues containing the most tasks (“longest Q1”) are accessed.

If the TANGRAM function is used, the local CPU queues are populated according to the TANGRAM assignment algorithms rather than cyclically (see section "TANGRAM concept"). In these circumstances, the “idle in preference to foreign” strategy is adopted.

Initiation priority

Calculation of the initiation priority is analogous to that of the activation priority. The following additional points are taken into account:

- the point in time t_A of the most recent system service slice runout.
  The difference: current point in time minus t_A is formed.

- the “task in control” is given preference when calculating priorities.
  The formula P = P + 5 is used in order to keep the preemption rate as low as possible.

Deinitiation

In the interest of efficient CPU utilization, the system will withdraw control from a task (even if this task has top priority) whenever there are immediately foreseeable wait periods:

- For short wait periods there is a transition to the “active, not ready” state. This applies to wait periods such as: performing an I/O operation (DMS I/O, paging I/O), waiting for bourse or ITC events, VPASS 0 wait call in program.
- For long wait periods deactivation will also occur.
  This applies to wait periods such as: terminal input in interactive mode, PASS wait call, VPASS wait call in program.

Preemption

At the initiation level, preemption is entirely dependent on the initiation priority.

In order to preempt the “task in control”, the priority of the task to be newly initialized must be:

- higher by at least the value 5
- higher by the value 10 if the task in control is in CPU state TPR.

After an SVC call which has not led to deinitiation of the task, no preemption takes place. Only when another task moves to the head of the CPU queue while this SVC is being processed, and when the above-mentioned conditions are fulfilled, does the task calling the SVC lose the CPU.

Micro Time Slice

The “micro time slice” (MTS) is the maximum amount of CPU time which a task can occupy without interruption.

The CPU time used in the TU and TPR CPU states is added together to determine the CPU intensity of the individual task. In the case of a “micro time slice runout”, the priority is checked or redefined. When multiprocessors are used, following a “micro time slice runout” the task with the highest priority is selected from all the queues (Q1).

The size of the MTS varies between two outer limits, and is dependent on CPU intensity. After each I/O operation involving a wait period, the MTS is redefined:

MTS = Current CPU time - CPU time of the penultimate I/O

The limits are dependent on the CPU speed; the faster the CPU, the lower the limit values.

The more I/O-intensive the characteristics of a task, the smaller its associated MTS (which, however, must not fall beneath a certain minimum value). Since I/O-intensive tasks have low CPU intensity, no “micro time slice runout“ occurs.

If the task characteristics change in the direction of greater CPU intensity, a “micro time slice runout“ occurs relatively quickly, followed by a calculation of priority. This ensures that the system can respond quickly to changes in workload characteristics from I/O-intensive to CPU-intensive (as it does not have to wait for the periodic priority check). This type of workload change is particularly critical, since the domination of CPU-intensive tasks has a corresponding adverse effect on the degree of simultaneous processing, thereby reducing throughput.

Controlling tasks via queues

Tasks existing within an operating system can be in any of five different states:

- TASK IN CONTROL
- ACTIVE READY
- ACTIVE NOT READY
- INACTIVE READY
- INACTIVE NOT READY

The number of tasks in the state TASK IN CONTROL can only be as great as the number of CPUs the server has.

The remaining 4 states are wait states. They can be divided into 2 groups, depending on whether they have already been allocated the right to use main memory (ACTIVE) or not (INACTIVE).

Tasks are assigned to the individual wait states with the aid of queues; for reasons of clarity the “NOT READY” states are further subdivided into events to show why the task is not executable.

In order to improve the MP factor for multiprocessors, the queues for the active space of each CPU are provided separately (locally for each CPU) and the remaining queues are common to all CPUs (see figure 7).

The PEND/UNPEND routines are used to change the state of a task (characterized by moving from one queue to another):

- UNPEND routines move the task from its present waiting position towards use of the CPU.

- PEND routines put the task in a wait state due to an intervening event.

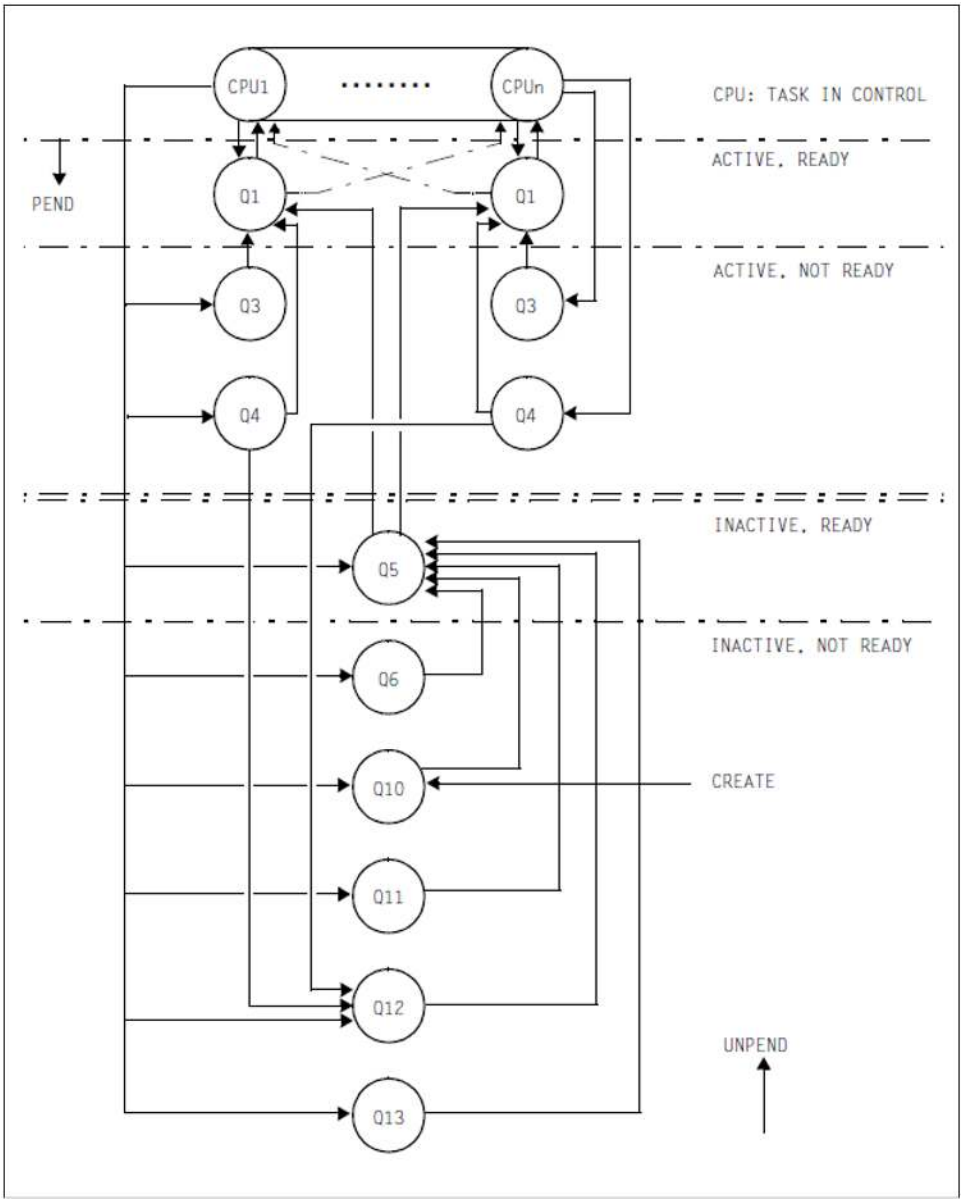


Figure 7: Queue transfers

Overview of the individual queues

Queues Q0, Q1, Q2, Q3 and Q4 are provided once for each CPU.

- TASK IN CONTROL

  Q0 Task occupies CPU

- ACTIVE, READY

  Q1 CPU queue
  contains active ready tasks waiting to use the CPU.

- ACTIVE, NOT READY

  Q2 Special case (not shown in the diagram): monitoring system tasks

  Q3 Waiting for termination of paging I/O

  Q4 Waiting for termination of DMS I/O (disk, tape)
  Waiting for bourse events (“short wait”)
  Waiting for termination of VPASS 0, VPASS MSEC=Y

- INACTIVE, READY

  Q5 Memory queue
  contains inactive ready tasks waiting for authorization to use main memory.
  This queue consists of 4 to 16 sub-queues corresponding to the following predefined categories:

  SYS
  TP
  DIALOG
  BATCH

  and up to 12 further categories which can be defined by systems administration.

  Q6 PCS-Not-Admitted queue
  This queue contains inactive ready tasks which are not allowed to be activated by the Performance Control Subsystem.

- INACTIVE, NOT READY

  Q10 HOLD queue
  Contains newly generated tasks (call: $CREA)
  Waiting for peripheral devices (/SECURE-RESOURCE-ALLOCATION)
  Stopped by systems support (/HOLD-TASK)
  Tasks cannot be terminated (serious errors)

  Q11 Inactive system tasks

  Q12 Waiting for terminal input (WRTRD, RDATA)
  Waiting for bourse events (“long wait”)
  Waiting for an answer to a console message

  Q13 Waiting for termination of PASS, VPASS calls
  Waiting for ITC event

Prior concept

The name PRIOR is understood to encompass those task management routines permitting

- the control of tasks with the aid of categories and priorities
- the monitoring and control of the system workload by means of internal mechanisms.

In effect this implies

- management of the resources main memory (MM) and CPU, including execution of the corresponding control functions
  (see section "Activating/deactivating tasks" and section "Initiation/deinitiation of tasks")
- controlling the tasks via wiat queues
  (see section "Controlling tasks via queues")

The concept is based on priority control without changing the range of resources allocated. This means that the influence of systems support is restricted merely to determining which one of a number of users will first obtain a requested resource. The influence of systems support on what share of a resource's capacity will be available to any applicant within a given time period is negligible.

Task categories

To improve the individual possibilities of control - depending on the respective user requirements - the tasks are subdivided into categories. There are 16 categories.

Four of these are always provided and are given the following default category names:

- SYS: for system tasks only
- TP: Transaction tasks
- DIALOG: Dialog tasks
- BATCH: Batch tasks

Further categories can be set up by systems administration under freely selectable names. Each task is assigned to a category.

In addition, there are four task attributes whose names are the same as the default category names SYS, TP, DIALOG and BATCH. The task attributes are assigned special execution parameters important to task scheduling.

The task attribute TP is different from the other task attributes in that it includes optimized main memory management tailored to fit the requirements of transaction mode (small number of functionally related user tasks, usually with relatively large working set requirements, serving a large number of terminals).
The task attribute TP can be provided by defining it in the job class or by calling the TINF macro (in the latter case the required authorization must have been entered in the user catalog).

The most important attribute of each category is the multiprogramming level (MPL). This refers to the number of tasks per category which are authorized to use main memory, i.e. are in the active state.

Systems support uses the category attributes MIN MPL, MAX MPL and WEIGHT (/MODIFY-TASK-CATEGORIES) to specify the relative importance of the categories with respect to activation (i.e. assignment of authorization to use main memory).

Whether or not a task actually becomes active or is preempted depends on the category attributes, as well as on the system workload and the priority assigned to the task in question (see section “Task priorities”).

Specifying **MIN MPL** guarantees a certain amount of minimum support for a category. The system sets out to reach the specified MIN-MPL value.

Specifying **MAX MPL** does not signify a fixed limit, i.e. activation will take place beyond the MPL value as long as there are no resource bottlenecks.

By means of the **WEIGHT** parameter, the activation or deactivation sequence is controlled. In addition, this parameter has a slight influence on which CPU is allocated.

Modifying the assignment of a task to a category

Systems support can use /MOVE-TASK-TO-CATEGORY to modify the assignment of a task to a category if, for example, different (better) support of this task or a reduction of the load on a category is required (with or without the use of PCS).

Effects of category control

Owing to the above-named attributes, category control has a negligible effect in the lower or normal workload range.

In the **full workload range** or **overload range** (i.e. in the event of resource bottlenecks) category control has a major influence and is therefore used for **load limitation**. Less significant categories are pushed into the background.

> **Important:** This full workload range is only reached in exceptional cases on systems with a very large main memory. Overloading of the CPU is not enough by itself to activate the load limitation facility. For this reason category control is largely ineffective where these systems are concerned.

Task priorities

By means of a priority assignment, the user specifies the urgency with which his/her requests are to be executed. There are 3 priority classes:

Values	Meaning	Priority
0 to 29	System priorities	high
30 to 127	Fixed priorities	medium
128 to 255	Variable priorities	low

The variable and fixed priorities are available to the user. In this form they are referred to as external priorities.

Priorities can be assigned:

- by the user at command level with

```
/SET-LOGON-PARAMETERS
/ENTER-JOB
/ENTER-PROCEDURE
/CHANGE-TASK-PRIORITY
```

- by the user at program level using the TINF macro
- by the system administrator or operator as default value in the job class (RUN-PRIORITY) and with /CHANGE-TASK-PRIORITY

The maximum permissible priority must be entered in the job class or in the user catalog.

Variable priorities

Variable priorities are characterized by the dynamic setting of the internal priority using the HRN algorithm (Highest-Response-ratio Next), based on the ratio Elapsed time / CPU time used and taking into account a “start priority.” The start priority is computed on the basis of the specified external priority, the category weights, the performance power of the system, and the current workload.

This makes for expedient operation, even in the case of tasks with a low start priority.

The variable priority is calculated or upgraded at the following times:

- at activation time (see section "Activating/deactivating tasks")
- at micro time slice runout (see section "Micro Time Slice" (Initiation/deinitiation of tasks))
- periodically (every second).

The period check is performed for all active tasks as well as all inactive tasks ready to execute (i.e. those waiting in the “inactive ready” state for authorization to use main memory).

Fixed priorities

In the case of fixed priorities, the predefined external priority remains unchanged. It is converted without change to the internal priority.

Effects of priority

Since the priority is taken into account during both activation and initiation, issuing a priority which deviates from the standard external priority of 255, i.e. a better start priority, has practically the same effect no matter what the workload is, i.e. high, low or excessive.

When an I/O request is placed in the device queue, the priority is not taken into account. The effects of priority increase in direct proportion to the recourse made to the CPU by competing tasks.

Recommendations and guidelines

PRIOR is used to control the allocation of the resources main memory and CPU to user and system tasks in accordance with the preset category and priority parameters.

In this case priority assignments play an especially important part, since they can be used to give priority to individual tasks, whereas category control can only be used to give priority to entire categories. In the case of TP applications in particular, the priority of the central tasks should be raised. Extreme caution should be exercised when working with fixed priorities. They are not subjected to the HRN algorithm and might therefore monopolize the system. Under no circumstances should priorities better than 80 to 100 be issued.

Note

The parameters for the **SYS category** cannot be changed and are set to the following values:

MIN MPL=30
MAX MPL=64
WEIGHT=512

System tasks have the following priorities:

- Permanent system tasks: 22

  Exceptions:
  TSN VMM: 16
  TSN PGE: 1
  TSN RMM: 128 at startup and 50 after SYSTEM READY
- Dynamically generated tasks (e.g. DIAA, DSSM): 60

  Exceptions (e.g. BCAM): 30

The user should bear in mind that resource shortages cannot be compensated for by the judicious selection of value settings. Although a user with a great deal of experience may be able to alleviate a CPU bottleneck or insufficient main memory, at least for especially important tasks, albeit to the detriment of others, this is scarcely possible when the bottlenecks occur in peripheral devices.

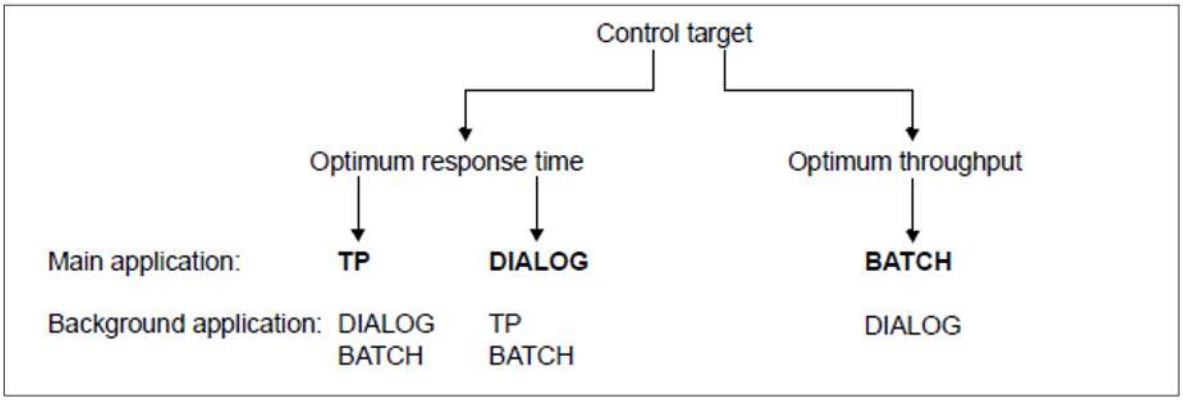
In the sections below, it is assumed that appropriate measures were taken during the capacity planning stage with regard to sizing the system to suit the problems involved. With the main objective being the optimum utilization of the resources installed.

Load analysis

In order to select the best possible values for PRIOR from the wide variety of possible settings, the user must have a clear idea of the workload. The following questions must be answered when analyzing the workload:

- Which applications demand which performance requirements?
- In the event of an overload, which applications can be pushed into the background to the advantage of more important applications?
- Which application belongs to which category?

First it is necessary to define the control target:



The distinction between optimum response time and optimum throughput is important because the two represent competing objectives: good response time behavior presupposes short waiting periods for the resources (especially channels, controllers and disk drives); thus the workload on these resources must not be excessive.

The further distinction within optimum response time as to whether the emphasis should be placed on TP or dialog applications is important for setting the category parameters.

- Dialog applications place a heavier burden on main memory with respect to paging intensity as compared to TP applications (see also section "Examining system performance with openSM2").
- On the other hand, TP applications react more sensitively to background jobs, especially where paging is involved. This sensitivity is even more pronounced the fewer the number of tasks which are responsible for processing the application. The TP application is regarded as the main application if the number of TP transactions is greater than or equal to 50% of the total number of transactions.

The number of transactions per second can be calculated for TP and DIALOG applications either by using the guidelines given in section "Formulating a performance expectation" or on the basis of measurements taken with the SM2 software monitor.

Example: Transaction rate

600 TP terminal users:

Average think time: 20 seconds (incl. network runtime)
Average response time: 1 seconds

50 dialog tasks:

Average think time: 20 seconds (incl. network runtime)
Average response time: 2 seconds

Transaction rate for the TP applications: 600 / (1 + 20) = 28.6 trans/s

Transaction rate for the dialog applications: 50 / (2 + 20) = 2.3 trans/s

In any event, the user should perform a measurement run using the SM2 software monitor (especially in the case of more complex applications). **Measurement of partial loads** is advisable.

The most practical approach is to start with the most important part of the workload, setting the PRIOR parameters for the main application according to the recommended values (see also "Setting the category parameters" and "Assigning priorities") and calculating the response time behavior, the transaction rate and the load on resources. These values are required in order to optimize the main application when operating with mixed loads, so that the user can determine how severely ancillary applications detract from the main application.

By measuring the main application in this manner, the assumptions made at the capacity planning stage can be verified or the actual resource requirements demonstrated. If an unsatisfactory level of performance should arise while the main application is operating by itself, the user should carry out a bottleneck analysis (see section "Basic procedure").

The next step is to set the PRIOR parameters for the background application in accordance with the recommendations (see "Setting the category parameters" and "Assigning priorities") and then to increase the workload step by step.

Example: Response time optimization

Giving the TP application precedence over DIALOG and BATCH applications

- Step 1:

  Measure the pure TP application (modified PRIOR parameters)
  Measurement period depending on workload:
  15 minutes when the load hardly fluctuates
  60 minutes when the load fluctuates greatly

  Determine response times, transaction rate, resource utilization

- Step 2:

  Add the DIALOG application (modified PRIOR parameters)
  Measurement period: 60 minutes

- Step 3:

  Add the BATCH application (modified PRIOR parameters)
  Measurement period: Several measurements, each 60 minutes

  Observe runtime behavior

As the result of the ancillary application, the main application will inevitably be slowed down somewhat (approx. 10 - 20%). This is unavoidable. By judiciously selecting and limiting the ancillary applications, the user can ensure that this delay remains within acceptable limits. The ancillary applications should, wherever possible, only make use of those resources which are scarcely required for the main application. Furthermore high sensitivity of the ancillary applications with respect to MPLs and priorities is an added advantage.

User privileges

Following the workload analysis, it is necessary to finalize user privileges via /MODIFY-USER-ATTRIBUTES or the JOB-CLASS parameters (i.e. define the highest priority a user may specify).

It is also necessary to check whether TP authorization has been entered for DCAM, UTM, UDS and SESAM applications so that it is possible to take full advantage of the TP task attribute.

Setting the category parameters

The primary goal for systems with limited memory space is to apportion **available** pageable main memory efficiently, according to the specific weight of the categories (NPP = number of pageable pages).

Memory capacity is limited if there are INACTIVE READY TASKS despite the fact that reasonable MPL values have been set (report “Active and inactive tasks for category” in the report group CATEGORY-QUEUE of SM2).
If this condition has never occurred during operation, the reader can skip this section and concentrate on the "Assigning priorities".

For PRIOR to be able to react flexibly to slight variations in workload, the working set requirements of the tasks, as represented by the sum of the MIN MPL values for all categories should be: (between 0.3 and 0.5) * NPP. The choice of the factor (0.3 or 0.5) depends on the emphasis desired for the application involved (see below).

Guidelines for the average working set requirements (4KB pages):

TP tasks:

WS_{TP} = 200 to 400 pages for UTM and DCAM tasks
WS_{TP} = 500 to 1.000 pages for DB tasks

Dialog tasks:

WS_{DIAL}= 50 to 150 pages

Batch tasks:

WS_{BATCH} = 50 to 200 pages

The working set for system routines and common memory pool, which are called by application programs, are managed globally in the SYS category (placeholder is TSN=PGE).
Category SYS:

WS_{SYS}= 2,000 to 4,000 pages

See the ACTIVITY report of SM2 for the NPP value.

> In the following, it is assumed for the sake of simplicity that, from the user's viewpoint, only the categories with the standard names TP, DIALOG and BATCH exist.

Main application: TP

Background application: dialog, batch

TP category

This category contains the tasks which place the greatest demands on performance. It is not desirable to limit the workload of this category, even in the event of overload. Therefore the value of MAX MPL must be equal to the total number of TP tasks in the system (including management tasks such as Data Base Administrator, etc.).

The value for MIN MPL should correspond to the desired number of active TP tasks. It is generally equal to or approximately equal to the MAX MPL value. If the value is too low and an overload occurs, this will cause TP tasks to be deactivated automatically (see section "Activating/deactivating tasks").

Recommended values:

MIN MPL_{TP} = 0,8 * MAX MPLTP

MAX MPLTP = Total number of existing TP tasks

WEIGHT = 500

DIALOG category

This category is usually characterized by wide variations in workload. PRIOR attempts to reach the MIN MPL value set by systems support, if necessary by suppressing tasks in other categories (see section "Activating /deactivating tasks").

Since in this case the DIALOG application is meant to run in the background, the value of MIN MPL must not be too high.

Due to the effect of think time, not all dialog tasks are active at the same time. The number of active dialog tasks can be estimated with the aid of the following formula:

- Number of active dialog tasks = total number of dialog tasks / N
- N = (Average think time + Response time) / Average response time

Guidelines:

- Average think time (incl. network runtime) = 16 seconds
- average response time = 4 seconds

N = 20 s / 4 s = 5

In order to calculate the MIN MPL value it is essential to know how much pageable main memory (according to the formula 0.5 * NPP with the emphasis on TP mode) is available for **active** dialog tasks after subtracting the working set requirement of the main application, the SYS category or, if applicable, of any necessary batch tasks for active dialog tasks (see example).

The value of MAX MPL is conceived as a workload limit for temporary overload situations. It should be set to twice the value of MIN MPL.

Recommended values:

MIN MPL_{DIAL} = (0,5*NPP - (MIN MPL_{TP} * WS_{TP} + MIN MPL_{BATCH} * WS_{BATCH} + WS_{SYS})) / WS_{DIAL}

WSTP: average working set requirements per TP task

WSDIAL: average working set requirements per dialog task

WSBATCH: average working set requirements per batch task

WS_{SYS}: average working set requirements for system routines and common memory pools

Note

At this point it is only necessary to take the BATCH portion into account if, for operational reasons, a certain minimum number of parallel batch tasks is absolutely necessary for the corresponding IT installation.

MAX MPL = 2 * MIN MPL
WEIGHT = 100 (50 < WEIGHT < 200)

BATCH category

Normally, batch tasks are run in the background in order to make use of the free resources. In this case, the value of MIN MPL is derived from the size of pageable main memory (more precisely 0,5 * NPP) minus the working set requirement for the main application TP, the DIALOG application and the SYS category.

The value of MAX MPL varies according to the system workload, and should not deviate markedly from the value of MIN MPL.

Recommended values:

$IN\ MPL_{BATCH} = (0{,}5*NPP - (MIN\ MPL_{TP} * WS_{TP} + MIN\ MPL_{DIAL} * WS_{DIAL} + WS_{SYS})) / WS_{BATCH}$

MAX MPL = MIN MPL + 1

WEIGHT = 10 (1 < WEIGHT < 50)

Example for main application TP

MM = 64 MB

Main application TP:

6 UTM tasks, working set for each 300 pages

2 DB tasks, working set for each 800 pages

2 further tasks, working set for each 250 pages

Background application:

10 dialog tasks, working set for each 150 pages

At least 2 batch tasks, working set for each 100 pages

TP category:

$MIN\ MPL_{TP}=8$

$MAX\ \ MPL_{TP}=10$

WEIGHT=500

DIALOG category:

Number of active dialog tasks = total number of dialog tasks / N = 2

$MIN\ MPL_{DIAL}=2$

$MAX\ \ MPL_{DIAL}=4$

WEIGHT=100

BATCH category:

$MIN\ MPL_{BATCH}=2$

MAX MPL_{BATCH}=3

WEIGHT=10

64 MB = 16,000 pages, resident MM requirement = 1500 pages, i.e. NPP = 14,500 pages.

(MIN MPL_{TP} * WS_{TP}) + MIN MPL_{DIAL} * WS_{DIAL} + MIN MPL_{BATCH} * WS_{BATCH} + WS_{SYS} <= 0,5*NPP

(6*300+2*800) + 2*150 + 2*100 + 2000 = 1800+1600+300+200+2000 = 5900

<=0,5*14500 = 7250

Main application: Dialog

Background application: TP, batch

TP category

The fact that the TP application appears as a background application does not necessarily imply that minimal or even no demands are placed on performance; it only means that the number of TP transactions is less than 50% of the total number of transactions.

This is always the case whenever new TP applications are introduced and the number of terminal users employing the new procedure is still relatively small. Since a **single** TP task always serves several terminal users, and also normally has greater working set requirements as compared to dialog tasks, it is generally not desirable to have TP tasks automatically deactivated in the event of overload.

Therefore, the value of MAX MPL should likewise be equal to the total number of TP tasks in the system. Similarly, the value of MIN MPL should be equal or approximately equal to the value of MAX MPL.

Recommended values:

MIN MPL_{TP2} = 0,8 * MAX MPL_{TP}

MAX MPL_{TP2} = total number of TP tasks

WEIGHT = 500

DIALOG category

If the emphasis of the application lies on the DIALOG side, this entails higher demands on main memory with respect to paging intensity (see section "Examining system performance with openSM2") as compared to TP applications.

Therefore, the working set requirements of the tasks as represented by the sum of the MIN MPL values of the TP, DIALOG and BATCH categories should be <= 0.3 * NPP.

For calculating the MIN MPL value, it is essential to know how much pageable main memory (according to the formula 0.3 * NPP with the emphasis on DIALOG mode) is available for active dialog tasks after subtracting the working set requirements of the TP tasks, the SYS category or necessary batch tasks for active dialog tasks (see example).

Here, too, the number of active dialog tasks (initial estimation) should be set to

- Number of active dialog tasks = total number of dialog tasks / 5

Recommended values:

$MIN\ MPL_{DIAL}=(0,3*NPP - (MIN\ MPL_{TP} * WS_{TP} + MIN\ MPL_{BATCH} * WS_{BATCH} + WS_{SYS})) / WS_{DIAL}$

Note

At this point, it is only necessary to take into account the BATCH portion if, for operational reasons, a certain minimum number of parallel batch tasks is required.

MAX MPL = 2 * MIN MPL

WEIGHT = (300 < WEIGHT < 500)

BATCH category

Given that batch tasks are running solely as a background workload, the following values apply:

$MIN\ MPL_{BATCH} = (0,3*NPP - (MIN\ MPL_{TP} * WS_{TP} + MIN\ MPL_{DIAL} * WS_{DIAL} + WS_{SYS})) / WS_{BATCH}$

MAX MPL = MIN MPL + 1

WEIGHT = 10 (1 < WEIGHT < 50)

Example for main application Dialog

MM = 64 MB

Main application DIALOG:

45 dialog tasks, working set for each 100 pages

BATCH:

At least 2 batch tasks, working set for each 100 pages

Background application:

2 UTM tasks, working set for each 200 pages

1 DB task, working set for each 500 pages

1 further task, working set for each 150 pages

TP category:

$MIN\ MPL_{TP}=3$

$MAX\ \ MPL_{TP}=4$

WEIGHT=500

DIALOG category:

Number of active dialog tasks = total number of dialog tasks / N = 9

$MIN\ MPL_{DIAL}=9$

$MAX\ \ MPL_{DIAL}=18$

WEIGHT=400

BATCH category:

MIN MPL_{BATCH}=2

MAX MPL_{BATCH}=3

WEIGHT=10

64 MB = 16,000 pages, resident MM requirement = 1500 pages, i.e. NPP = 14,500 pages.

(MIN MPL_{TP} * WS_{TP}) + MIN MPL_{DIAL} * WS_{DIAL} + MIN MPL_{BATCH} * WS_{BATCH} + WS_{SYS} <= 0,3*NPP

(2*200 + 1*500) + 9*100 + 2*100 + 2000 = 400+500+900+200+2000 = 4000

<= 0,3*14500 = 4350

Main application: Batch

Background application: Dialog

This load is typical for night-shift operation of systems whose main memory is dimensioned for daytime online operation. As a rule, batch applications place a considerably lighter burden on main memory than online applications with regard to paging intensity; however, they put a far greater burden on the CPU.

The number of batch tasks that can be handled thus depends less on the main memory size than on the speed of the CPU.

The settings for the category parameters is of no significance here (the required load limitation can be specified using the job classes parameter CLASS-LIMIT).

Assigning priorities

By assigning a priority higher than the default priority of 255, it is possible to give preference to individual tasks with regard to activation and, especially, to the utilization of the resource CPU.

As a rule, assigning a variable priority of medium size is sufficient.

Variable priority

Priority should be varied in accordance with the workload in units of 5 or 10 until the desired performance target is attained. Larger priority increments are required when

- many tasks are active
- the active tasks are predominantly I/O-intensive.

Depending on the control target, the following values are recommended for the initial setting:

Main application: TP

Background application: dialog, batch

TP category

> If a DB/DC application with several tasks is involved, database systems which operate in multithread mode (SESAM/SQL, UDS) should be assigned a lower priority than the associated DC tasks (UTM, DCAM, ...). This allows a considerable reduction in the necessary communication between the DB and DC tasks for each database call.
>
> If there are two or more TP applications of different weight, the priorities may be assigned incrementally.
>
> Recommended priority range: 130 - 180

DIALOG category

> The dialog application is to run in the background. As a rule, no preference is given to individual dialog tasks in this category.
>
> Recommended priority: 210

BATCH category

> For pure background batch tasks, the default priority is sufficient. To emphasize tasks within the category, it is sufficient to raise the priority to 240 (maximum).
>
> Recommended priority range: 240 - 255

Main application: Dialog

Background application: TP, batch

DIALOG category

> In general, dialog tasks are treated equally as far as priority is concerned.
>
> Recommended priority: 180

TP category

Unlike the setting of category parameters, where an attempt is made to avoid deactivation of TP tasks even when an overload occurs, the background application TP should receive lower priority with regard to the use of the resource CPU.

If, in the case of dialog tasks, loads similar to the batch category are found (compilations, programming sequences within a response time) processing should continue as for the main application TP.

Recommended priority: 210

BATCH category

When operating solely in the background, the default priority is sufficient.

Recommended priority range: 230 - 255

> Under normal circumstances, batch tasks are run in the background, with the purpose of taking advantage of free resources. All tasks which are background batch tasks should be executed with the lowest priority (255), whereas all other tasks should have a priority of 210 or better. Regardless of whether the TP or DIALOG category is the main application, batch tasks which execute in the background must consist of compute-intensive programs, so that the system can regulate the use of available resources. These tasks are then made to handle their I/O operations via channels and disk drives which are not being used for foreground tasks.

Fixed priorities

Fixed priorities are provided for special applications with extremely high real-time requirements. The following points should be borne in mind in this connection:

Fixed priorities severely restrict the system's freedom to make decisions. Tasks awaiting activation or initiation cause other tasks to be immediately suppressed when resources are being used to full capacity.

Therefore, to ensure improved system behavior, fixed priorities should be chosen only after a thorough analysis of the workload and the allocation of resources has been made and in conjunction with steps to limit the workload (MAX MPL).

Tasks given a fixed priority should not be compute-intensive, neither in part nor as a whole, since they are subject to stringent demands with regard to freedom from errors.

I/O priority handling for tasks using IORM

Normally all tasks have the same priority when disk I/Os are executed.

An I/O-intensive application with low priority can hinder another, higher-priority application if these application execute I/Os on the same (logical) device. Operations can also be impaired if the I/Os are executed on different (logical) devices which are located on the same physical device or are connected via the same paths, are accessible via the same ports or connected to the same channels.

The IOPT (I/O Priority Handling for Tasks) function enables IORM to detect such conflict situations and to intervene in I/O operation for control purposes. Here IOPT examines both the utilization level of the I/O units (devices, paths, ports and channels) and the I/O priorities of the tasks which use them.

IOPT distinguishes three I/O priorities for tasks:

- HIGH (high I/O priority)
- MEDIUM (medium I/O priority)
- LOW (low I/O priority)

It is possible to assign the task categories (with the exception of SYS) corresponding I/O priorities (IO-PRIORITY attribute) using the following command:

/MODIFY-TASK-CATEGORY IO-PRIORITY=*NONE/*HIGH/*MEDIUM/*LOW

Further information on IORM and IOPT can be found in the “Utility Routines” manual [6 (Related publications)].

PCS concept

PCS (Performance Control System) is available as a performance control subsystem for regulating loads; it is based on PRIOR and provides the following functions:

- Improved preferential treatment of load components thanks to targeted SERVICE allocation; the SERVICE of a server is determined from the time the CPU and main memory (MEMORY) are reserved and from the number of I /O operations (IO).
- Increased transparency with regard to the distribution of system capacity over the various categories
- Dynamic adjustment of the service allocation to categories or tasks in accordance with fluctuations in load
- Throughput control by monitoring the capacity consumption in the categories
- Differentiation between “short-running jobs” and “long-running jobs” and thus improved control possibilities via automatic category changeover
- Improvement of the response time by means of the preferential treatment of short-running jobs.
- Differentiation in the control of a number of TP applications through allocation to several categories

PCS operation can be controlled by means of global system parameters and/or category-specific parameters:

Category-specific control parameters

With the aid of the SERVICE-QUOTA (S-Q) parameters, the performance capacity of the system is distributed among the categories. The parameters SERVICE-QUOTA-MIN and SERVICE-QUOTA-MAX define a framework within which PCS plans service requirements for the specified category (S-Q-PLN). If a category does not make full use of its share of performance, the remainder is then available to other categories.

By means of the dilation factor (REQUEST-DELAY parameter), PCS recognizes how extensively the tasks of this category are delayed by competing tasks. Dilation thus serves to indicate workload levels and specifies the factor by which the runtime of a job will be delayed if the job is forced to wait for the allocation of a resource owing to multiprogramming operation.

- Dilation = Runtime in multiprogramming operation / Single runtime

In the case of the simultaneous use of SM2 and the accompanying SM2 monitoring program SYSTEM for all disks, PCS periodically measures the current I/O time per category, which increases the accuracy of the dilation calculation accordingly.

Within a given dilation range (REQUEST-DELAY-MIN, REQUEST-DELAY-MAX), PCS dynamically adjusts the S-Q-PLN value of the category in accordance with the current load (REQUEST-DELAY-ACTUAL).

SERVICE is distributed over the various individual categories according to the following hierarchy:

1. Categories with a dilation range and max. share of performance of 100%

   These categories are allocated the performance capacity to which the current dilation entitles them before all other categories (“planned values”).

   Insufficient capacity:
   If several such categories exist and if the sum of their planned values exceeds 100%, the planned values are reduced proportionally so that they total 100%.

   Residual capacity:
   If the tasks of the categories with a dilation range and max. share of performance of 100% are not in a position to use the planned capacity, the (current) residual capacity is made available to the other categories with and without dilation range. (Exception: full global response-time orientation).
2. Categories with a dilation range and max. share of performance < 100%

   These categories are allocated the performance capacity which is left over by categories with a dilation range and max. share of performance of 100% and to which their current dilation entitles them, before categories without a dilation range.

   Insufficient capacity:
   If several categories with a dilation range exist and if the sum of their planned values exceeds the capacity still available, the planned values are reduced proportionally so that their sum is equal to the available capacity.

   Residual capacity:
   If the sum of the planned values is less than the available capacity, the residual capacity is planned for use by categories without dilation range.
   If the tasks of the categories with a dilation range are not in a position to use the planned capacity, the (current) residual capacity is made available to the categories without dilation range. (Exception: full global response-time orientation).
3. Categories without dilation range

Categories without dilation range are allocated the performance capacity which is left over by the categories with a dilation range. This capacity is distributed proportionally among the categories without dilation range according to their defined maximum shares of performance.

In order to take full advantage of the regulation functions of PCS, at least one category without a dilation range should always be provided.

The function “automatic category changeover” serves to improve differentiation between load requests with high resource requirements (long-running jobs) and those with low resource requirements (short-running jobs). In this case the user employs the DURATION variable to specify how many SERVICE UNITs may be used per processing step before an automatic changeover to the following category (NEXT CATEGORY) with a “weaker” performance capacity is to take place.
SERVICE UNITs are the weighted sum of the work carried out by the resources CPU, IO and MEMORY.

By means of the THROUGHPUT-QUOTA parameter, a percentage indicating the ratio between response-time and throughput optimization within the category is defined. The effects of this are as follows:

THROUGHPUT-QUOTA	response time optimization		throughput optimization	
	full			full
	0%	<50%	>50%	100%

For systems with TP or interactive mode, response-time optimization is of more importance; THROUGHPUT-QUOTA=0 is therefore recommended in the corresponding TP categories as well as in the DIALOG start category (short-running jobs).

For systems operating predominantly in batch mode, the most important performance goal is the greatest possible throughput together with high utilization of the resource capacity. For throughput-oriented categories, THROUGHPUT-QUOTA values greater than 50% should be provided.

Global system control parameters

Global dilation throughout the system (REQUEST-DELAY-MAX) is a threshold value for the dilation of all tasks belonging to a category with a dilation range. When this value is exceeded, load limitation goes into effect.

THROUGHPUT-QUOTA specifies a percentage value that determines the relation between response-time and throughput optimization for the entire system. The parameter has a very pronounced effect on the limit values:

- THROUGHPUT-QUOTA=0
  results in hard response-time optimization. This means that background categories are totally suppressed in the event of an overload (exceeding the value of the global system parameter REQUEST-DELAY-MAX). As a consequence, utilization of the resources and thus the throughput may be impaired.
- THROUGHPUT-QUOTA=100
  is only used for pure batch mode.

Procedure when using PCS

As a rule, the hardware configuration (is the configuration adequate to deal with load requirements?) and the software configuration (location and size of the system files TSOSCAT, paging areas and SYSEAM) should be checked before the system is used for the first time.

Depending on the particular load situation, first the predefined default options

- STD#TP (predominantly TP mode),
- STD#DIA (predominantly interactive mode) and
- STD#BATCH (predominantly batch mode)

should be used and the performance results measured (for more details see the “PCS” manual [23 (Related publications)]). The results should be at least as good as those achieved with PRIOR, but are generally better. If the check shows that performance goals have not been achieved in individual categories, the service planning values for these categories must be increased. The current dilation value of REQUEST-DELAY-ACT constitutes the main criterion here (information can be obtained with /SHOW-PCS-OPTION CATEGORY-NAME=*ALL or from the report “Request delay for category” in the report group PCS of SM2).

In transaction mode and interactive mode the user is advised to proceed step by step as described below in order to increase the service values in the categories in question:

1. If the current dilation is between the values for REQUEST-DELAY-MIN and REQUEST-DELAY-MAX, first of all reduce REQUEST-DELAY-MAX for the appropriate categories. As a result, the value for SERVICE-QUOTA-PLAN will increase. The optimum work point is reached when REQUEST-DELAY-ACT is slightly below the value of REQUEST-DELAY-MAX.
2. Increase the SERVICE-QUOTA-MAX value for the appropriate category. The ratio of S-Q-MAX / S-Q-MIN <= 3 is recommended
3. Increase the value of SERVICE-QUOTA-MIN. This parameter serves the special function of reserving capacity to cater for sudden load fluctuations.

If these actions do not produce the desired results, i.e. shorter response times, further tuning measures are required; these vary depending on which mode of operation has priority.

When interactive mode has priority, long-running transactions can be suppressed by taking the following actions:

1. Reduce S-Q-MAX for the next category (Dialog 1)
2. Reduce S-Q-MIN for the next category (Dialog 1)

When TP mode has priority, first the service planning value for the DIALOG background load can likewise be reduced by reducing the value of S-Q-MAX and S-Q-MIN for the DIALOG category.
If there are a number of mutually dependent tasks in the TP category (e.g. UTM/UDS, UTM/SESAM), it is also possible to increase the priority for those tasks with greater service requirements.

If there are several applications in the TP category, the TP response times can generally be improved by increasing the service planning value. When the aim is targeted improvement of individual applications, it is advisable to keep each application in a separate TP category.

If you wish to ensure that a category is allocated whatever capacity it needs (if necessary, the entire capacity of the system), enter the value SERVICE-QUOTA-MAX=100 for that category. It does not make sense to give this value to two or more categories.

All modifications to the PCS default settings described here must be performed by systems administration with the aid of the PCSDEFINE utility routine. Once PCS has been started as a subsystem, systems support can obtain information about PCS using system commands and dynamically update the control parameters (for more details see the “PCS” manual [23 (Related publications)]).

Enhancements to PCS V2.7 and higher:

- The BS2000 command MOVE-TASK-TO-CATEGORY was introduced in particular to provide individual tasks with a higher service allocation in exceptional cases. This enables tasks to be placed in a different category (defined in JMU).
- The MODIFY-TASK-CATEGORY command, IO-PRIORITY operand, enables all tasks in a category to be connected to the priority control of overloaded volumes by means of IORM.

Examples of the effect of the PCS parameters

Example 1

Effect of the default option STD#DIA in predominantly interactive mode for developing programs with a batch load in the background.

Default option parameter STD#DIA:

Category	SERVICE-QUOTA=		REQUEST-DELAY=		DURATION	NEXT CATEGORY	THROUGHPUT-QUOTA
	MIN	MAX	MIN	MAX			
TP	0	30	1	3	-	-	-
DIALOG	20	40	2	4	500	DIALOG1	-
DIALOG1	10	30	2	5	2000	DIALOG2	10
DIALOG2	0	50	-	-	-	-	20
BATCH	0	20	-	-	-	-	70
BATCHF	0	20	1	3	-	-	-

All incoming interactive transactions are first accepted by the default category DIALOG. As a rule, short interactive jobs such as EDT statements require less than 200 SERVICE UNITs and thus are retained in the DIALOG category until the transaction has ended.

If the service requirements exceed 500 SERVICE UNITs (e.g. in the case of file operations), an automatic changeover to the “weaker performance” category DIALOG1 is effective.

Compiling and program testing tasks with a service requirement of more than 2000 SERVICE UNITs are switched to category DIALOG2 (no REQUEST-DELAY entry, i.e. no request for response time), where they execute until completed.

As compared to the throughput-oriented category BATCH (THROUGHPUT-QUOTA=70) for the background batch load, long-running interactive jobs in category DIALOG2 are given priority as the result of the larger SERVICE-QUOTA-MAX value.

In exceptional cases the BATCHF (batch fast) category must be used for particularly important batch runs.

Using PCS results in considerably improved response times for short-running interactive jobs when compared to PRIOR, even if PRIOR is running efficiently. The response times for interactive transactions with high service requirements are longer.

Example 2

Effect of the default option STD#TP in predominantly TP mode with an interactive and batch load in the background.

Default option parameter STD#TP:

Category	SERVICE-QUOTA=		REQUEST-DELAY=		DURATION	NEXT CATEGORY	THROUGHPUT-QUOTA
	MIN	MAX	MIN	MAX			
TP	50	100	1	3	-	-	-
DIALOG	10	30	2	6	500	DIALOG1	-
DIALOG1	0	30	-	-	-		10
BATCH	0	10	-	-	-	-	70
BATCHF	0	20	1	3	-	-	-

Categories with a dilation entry (REQUEST-DELAY) are given priority over categories without dilation.

If the total of the SERVICE-QUOTA-MAX values of the categories with dilation exceeds 100%, appropriate standardization measures are implemented, i.e. the competition of the categories relative to one another remains the same. By explicitly specifying SERVICE-QUOTA-MAX=100 for TP it is ensured that the background load will be fully suppressed if so required.

All incoming interactive transactions are first accepted by the DIALOG category. If the service requirements exceed 500 SERVICE UNITs, changeover to the category DIALOG1, in which no response time requirements are made, is automatic.

Prioritizing the more resource-intensive interactive transactions in DIALOG1 at the expense of the real batch tasks in the BATCH category is accomplished by means of the higher SERVICE-QUOTA-MAX value (=30).

In exceptional cases the BATCHF (batch fast) category must be used for particularly important batch runs.

When TP mode is given priority, PCS brings about a dramatic reduction of response times for TP applications in comparison to PRIOR. The short-running jobs of the background load DIALOG also benefit, while the long-running jobs are heavily suppressed. Satisfactory TP response times during heavy workload situations cannot be achieved with variable priorities under PRIOR. Under PCS control, all improvements in performance can be easily achieved using **one single control parameter**.

Example 3

Controlling more than one TP application

To control TP applications, each application, or a number of equal-ranking applications, are placed in a specific category. The next option is used to control 3 TP applications and an interactive load and batch load for the remaining capacity. The 3 TP applications are assigned to the categories TP, TP2 and TP3 according to their priority.

Categories TP2 and TP3 must be defined as separate categories using the task attribute TP in the DEFINE-JOB-CLASS statement (see the “Utility Routines” manual [6 (Related publications)]).

PCS control parameter:

Category	SERVICE-QUOTA=		REQUEST-DELAY=		DURATION	NEXT CATEGORY	THROUGHPUT-QUOTA
	MIN	MAX	MIN	MAX			
TP	20	40	1,0	2,5	-	-	-
TP2	15	25	1,0	3,0	-	-	-
TP3	10	15	1,0	3,5	-	-	-
DIALOG	5	10	2,0	5,0	300	DIALOG1	10
DIALOG1	0	30	-	-	0	-	20
BATCH	0	20	-	-	-	-	70
BATCHF	0	5	1,0	4,0	-	-	-

Categories with a dilation entry (REQUEST-DELAY) are given priority over categories without dilation. All TP transactions are given priority over interactive transactions owing to the more favorable values for SERVICE-QUOTA and REQUEST-DELAY.

Up to 80% of the systems performance capacity can be assigned for the TP categories (SERVICE-QUOTA-MAX for TP, TP2 and TP3).

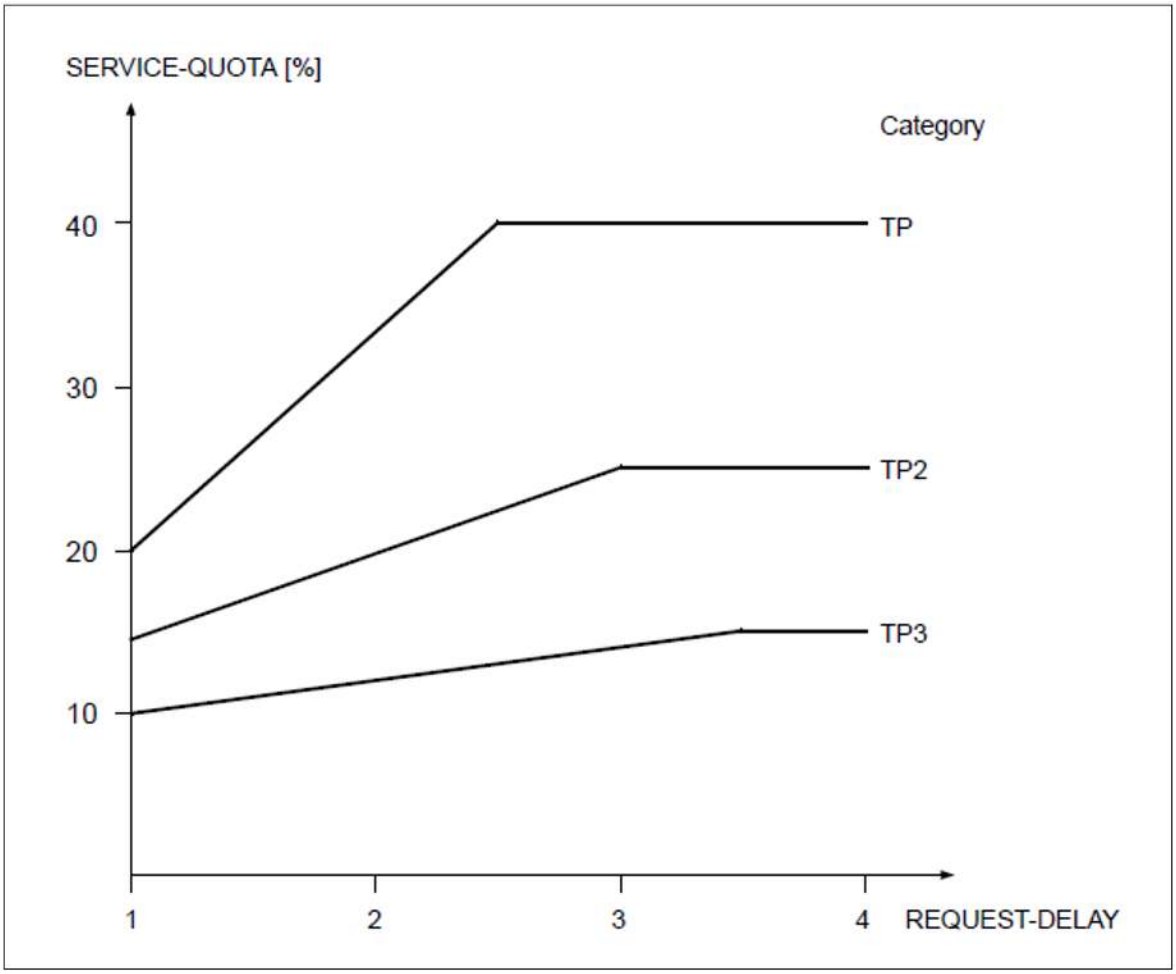


Figure 8: Proportion of TP applications in the performance capacity dependent on the current dilation

The figure 8 shows the proportion of TP applications dependent on the current dilation. When the workload fluctuates (which is expressed by the dilation) a fluctuating proportion of the performance capacity must be reckoned on.

For applications in the TP category, the value SERVICE-QUOTA MAX (40%) is planned for a dilation of 2.5 (REQUEST-DELAY MAX). If the current dilation value is between 1 and 2.5, a service planning value between 20% and 40% is assigned.

Applications in categories TP2 and TP3 are assigned a correspondingly smaller proportion of the performance capacity since the values for SERVICE-QUOTA and REQUEST-DELAY are lower.

The incoming interactive transactions are stated in the DIALOG category. If the service planning requirements exceed 300 service units, the interactive jobs are placed in the lower-performance category DIALOG1. Specifying the value 10 under THROUGHPUT-QUOTA for the DIALOG category reduces the activation priority for interactive transactions.

The more complex interactive transactions in the next category DIALOG1 are given priority over genuine batch jobs in the BATCH category by means of a higher SERVICE-QUOTA-MAX value.

In exceptional cases the BATCHF (batch fast) category must be used for particularly important batch runs.

In the case of larger systems with a very wide variety of workloads, rapidly changing performance requirements and a large number of competing tasks, desired performance goals can only be achieved by using PCS. The more heterogeneous the workload, the greater the number of tasks; the more varied the resource requirements of the individual load components, the more effective PCS will be.

TANGRAM concept

The objective of the TANGRAM (task and group affinity management) concept is to make better use of the performance available from multiprocessor hardware.

All the servers use fast caches to minimize the differential between CPU speeds and access times to main memory. Depending on the instruction profile (in particular the number of accesses to main memory required), the possible performance is to a large degree determined by the hit rate in the cache.

There are two types of cache, which are generally combined:

- Store-Through Caches (STCs)
  The data is “passed through” as soon as possible, i.e. the data in main memory is updated as soon as possible after a write request.
- Store-In-Caches (SICs)
  The data is retained in the cache and is written back to main memory on the basis of the LRU (least recently used) principle. This means that the data in main memory is not up to date.

Whereas in the case of monoprocessors, full performance can be achieved using a sufficiently dimensioned cache, the situation with multiprocessors is far more complex.

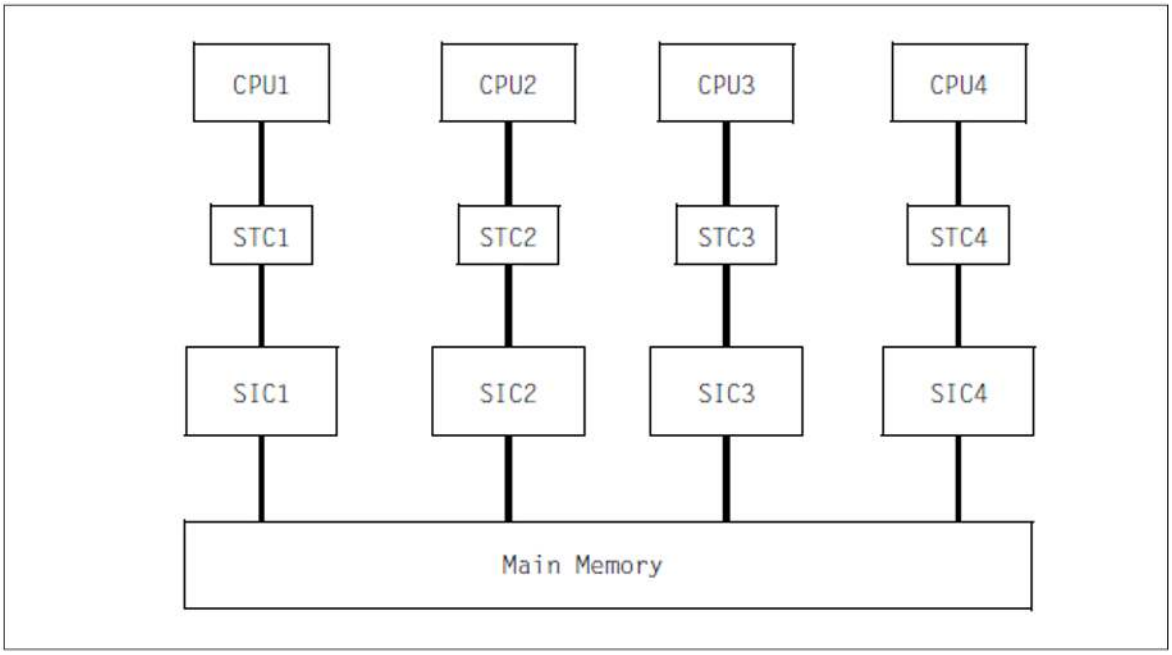


Figure 9: Cache hierarchy in a quadroprocessor with First-Level Store-Through-Caches and Second-Level Store-In-Caches

The greater the number of tasks running on different CPUs which access the same memory areas (e.g. memory pools), the greater the probability that the current information is not located in the CPU's own cache (SIC), but in a neighboring cache.

The current data must first be moved to the CPU's own cache before it can continue processing, with the result that the CPU resources available to the user are reduced. This situation is exacerbated as the proportion of write accesses to the same main memory pages rises.

The TANGRAM concept attempts to counteract this loss of performance by forming task groups and assigning them to certain CPUs (preferably a subset of the CPUs available).

Formation of task groups

A task group is made up of so-called “affined tasks” which have write access to a large set of common data.

The TINF macro is used to sign onto a task group. The TANGBAS subsystem, which is started automatically when the system is ready, is responsible for managing the task groups.

> The software products UTM, UDS and SESAM/SQL automatically issue this TINF call when each application or database system is started. The tasks of a single UTM application thus form a single task group.

Assignment to CPUs

Task groups are assigned to CPU groups dynamically, depending on the load, if viewed over longer periods.

For shorter periods (PERIOD parameter with a default value of 10 seconds), on the other hand, fixed assignments apply, in order to alleviate the problems with the caches described above.

The **assignment algorithm** in the TANGRAM subsystem, which must be started explicitly, measures the following values at periodic intervals:

- CPU usage of each task group
- workload of each CPU
- CPU total workload

If necessary, the task groups are reassigned to the CPU groups according to the following criteria:

- The task group must have a minimum CPU requirement (THRESHOLD parameter, with a default of 10%), before it is taken into account for individual assignment to a CPU.
- If a task group loads one or more CPUs to a great extent (CLEARANCE parameter, with a default of 20%, corresponding to a CPU workload of 80%), it is assigned one more CPU in order to ensure that any increase in load for this task group in the next time interval can be adequately handled.
- Task groups are assigned to CPUs in such a way that all the CPUs have a similar workload, as far as this is possible.

At each change of task, the **scheduling algorithm** in PRIOR checks whether a task is allowed to run on the CPU according to the assignment made by the assignment algorithm. An “idle in preference to foreign” strategy is applied, i.e. idle time in a CPU is preferred rather than initiating a foreign task (which would have to rebuild the contents of the cache entirely).

Use of TANGRAM

Improved use of the performance available from multiprocessor hardware depends on the following conditions:

- number of CPUs (greater advantages with more CPUs)
- Hardware architecture (large Store-In Caches)
- application (proportion of write operations, options for distribution over a subset of the available CPUs)
- workload (significant advantages are only achieved with high workloads)

It is difficult to make generalized statements with regard to the advantages of using TANGRAM. Measurements with different workloads showed an increase in hardware performance of approx. 5% with biprocessor systems and approx. 10% with quadroprocessor systems.

Job management

The job management system (JMS) provides a number of control options in the selection of jobs for processing.

Job classes serve to define certain job characteristics. In addition to the parameters for the selection of jobs, job classes also include statements important for subsequent processing. These parameters can furthermore be used to control the execution of other applications, such as TP or interactive applications. Therefore specification of a job class is not just important for batch processing.

Selection strategies for jobs (job scheduling strategies) selected by the user are implemented with the aid of job streams.

Job classes

The parameter JOB-TYPE=*BATCH of a job class specifies that this class is subject to job scheduling. Jobs of a job class to which the parameter JOB-TYPE=*DIALOG applies are started immediately, provided that this is permitted by the CLASS-LIMIT parameter.

Selection parameters for batch jobs

`CLASS-WEIGHT`

Designates the urgency of the jobs waiting in a job class as compared to another class (important only after system saturation occurs).

Value range <integer 1..9>; 1 is the lowest urgency, 9 is the highest urgency.

`JOB-PRIORITY`

Defines the importance (priority) of the waiting jobs within a job class.

Value range <integer 1..9>; 1 is the highest priority, 9 is the lowest priority.

`CPU-LIMIT`

Determines the CPU time which a job in this job class may use.
The parameter NO-CPU-LIMIT=*YES has the same effect as the entry PRIVILEGE=*NO-CPU-LIMIT in the user catalog.

`START`

Specifies the possible start times (*SOON, *WITHIN,...).
The additional condition ALLOWED=*IMMEDIATE has the same effect as the entry PRIVILEGE=*START-IMMEDIATE in the user catalog. This means that a job may be started immediately, even if the job class limit has already been reached.

If, in an XCS network, the HOST=*ANY operand is specified in /ENTER-JOB or /ENTER-PROCEDURE, the jobs are automatically distributed to the individual computers in the network (for details, see the "HIPLEX MSCF" manual [12 (Related publications)]).

To select the target system, the criteria are considered in the following order:

1. Status of the job class

   The desired job class must be defined and active (it must not have been suspended by means of /HOLD-JOB-CLASS, for example).
2. The system must not be in a state of saturation.
3. Utilization of the job class

   As long as the job class limit (CLASS-LIMIT) has not been reached, for jobs that are to be executed immediately the difference between the job class optimum (CLASS-OPTIMUM) and the number of jobs already started is taken.

   If the job class limit has been reached or exceeded, the difference between the job class limit and the number of accepted (i.e. already started and pending) jobs is taken.

Parameters for executing batch jobs, TP applications and interactive applications

The following parameters are important when it comes to executing applications in the desired category, taking into account the relevant task attribute TP, DIALOG or BATCH (see also section "Task management") and the desired priority.

```
TP-ALLOWED     = *NO / *YES(CATEGORY=<name>)
DIALOG-ALLOWED = *NO / *YES(CATEGORY=<name>)
BATCH-ALLOWED  = *NO / *YES(CATEGORY=<name>)
```

The above parameters define the task attribute for the job class. This serves to assign special internal execution parameters, optimized for the operating mode TP, interactive or batch.

If no category name has been specified, the associated default category name is assumed (e.g. for TP-ALLOWED=*YES the default category name is TP).

A category name must not be assigned to more than one task attribute.

`START-ATTRIBUTE = *BATCH / *DIALOG / *TP`

This parameter defines the task attribute used to start tasks of this job class. The parameter values for START-ATTRIBUTE and TP-ALLOWED, DIALOG-ALLOWED and BATCH-ALLOWED must be mutually compatible.

`RUN-PRIORITY`

This parameter defines the execution priority (= task priority).

The parameter RUN-PRIORITY=*PARAMETERS (DEFAULT=n, MAXIMUM=n) an be used to define the maximum task priority a user may assign.

Example

An important production application (job handling) is to be run as a TP application with high priority and in a separate category.

With the aid of the JMU utility routine (see the “Utility Routines” manual [6 (Related publications)]) a job class is defined (//DEFINE-JOB-CLASS), the access right is specified (//GRANT-JOB-CLASS-ACCESS) and storage in the file SJMSFILE is performed.

```
//DEFINE-JOB-CLASS NAME=PRIO1,
  CLASS-LIMIT=5,
  CLASS-WEIGHT=2,
  JOB-PRIORITY=*PARAMETERS(DEFAULT=3),
  JOB-TYPE=*BATCH,
  TP-ALLOWED=*YES(CATEGORY=<name>),
  START-ATTRIBUTE=*TP,
  RUN-PRIORITY=*PARAMETERS(DEFAULT=150,MAXIMUM=128),
  NO-CPU-LIMIT=*YES,
  CPU-LIMIT=*PARAMETERS(DEFAULT=*NO-LIMIT,MAXIMUM=32767),
  SYSLST-LIMIT=*PARAMETERS(DEFAULT=*NO-LIMIT),
  START=*PARAMETERS(DEFAULT=*SOON,ALLOWED=...)
```

//SET-MODIFICATION-MODE is used to specify whether the JMU utility routine is to put the modification into effect immediately, in the current session, or/and write it in the SJMSFILE file, which is not evaluated until the next system startup.
The production application can be started simply by issuing /ENTER-JOB with JOB-CLASS=PRIO1.

Note

For reasons of compatibility, a logical OR link between the entries for the job class and the user ID is provided, the respective “higher” authorization having priority in each case.

Example

Job class	**User catalog**	**Status**
`TP-ALLOWED=*NO`	`MAX-ALLOWED-CATEGORY=TP`	TP permitted

`RUN-PRIORITY=`
`  *PARAMETERS(DEFAULT=150,`
`  MAXIMUM=128)`
	`MAXIMUM-RUN-PRIORITY=180`	maximum priority=128

Job streams

BS2000 is supplied with predefined job classes. If no job classes are defined in a BS2000 system, all jobs are assigned to class $SYSJC (to which users have no access) and managed by the **system job scheduler $SYSJS**, which is permanently coupled to the system.

The system job scheduler is a system task which executes as a privileged task in CPU state TPR. Its selection strategy is based on the LIFO (Last In First Out) principle and cannot be influenced.

Independent of the job classes defined and any selection strategies which may have been implemented, the system job scheduler permits jobs to be started under the system administrator ID TSOS at any time during the session.

Job classes defined with the parameter JOB-TYPE=*BATCH are assigned (STREAM=name) to a job stream (also known as a job scheduler). A job class can only belong to **one** job stream, but a job stream can manage more than one job class.

The jobs in the stream are sorted by the **standard job scheduler** according to a prescribed selection strategy. The standard job scheduler is an application program that executes in CPU state TU under the system administrator ID TSOS. The selection strategy is specified by means of the stream parameter (STREAM -PARAMETER). It is set by means of the JMU utility (//DEFINE-JOB-STREAM).

In order to implement various selection strategies, the scheduling parameters specified for the stream definition can be dynamically updated (using //MODIFY-JOB-STREAM of the JMU utility routine).

Up to 16 job streams can be installed at the same time.

Selection strategies (STREAM-PARAMETER='...')

- Selection according to arrival time:

  `JOB-PRIORITY=NO,CPU-TIME=NO,WAIT-TIME=NO`

  WAIT-TIME defines the waiting time of the job after acceptance.
  This strategy is advisable if the CPU time requirements for the different jobs are more or less the same.
- Selection according to priority:

  `JOB-PRIORITY=YES,CPU-TIME=NO,WAIT-TIME=NO`

  This ensures that important jobs will be given priority.
- Selection according to CPU time required:

  `JOB-PRIORITY=NO,CPU-TIME=YES,WAIT-TIME=NO`

  Jobs with lower CPU time requirements are given priority.
- Selection according to priority and CPU time requirements:

  `JOB-PRIORITY=YES,CPU-TIME=YES,WAIT-TIME=NO`

  When different jobs have the same job scheduling priority, those requiring less CPU time are given priority.
  When the CPU time required is the same, job scheduling priority is the criterion for selection.
- Selection according to priority and waiting time:

  `JOB-PRIORITY=YES,CPU-TIME=NO,WAIT-TIME=YES`

  By including the waiting time after job acceptance, jobs with a lower priority also have a chance of being selected.

- Selection according to throughput:

  `JOB-PRIORITY=NO,CPU-TIME=YES,WAIT-TIME=YES`

  Jobs with low CPU time requirements are given priority, taking into account the waiting time following acceptance.
- Selection according to throughput and priority:

  `JOB-PRIORITY=YES,CPU-TIME=YES,WAIT-TIME=YES`

  In addition to selection according to throughput, the job scheduling priority is taken into account.

The job scheduler passes the jobs it has selected and sorted to the **class scheduler**.

The class scheduler is not a separate task, but belongs to the core of the job management system. It ensures that the job mix desired by systems support (via the CLASS-LIMIT parameter of the individual job classes) will be adhered to.

When clearing bottleneck situations resulting from system saturation (saturation of the paging area), the CLASS-WEIGHT parameter determines the urgency the individual job classes will have.

Data management

Management of volumes and access paths

The device management facility consists essentially of the following components:

- **Resource allocation**
  This function controls and monitors the logical statuses (occupied/available) of the resources (devices, volumes) requested by the user.
- **Resource reservation**
  This function processes requests for resources. These requests are formulated by the user with the aid of /SECURE-RESOURCE-ALLOCATION.
- **Reconfiguration**
  This function permits the operator to add or remove hardware units and their virtual connections on the command level.
- **Volume monitoring**
  This function monitors the accessibility of volumes, particularly during mounting procedures.

In the case of resource allocation the following points should be noted with reference to system overhead:

Each time a volume is occupied, information to this effect is stored in its standard volume label (SVL). In the case of private volumes, system overhead varies, depending on the type of allocation (task-exclusive/task-shareable) and the time the allocation is made (ASSIGN-TIME=*USER/*OPERATOR).

- When the time the allocation is made is determined by the user and the allocation is task-exclusive using the command

  ```
  /SET-DISK-PARAMETER VOLUME=vsn,ASSIGN-TIME=*USER, USER-ALLOCATION=*EXCLUSIVE
  ```

  the SVL on the volume is updated at the beginning and end of each file operation involving catalog access (/COPY-FILE, /DELETE-FILE, /CREATE-FILE, /MODIFY-FILE-ATTRIBUTES, OPEN, CLOSE).
- When the time the allocation is made is determined by the user and the allocation is task-shareable using the command

  ```
  /SET-DISK-PARAMETER VOLUME=vsn,ASSIGN-TIME=*USER, USER-ALLOCATION=*SHARE
  ```

  the SVL is read when the first user attempts access, and written back when the volume is released by the last user. In the meantime, the SVL on the volume is not updated, but internal table management activities take place whenever there is a file operation involving catalog access.
- When the time the allocation is made is determined by the system (activation interrupt) SVL updating is performed once for each session, regardless of the type of allocation.

Recommendations

- The system should determine the time of allocation.
  This is set by means of folloiing command:

  ```
  /SET-DISK-PARAMETER VOLUME=vsn,ASSIGN-TIME= *OPERATOR,USER-ALLOCATION=*EXCLUSIVE
  /*SHARE
  ```

  Release of the volume (special case) can be performed by means of:

  ```
  /SET-DISK-PARAMETER VOLUME=vsn,ASSIGN-TIME=*USER
  ```

Measurement tool openSM2

This chapter describes the following aspects of measurement tool openSM2:

- Parameters for measurement and analysis
- Reports
- Capacity requirement of openSM2
- Performance analysis with COSMOS

Parameters for measurement and analysis

It is advisable to have SM2 running in each session. For workload logging over longer periods of time SM2 can record the results into a file for later evaluation. The evaluation of such files can be done via SM2R1 in the BS2000 or openSM2 Manager on the Management Unit. With long-term measurements, the reliability of the measured data (which is dependent on the number of samples taken) can be ensured if one sample is taken per second:

- sampling period (SAMPLING-PERIOD): 1000 milliseconds
- monitoring cycle (OFFLINE-PERIOD): 5 minutes
- analysis subinterval: 1 hour
- Monitoring period: from System Ready to Shutdown

It is not necessary to evaluate each and every session. However, at least random samples in the form of daily evaluations must be made quite frequently.
Analyses over longer periods of time are the best means of evaluating trends and providing a sound basis for backing up investment decisions. The number of annual reports should be kept to a minimum.

When unusual circumstances arise, e.g. in the event of complaints, queries or changes in the configuration, it is not sufficient to analyze solely the one day in question; analysis of a second day is additionally required for purposes of comparison. If the graphs for the two analyses are placed next to one another and the figures therein are compared, relevant statements can be made much faster by noting the differences. It should be borne in mind that corresponding graphs in the two analyses might be printed using different scales, which could make the before-and-after comparison somewhat more complicated.

SM2 is also used in the support of system measurements. To this end it supplies several reports which go beyond the interests of routine system monitoring based on standard measurement criteria.

Reports

For routine monitoring purposes, the following reports of the BS2000 SM2R1 and openSM2 manager analysis routine have proved to be especially helpful:

Report group	Report	Meaning
CATEGORY	CPUTIME[%]	CPU utilization (TU+TPR) for category [1]
CATEGORY	IOs[/s]	DMS I/O rates for category [1]
	IOSW/HWTIME[ms]	Hardware/software service time for category [1]
CATEGORY	Active/InactiveTasks	Number of tasks for category
CHANNEL	Data[kB/s]	Channel throughput [2]
CPU-TOTAL	REAL	CPU utilization (genuine value, also with VM2000)
	SVC	SVC calls
DISK	IOs	DIsk IO rates
IO	TotalIOs[/s]	Global I/O rates
	Page[/s]	Paging I/O rates
MEMORY	PGAREA	Paging area utilization
	MEM	Size of the System Working Set
	WORKINGSET	Main memory utilization
PERIODIC-TASK		Utilization data for job name:
	JobCPUTIME[%]	CPU share
	JobIOs[/s]	I/O rates
		Utilization data for TSN:
	TSNCPUTime[%]	CPU share
	TSNIOs[/s]	I/O rates
		Utilization data for user ID:
	UIDCPUTime[%]	CPU share
	UIDIOs[/s]	I/O rates
VM2000	UTILIZATION	Load shares of VM2000 guest systems [3] VM2000 hypervisor share [4]

[1] with monitoring program SYSTEM

[2]with monitoring program CHANNEL-IO

[3]with monitoring program VM (the values for all guest systems are supplied only on the monitor system)

[4]with monitoring program VM (only on /390 servers)

The sections below describe the various reports in detail.

Reports of the report group BCAM-CONNECTION

These reports provide useful results on the share of response times caused in the server. The transmission times of the messages and, if applicable, the status in the network are not recorded. The reports are therefore only suitable for monitoring the load situation in the host system. Information on the response times experienced by the user (see section "Formulating a performance expectation") cannot be provided.

Report "CPUTIME" of the report group CATEGORY

This report and the report IOs of the report group CATEGORY described below are only obtained when the SM2 monitoring program SYSTEM is activated. The report is used to find which category is the source of an excessively high CPU workload or to monitor the distribution of CPU capacity over the different categories. The latter is also of assistance in setting PRIOR or PCS.

Report "IOs" of the report group CATEGORY

This report is used to monitor the number of I/O operations for the categories. It is of little relevance for system control, since the number of I/O operations of the category is not taken into account as part of category control. This report is more of a starting point for solving performance problems. For a more in-depth analysis of bottlenecks, the resource requirements (CPU time, number of I/O operations) can be recorded up to task level by using the TASK monitoring program.

Report group CHANNEL

The utilization of channels cannot be directly determined. In section “Performance of one 16-Gbit channel” you will find standard values for the throughputs that can be achieved in practice. It is recommended that only around 60% of these throughputs are exploited:

- Very high channel throughputs do not necessarily show high performance, but can indicate a bottleneck. The highest channel throughputs are normally achieved with a large number of tasks, each of which can only achieve a low proportion of the total throughput.
- By contrast, a small number of tasks that are not influenced and thus operating at high performance do not generally exploit the maximum possible channel throughput.
- Excessively high channel loads may therefore indicate that too many devices are attached to the same channel. However, the use of page chaining also makes the channel load too high. For this reason this function should only be used if it brings about distinct advantages. This is always the case if all the blocks transferred in one I/O operation are required and there is no need for further I/O operations.

> When comparing channel throughputs, always take into account the fact that FC channels achieve their specified throughput in each transmission direction. Higher throughputs are thus possible for mixed reading and writing than for pure reading or writing.

Channel throughput with VM2000 operation:

Recording of the input/output rate is guest system specific. Therefore, the throughputs output relate to a guest system.

Channel throughput for x86 servers

The information in the description of the CHANNEL-IO monitoring program in the “openSM2 (BS2000) [18 (Related publications)]” manual must be taken into account.

Reports of the report group "CPU-Total"

The “REAL" report shows the CPU utilization monitored (also in VM2000 operation). The “NORM” report should not be used, in particular not in VM2000 operation!

Assessment of the CPU workload varies according to the type of application and type of load. A workload of 100% is feasible and desirable for pure batch applications. Balanced utilization of the remaining resources is to be strived for.

In the case of mixed loading with TP, interactive and batch applications, the main application should comprise no more than 70% of the total CPU workload for the system (in the case of multiprocessor systems, depending on the number of CPUs, up to 90% is acceptable for the main application). An ideal background load triggers neither I/O operations nor paging; being extremely compute-intensive it is well-suited to making use of surplus capacity. This combination of features makes it possible to take full advantage of CPU capacity; in practice, however, it seldom occurs. Any one I/O operation or paging transfer can interfere with the TP and interactive applications. I/O-intensive loads and programs with extensive memory requirements should not be loaded in the background. The background load must be made clearly subordinate to the main application by assigning appropriate priorities or PCS settings. Ideally, the files of the main application should reside on other disk drives (and at least on other volumes) and be accessible via other paths.

The SIH share depends, among other things, on the number of I/O operations of all kinds and the intensity of task communication. The SIH portion should not exceed 20% (/390 servers) or 10% (x86 servers). A high percentage of TPR operations is caused by SVC calls, which perform resource-intensive system services. The workload can be regarded as normal if the following formula applies:

TU + TPR > 3 * SIH (/390 server) or TU + TPR > 7 * SIH (x86 server)

Now and again there are instances where the load violates the above rules without actually impairing performance behavior. As long as the response times are good there is no cause for intervention in these cases.

Generally the applications run on an x86 server in /390 mode. As programs which run in this mode have a higher CPU requirement than on /390 servers, a high share in the processor state TU is normal (some 60%, standardized as 100% utilization). If a higher standardized TU share is shown, this means the performance of the server with x86 architecture is below the nominal value (the official RPF value). However, of the (standardized) TU share is lower, this means the performance is above the nominal value.

"HW- and SWServiceTime" reports of the DISK report group

The “HWServiceTime” and "SWServiceTime" reports are available only if the SAMPLING-DEVICE monitoring program with recording of hardware and software service times is activated. If software service times occur which are more than 10% over the hardware service times, I/O operations must sometimes wait in front of an occupied volume.
The SM2 monitoring program DISK-FILE and the SM2 online report DISK-FILE can then be used to ascertain the files affected.

> If the application works with asynchronous inputs/outputs, an increased number of long software wait times can occur without a tuning measure being required or possible.

Report "IO" of the report group IO

This report should be assessed in conjunction with the report “Utilization real” of the report group CPU. Certain loads are both compute-intensive and have a high number of I/O operations. Consequently, the most important criterion is that the channels and volumes are not overloaded. The standard values for the maximum number of DMS I/O operations per second are determined in accordance with the rule that (depending on the speed of the system) only 15 to 25% (/390 servers) or 8 to 12% (x86 servers) of the CPU capacity in processor states SIH+TPR should be used for processing the I/O operations. In certain situations it is feasible to exceed this limit.

See also section “Standard values for BS2000 servers”.

Report "IO" of the report group IO and the measurement Page[/s]

Paging enables the efficient utilization of a relatively small main memory as compared to the number of virtual address space requests.

Thanks to the availability of main memory at a reasonable cost, there is generally sufficient main memory to ensure that dynamic paging does not represent a problem. What follows below therefore only needs to be taken into account in special cases.

Paging is the result of a lack of memory. It burdens both the peripherals and processor. For this reason the paging rates (page reads per second) shown in section "Standard values for BS2000 servers" should not be exceeded. Violating the maximum recommended values is only permitted if the response times are acceptable **and** if all other workloads (CPU, channels, volumes) are low.

In the event of bottlenecks on the channel or paging volumes, it may be the case that the specified paging rates have not been reached, yet the response times are still unsatisfactory. This is caused by excessively long waiting times for PAGE READ. Poor configuration of the paging peripherals has serious detrimental consequences and must be avoided at all costs.

TP mode is very sensitive as far as paging is concerned - and even more so, the higher the number of important operations performed by relatively few central tasks. Such central tasks must not be burdened with paging I/O operations, since the resulting delays would be passed on to many other tasks. Paging tends to increase at breakneck speed. For this reason it is not advisable to exploit the main memory of a system to its full extent (see also the notes on the “Main memory utilization” report of the report group WORKING-SET as well as the sections "Monitoring main memory utilization" (Characteristic main memory values) and "Paging").

Report "PGAREA" of the report group MEMORY

The maximum occupancy level of the paging area should be 50%.

If the average occupancy level of the paging area is not greater than 50%, peak paging loads with acceptable paging I/O times can be intercepted. Sufficiently large contiguous areas are then available for high-performance transfer of modified memory pages.

A considerable increase in response times occurs at the latest when peak paging loads occur when the average occupancy level is 70%.

Reports "MEM" and "WORKINGSET" of the report group MEMORY

The value “Available pages (NPP)” specifies how much real main memory is still available for paging.

Paging operations start to occur if the value “Number of page frames for system global set” (SWS) reaches approx. 80 - 90% of NPP.

If the SWS value increases to above 90% of NPP, an expansion of the main memory should be taken into account. In VM2000 operation it is often (at least as a transitional measure) sufficient to distribute main memory more efficiently between the guest systems. With these high SWS values you are warned against adding new applications or changing the version of frequently used software. This could result in a significant rise in the paging rate.

Reports of the PERIODIC-TASK report groups

These reports are very useful for analyzing possible reasons for resource bottlenecks (in particular CPU and IO). Depending on the application it can make more sense to perform the analysis on the basis of the user ID, the job name or the TSN.

Reports of the VM2000 report group

/390 servers

Measurements on the monitoring system can be used together with the "UTILIZATION" report to analyze the CPU utilization of all the guest systems. This report also provides values for Hypervisor usage (HypervisorActive[%]).

x86 servers

The report "VM2000 utilization" shows the CPU utilization of all BS2000 guest systems, however relative to all BS2000 CPU pools. The additional hypervisor overhead cannot be measured within BS2000.

Under VM2000 (depending on the number and load of the guest systems), the utilization of the BS2000 CPU can be 5-15% higher than in native operation.

Capacity requirement of openSM2

The capacitiy requirement of openSM2 consists of:

- Capacity requirement of the SM2 subsystem in BS2000
- Capacity requirement of the openSM2 Manager on the Management Unit

Capacity requirement of the SM2 subsystem in BS2000

To record the monitored data SM2 requires the resources CPU and disk storage. The requirement depends on:

- the type of monitoring programs and their parameters
- the hardware configuration
- the load profile (principally the number of tasks and the I/O intensity)

In all the resource utilization by SM2 is relatively low. Occasionally a check should be made to see whether a restriction of the monitoring programs is necessary or possible. However, a restriction of the monitoring programs is detrimental to the quality of performance monitoring.

Monitored data

The resource requirement of SM2 was measured in a laboratory test under a typical OLTP load which utilizes approx. 70% of the server capacity.

The following CPU requirement for SM2 in absolute percentage values was measured:

- 0.5 - 1% for basic monitoring (standard settings and SYSTEM monitoring programs)
- Additional 1,5% for recording response times via BCAM-CONN and the use of monitoring programs PER-TASK and TASK
- Additional 1,5% when decreasing the sampling period to 400 ms

These values were monitored using the standard sampling period of 800 ms. When the sampling period is changed, the CPU requirement of SM2 changes linearly with the share of periodically recorded data. For example, in basic monitoring with recording of response times via BCAM-CONN, the CPU requirement of SM2 drops from 1.5% to just under 1% when the sampling period is changed to 2000 ms.

Disk storage space is required to store the monitored data. It is not possible to specify standard values in absolute measures as the requirement is too heavily dependent on the relevant hardware and software load and the configuration of the monitoring programs. However, it must be borne in mind that in particular the PERIODIC-TASK monitoring program makes the output file increase by 50% or more compared to basic monitoring.

Capacity requirement of the openSM2 Manager on the Management Unit

The openSM2 manager is an add-on software module in SE manager (on the management unit of an SE server) and enables central monitoring of the systems on the /390 and x86 server units, the application units (x86), storage systems, and all devices with SNMP capability. Systems outside this SE server (e.g. storage systems, devices with SNMP capability, Windows systems) can also be monitored.

The openSM2 manager includes

- the **agents** for recording the measured data, and
- the **web-based user interface** for performance monitoring.

The agents periodically record measured values for the monitored systems and save them in one or two databases

1. in an online database with limited size for current measured data, and
2. in an optional database for long-term archiving, whose size is only limited by the storage medium (an nfs drive).

The openSM2 manager can be configured and administrated using a browser. Measured data for the monitored systems or system groups can be evaluated, presented on the screen, or exported to csv files.

The openSM2 manager also enables systems to be monitored with user-defined rules and automatic evaluations to be sent by e-mail.

The procedure for calling up and using the openSM2 manager, along with a description of the functions, can be found in the “openSM2 (BS2000) [18 (Related publications)]” manual. The openSM2 manager provides comprehensive online help.

When used on the management unit of an SE server, the openSM2 manager uses a variable amount of resources (CPU, disk space, memory, network). The actual resource requirement depends on numerous factors.

Resource requirement for data recording

For data recording, the resource requirement depends on

- the length of the measuring interval
- the number and type of monitored systems (BS2000, Linux, Windows, X2000, VMware vSphere, storage systems, devices with SNMP capability)
- the number of measuring objects (for each measuring object type, e.g. CPUs, disks, channels, tasks, etc.) on the monitored systems
- the type and number of active SM2 monitoring programs in BS2000
- recording of process data and data for partitions. When recording process data, very high data volumes occur.

If process data is to be recorded, we recommend combining the relevant processes into meaningful process groups, and then monitoring only the process groups and not all individual processes.

Resource requirements for presentation of measured data

The resource requirement for presentation of measured data using the web application depends on

- the number of logged in users

- the number of open reports per user
- the number of measured variables or measuring objects per report
- for reports with measuring objects, the type of hit list and the number of measuring objects in the database
- for system group reports, the number of systems in the group
- the length of the time axis and the number of data points
- the type of data source (online or archive database).

The resource requirement for openSM2 is also influenced by:

- the number and scope of automatic evaluations performed
- the archiving scope
- the number of systems monitored
- the number and complexity of active rules

For online presentation of the measured data using the web application, it is primarily additional CPU resources that are needed. As the CPU use depends on the number of users and the reports displayed, it is important to ensure that logged in users close their online report views as soon as they no longer require them when they return to the SE manager or log out.

Performance analysis with COSMOS

COSMOS is an event-driven monitor for recording detailed monitored data for the specific performance diagnosis of BS2000 systems. COSMOS is an SM2 monitoring program integrated into openSM2 and as such can be controlled using SM2 commands.

COSMOS records the entire course of the system events over time. Approx. 90 different types of event can be registered, e.g. start and end of an I/O operation, system calls (SVCs), and generation and termination of tasks. Each event is identified by a 4-character name. To record the events, COSMOS interfaces, so-called “hooks”, are implemented at various places in the system. Depending on the parameters selected the hook is “opened”, i.e. the data capture code is activated each time the relevant event occurs and a record is written. When COSMOS is terminated, all hooks open at this point are closed.

The data can be collected for all tasks or for tasks selected according to specific criteria (user ID, category, job name or TSN).

The monitored data recorded is written to output files (to tape or disk). Special analysis programs are available for analyzing the COSMOS output files. Generally specialist knowledge is required to evaluate and analyze COMOS measurements.

Servers

This chapter describes the aspects of BS2000 servers which are relevant to performance:

- Recommendations for CPU utilization
- Scaling behavior of multiprocessor systems
- Standard values for BS2000 servers
- Controlling buffer sizes of the x86 hardware (CISC)
- Migration

Recommendations for CPU utilization

Uniprocessor systems

The following standard values apply for online applications and batch applications

- In the case of response-time-critical online applications, with uniprocessors a CPU workload of 70% should not be exceeded for the main application, because as the workload increases, the wait time in the queue ahead of the CPU rises disproportionately (see also section "Examining system performance with openSM2").
- In the case of pure batch applications a CPU utilization of up to 100% is justifiable if the processing duration of the individual batch tasks in the system is not significant.

Multiprocessor systems

With multiprocessor systems the probability of finding a free CPU increases as more CPUs become available. Therefore higher workloads can also be tolerated. With response-time-critical applications the following CPU workloads should not be exceeded for the main application:

- 75% with 2 CPUs
- 80% with 4 CPUs
- 85% with 6 CPUs
- 90% with 8 or more CPUs

Scaling behavior of multiprocessor systems

The higher the number of CPUs, the more sharply the overhead for the main memory synchronization rises, combined with a reduction in the cache hit rate.

The CPU time will increase as follows:

- with 2 CPUs by 5 to 10%
- with 4 CPUs by 10 to 15%
- with 6 CPUs by 15 to 20%
- with 8 CPUs by 20 to 30%
- with 10 CPUs by 25 to 35%
- with 12 CPUs by 30 to 40%
- with 16 CPUs by 35 to 45%

Provided the load is distributed over tasks which run in parallel and are, if possible, independent of each other, the transaction rate can be considerably increased. The following improvements in the throughput rate can be achieved on /390 servers in comparison to a uniprocessor:

- with 2 CPUs by a factor of 1.9 to 2.1
- with 4 CPUs by a factor of 3.7 to 4.0
- with 6 CPUs by a factor of 5.1 to 5.9
- with 8 CPUs by a factor of 5.7 to 6.7
- with 10 CPUs by a factor of 7.0 to 7.9
- with 12 CPUs by a factor of 7.8 to 9.0
- with 16 CPUs by a factor of 9.5 to 11.3

In order to satisfy the increased resource requirements caused by raising the number of tasks, extension of the main memory and an increase in the number of disk drives are absolutely essential.

Reference values for BS2000 servers

The tables below show some performance data and reference values for BS2000 servers. The recommendations apply for typical TP operation with a normal background load.

- Relative Performance Factor RPF and number of CPUs.
- Typical main memory configuration.
  The actual main memory requirement depends on the load with which the server is operated. TP and dialog loads normally have a significantly greater main memory requirement than batch loads.

  On x86 servers, the main memory is distributed between BS2000 and Dom0/X2000 (I/O processor). A recommendation for the configuration and distribution of main memory is shown in the corresponding tables. Information on when more main memory than recommended there should be used is provided in section “Special features for x86 servers”.
- Maximum recommended number of DMS I/O operations.
  An IT system is used economically when it is executing a balanced mixture of programs requiring an intensive use of calculation functions and programs requiring an intensive use of I/O functions. The recommended maximum number of DMS I/O operations per second is not an upper limit determined by technical requirements, but guarantees that only a certain proportion of the CPU capacity is used to perform I/O operations. It can be exceeded without problems, provided bottlenecks in the peripherals are avoided.
- Maximum recommended number of paging I/O operations.
  The maximum recommended paging rates specified for transaction mode or interactive mode ensure that only a certain proportion of the CPU capacity is used for paging. To avoid delays through paging (especially in transaction processing mode), you should try to aim for less than the stated values in view of the main memory available.
- Maximum recommended number of SVCs in TU.
  When the maximum number of SVC calls per second is reached, the effort required for the interrupt analysis and termination handling of the system routine (i.e. not the resources used for the system routine itself) amount to approx. 5% of CPU time.

Reference values for /390 servers

Server	Variant	RPF	Number CPUs	Typical main memory size in GB	#max[1] DMS inputs/ outputs	#max paging inputs/ outputs TP mode	#max paging inputs/ outputs dialog mode	#max SVCs in TU
SU710	-10A	360	1	4	6100	100	300	18000
	-10B	430	1	6	7400	100	300	21500
	-10C	520	1	6	8600	100	300	26000
	-10D	630	1	8	10000	100	300	31000
	-20A	700	2	8	11000	100	300	35000
	-20B	840	2	8	13000	100	300	43000
	-20C	1010	2	12	16000	100	300	51000
	-20D	1220	2	12	19000	100	300	61500
	-30	1800	3	24	25000	100	300	90000
	-40	2350	4	24	30000	100	300	118000
	-50	2800	5	32	36000	100	300	143000
	-60	3200	6	32	40000	100	300	162000
	-70	3600	7	48	43000	100	300	181500
	-100	4400	10	48	50000	100	300	223000
	-120	4900	12	48	55000	100	300	250000
	-140	5400	14	64	58000	100	300	275000
	-150	5700	15	64	60500	100	300	288500
SU730	-10A	360	1	4	6100	100	300	18000
	-10B	430	1	6	7400	100	300	21500
	-10C	520	1	6	8600	100	300	26000
	-10D	630	1	8	10000	100	300	31000
	-20A	700	2	8	11000	100	300	35000
	-20B	840	2	8	13000	100	300	43000
	-20C	1010	2	12	16000	100	300	51000
	-20D	1220	2	12	19000	100	300	61500
	-30	1800	3	24	25000	100	300	90000

	-40	2350	4	24	30000	100	300	118000
	-50	2800	5	32	36000	100	300	143000
	-60	3200	6	32	40000	100	300	162000
	-70	3600	7	48	43000	100	300	181500
	-100	4400	10	48	50000	100	300	223000
	-120	4900	12	48	55000	100	300	250000
	-140	5400	14	64	58000	100	300	275000
	-150	5700	15	64	60500	100	300	288500
SU740	- 10A	400	1	4	6.800	100	300	20.000
	- 10B	480	1	6	8.300	100	300	24.000
	- 10C	570	1	6	9.500	100	300	28.500
	- 10D	670	1	8	11.000	100	300	33.500
	- 20A	760	2	8	12.000	100	300	38.000
	- 20B	920	2	8	15.000	100	300	46.000
	- 20C	1.100	2	12	18.000	100	300	55.000
	- 20D	1.300	2	12	21.000	100	300	65.000
	- 30	1.900	3	24	27.000	100	300	95.000
	- 40	2.500	4	24	33.000	100	300	125.000
	- 50	3.100	5	32	40.000	100	300	155.000
	- 60	3.700	6	32	47.000	100	300	185.000
	- 70	4.300	7	48	52.000	100	300	215.000
	- 80	4.500	8	48	55.000	100	300	225.000
	-100	5.000	10	48	57.000	100	300	250.000
	-120	5.700	12	48	64.000	100	300	285.000
	-140	6.400	14	64	67.000	100	300	320.000
	-150	6.750	15	64 GB	72.000	100	300	337.500

[1]#max = maximum recommended number (per second)

Reference values for x86 servers

Server	Variant	RPF	Number CPUs	Typical main memory size in GB [1]	#max[2] DMS inputs/ outputs	#max Paging inputs/ outputs TP-mode	#max Paging inputs/ outputs dialog mode	#max SVCs in TU
SU310	-10R	80	1	128	1.900	70	250	4.000
	-10	240	1	128	4.500	100	300	12.000
	-20	430	2	128	7.300	100	300	21.000
SU320	-120	1.600	12	96	22.400	100	300	80.000
SU330 & SU330B	-10A	20	1	128	1.000	9	28	1.000
	-10B	42	1	128	1.400	13	40	2.100
	-10C	80	1	128	1.900	70	250	4.000
	-10D	130	1	128	2.900	70	250	6.500
	-10E	200	1	128	4.100	100	300	10.000
	-10F	300	1	128	5.900	100	300	15.000
	-20	520	2	128	8.600	100	300	26.000
	-40	950	4	256	15.500	100	300	47.500
	-80	1.600	8	256	26.500	100	300	80.000
	-120	2.000	12	256	30.700	100	300	100.000
	-160	2.500	16	256	34.500	100	300	125.000
SU340	-10A	20	1	128	1.000	9	28	1.000
	-10B	42	1	128	1.400	13	40	2.100
	-10C	90	1	128	2.100	70	250	4.500
	-10D	140	1	128	3.100	70	250	7.000
	-10E	220	1	128	4.500	100	300	11.000
	-10F	330	1	128	6.500	100	300	16.500
	-20	550	2	128	9.100	100	300	27.500
	-40	1.000	4	128	16.000	100	300	50.000
	-80	1.700	8	128	27.000	100	300	85.000
	-120	2.100	12	128	31.000	100	300	105.000

	-160	2.600	16	128	34.000	100	300	130.000

Table 3: Reference values for x86 servers

[1]To determine the memory available for BS2000 guest systems, the share for Dom0/X2000 must be deducted. The X2000 V6.4 Dom0 memory requirement defaults to 25% with a maximum of 16GB, whereas under X2000 V6.5 the Dom0 size is always 32GB. It must also be borne in mind that approx. 40% (depending on the system parameter JTABSMEM, see "Controlling buffer sizes of the x86 hardware (CISC)") of the main memory available for BS2000 is used as a JIT buffer and is therefore not directly available to the operating system and applications, see section "Special features for x86 servers".

Example: On an SU330-40F (main memory: 128 GB) BS2000 and the BS2000 applications can use around 58 GB of main memory in accordance with the following calculation:

128 GB - 32 GB (Dom0/X2000) = 98 GB

98 GB * 0,6 (JIT) = 57,6 GB

[2]#max = maximum recommended number (per second)

Controlling buffer sizes of the x86 hardware (CISC)

The following system parameters control main memory use on x86 servers for the Just-In-Time compilation of /390 code by the CISC firmware:

JTABSMEM

Parameter name	Value range	STD value
JTABSMEM	0...1,000,000	0

This parameter defines the maximum JIT buffer memory size in MB. The memory is requested as resident and must therefore match the memory available on the machine.

The parameter's default value is 0. In this case the CISC firmware calculates the maximum memory size using the formula: MEMORY-SIZE * 40 %. The computed value of JTABSMEM is logged with the message HJT0039 on the console.

The upper limit for the resident memory used by the CISC firmware can also be permanently defined by specifying an explicit value from 1 - 1,000,000.

The value set can also be modified in ongoing operation using /MODIFY-SYSTEM-PARAMETERS, see below. However, you are urgently recommended to work with the preset default values and not to enter an explicit value for JTABSMEM as the use of the STD value has proved very effective in practice and also offers the automatic feature that the size of the JIT buffer is adjusted dynamically in relation to the main memory actually available (e.g. in the context of memory reconfiguration).

Modifying the setting

The following points must be borne in mind when the value set is modified:

- The value set must match the details of the parameter value for BIG-PAGE-QUOTA, see the section on "Parameter service for memory management (MEMORY)" in the “Introduction to System Administration[10 (Related publications)]”. BIG-PAGE-QUOTA defines the maximum number of resident BIG PAGES in BS2000 which are stored by Memory Management and can be used by any BS2000 instances. Currently the CISC firmware is the only instance which uses BIG PAGES. For this reason the STD values for the two parameters must be adjusted to each other:

Parameter	Meaning
JTABSMEM=0	The CISC firmware utilizes up to 40% of the MEM-SIZE
BIG-PAGE-QUOTA=40	BS2000 Memory Management reserves 40% of the MEM-SIZE for BIG PAGES

- If a value which is not equal to 0 is specified explicitly for JTABSMEM, you must always ensure that neither too many nor too few BIG PAGES are stored by BS2000 Memory Management:
  - In the first case too much reserved memory space would remain unused.
  - In the second case the CISC firmware would request any other resident memory above the available BIG PAGES up to the upper limit defined by means of JTABSMEM. As the address translation for this is significantly less powerful than for the BIG PAGES, this would lead to poorer performance for the TU programs.

- A change to the preset values for the above-mentioned parameters might possibly make sense if bottlenecks in the use of the KIT memory do actually occur when the system executes. When the memory space already used in the JIT buffer approaches the limit value, the message HJT0035 is output with a corresponding JIT memory saturation level:
  - When level 1 is reached it is advisable to increase the threshold value.
  - When level 2 is reached you are urgently recommended to increase the threshold value, otherwise considerable losses of performance must be expected. Any increase should, however, always comply with the total main memory available and the memory requirement of the remaining BS2000. In such cases Customer Support should assess whether the main memory available is sufficient for the applications running on the system.
- Regarding memory reconfigurations it should be considered:
  - While the size of the JIT biffer is dynamically adjusted to the actually available memory, occupied memory is only released upon program end.
  - It may therefore be desired to prepare a VM for a later /REDUCE-VM-MEMORY by setting a "too small" JIT buffer of 40% of the permantly remaining memory.

JTSTDMEM

Parameter name	Value range	STD value
JTSTDMEM	1...65535	16

This parameter, JTSTDMEM, determines how much JIT buffer the system can make available to each task within the global JIT buffer, provided that no other task-specific settings have been made by the /MODIFY-DBL-DEFAULTS command.

The default size of the task-specific JIT buffer is 16 MB. As the CISC firmware only requests as much memory as is necessary to run the current /390 programs, we recommend that you do not change the STD value. When the JTSTDMEM value is reached the message HJT0032 is given, the task-specific JIT buffer is reset and rebuilt. As the associated reduction in performance is normally low, the message is only issued in the CONSLOG file.

Enlarging the JIT buffer is only worthwhile in rare cases (e.g. a task with particularly large working set requirements).

To increase the size of the JIT buffer the new size must be specified with the following command (nn = number of MB):

```
/MODIFY-DBL-DEFAULTS SCOPE=*CMD-CALLS(CISC-COMPILATION=*YES(WORKSPACE=nn))
```

The change takes effect for all DBL calls (/START-, /LOAD-(EXECUTABLE-)PROGRAM, BIND). The command must be inserted into all Enter jobs or procedures if programs that need a larger JIT buffer are loaded there.

JTMAXMEM

Parameter name	Value range	STD value
JTMAXMEM	1...65535	128

The parameter JTMAXMEM fixes the maximum value which can be set as task-specific using the command /MODIFY-DBL-DEFAULTS.

JTSHMEM

Parameter name	Value range	STD value
JTSHMEM	0...256	64

Specifies how much storage space JITSYS is to use for storing shared compiled code (in MB).

Shared compiled code is created when emulating subsystems loaded in class 4 memory. If no such subsystems exist or these are not to be compiled as “shared”, the creation of shared combined code can be prevented with JTSHMEM = 0. The JTSHMEM value should be adjusted to the total amount of class 3 memory available.

The storage size specified with JTSHMEM is allocated immediately when JITSYS is initialized. Modification of the value during ongoing operation will, until further notice is given, not be supported. The value can only be specified in steps of 4; other values are rounded up to the next multiple of 4.

The default value for JTSHMEM should be changed together with the BIG-PAGE-SHRSIZE parameter in the MEMORY parameter set of the startup parameter service

The connection to the SHXSIZE parameter of the setup parameter service is indirect:

> When the value selected for SHXSIZE is very high and the corresponding number of programs is actually loaded as “shared”, it also makes sense to increase JTSTDMEM.

Migration

Irrespective of the source and target HSIs, there are some general aspects which should be borne in mind when migrating a server:

Number of BS2000 CPUs of a server

When the multiprocessor level of a server changes when migration takes place, this can also result in a change to the performance behavior of the software running on the server. In such a case check the relevant parameters and settings which control the performance behavior of the software.
In particular check the settings for OLTP mode, e.g. the number of UTM tasks used, the number of TAC classes, the number of tasks per TAC class, the number of DBHs, and the buffer sizes of the database system.
Adjust the parameters if necessary.

Detailed information on this is provided in the manuals on openUTM, in the "SESAM/SQL Server Performance" manual [27 (Related publications)], and in the ORACLE manuals.

Main memory size

When migration takes place, check whether the main memory needs to be increased to enhance the performance:

- to avoid unnecessary paging I/Os
- to expand the buffer for OLTP mode
- to expand the caches for the software product DAB

Operating mode: Native or under VM2000

Take note of the following points if, in the course of migration, native mode is to be switched to virtual mode under VM2000:

- Virtual mode causes an overhead which must be allowed for accordingly when calculating the performance of the new server. The amount of overhead depends on the configuration, see "Virtual machines".
- Take note of a few constraints when configuring VM2000 to avoid unnecessary overhead. See section "Recommendations for an optimal configuration".

Initial status

The initial status in ongoing operation is the basis for these considerations. For this purpose the most important application cases, such as OLTP mode or critical batch applications, must be monitored on the original server using openSM2. The results are then evaluated and which server will be most suitable as a target server is forecast.

The following must always be observed in the case of server migration:

- Current CPU workload of the original server (high / low)
  In OLTP mode the server's CPUs should have a workload of less than 70%.In batch mode a CPU workload of up to 100% is generally maintainable.
- Planning the main memory requirement when changing the BS2000 versionThe extra capacity required depends on the version. On average 1 - 2% per version must be estimated.
- Taking into account the system's own growth requirements
  Example: 5% per year for 3 years, i.e. around 15% growth requirements.
- Influence of adding new applications or relocating existing ones
  If applications are omitted from the new server or new ones are added, this must be taken into account.

> If required, it is recommended that the offering of the BS2000 Optimization Service and the BS2000 demo center should be used:
>
> - Performance consulting when migrating
> - Test measurements for customer applications on the target hardware
>
> Please contact your sales representative for information.

Migration from /390 servers to x86 servers

The following must be observed:

- Uniprocessor performance
  When migration is to take place from a /390 server (uniprocessor) to an x86 server (multiprocessor), the total RPF value can remain unchanged, but the uniprocessor performance will be lower in most cases. This must, for example, be taken into account for batch applications which cannot be operated in parallel.
- CPU performance: TU portion
  The CISC firmware ensures that customer applications execute unchanged on x86 servers. The RPF value of the server with x86 architecture applies for a TU portion of 40 to 50% of the entire CPU requirement. When the TU portion is 50% or higher, an additional performance requirement of approx. 10% should be provided for the new x86 server.
- Peripherals: networking
  The integrated LAN controllers of the server with x86 architecture are available. An HNC is not required, nor can it be used.
- Memory requirements
  As a rule the standard main memory configuration of the x86 servers is sufficient. For the exceptions see section "Special features for x86 servers".
- Object format
  System exit routines must be cleaned up and recompiled. As all other privileged routines, system exit routines are required to run directly on the CPU (native), in this case as x86-Code on the x86-CPU. For further information see the "Manual System Exits" [17 (Related publications)].

Example: calculating the RPF value for a target x86 server

The following assumptions apply for the /390 servers used to date:

- SE710-10D (630 RPF)
- Load: OLTP mode (UTM/SESAM)
- CPU workload 80% (50% of that TU)
- BS2000/OSD V.0B
- Growth requirements 15%

This results in the following calculation for the target x86 server:

630 RPF * 1.1 (higher performance requirement because of TU portion 50%) * 1.02 (migration from BS2000 V20.0 to V21.0) = 707 RPF

In order to keep the workload to the 65% recommended for TP mode despite the growth:

707 RPF * 1.2 (target workload 65%) * 1.15 (growth requirements) = 976 RPF

Th x86 server with the next higher RPF level is an SE340-40 server with 1,000 RPF and therefore would be suitable. The switch from a uniprocessor to a multiprocessor with four CPUs must be borne in mind here which may effect the batch runtimes.

In rare cases application-dependent special influences can reduce the CPU performance, e.g.:

- Intensive use of decimal, floating point or EX commands
- Self-modifying code

- Mixed code and data on the same memory page
- Alignment violations
- Very frequent SVC calls

Peripherals

This chapter contains measurements and recommendations for storage systems and network connections:

- Recommendations for high-performance operation of storage systems
- Measurements for storage systems
- Measurements for LAN connections on SE servers
- Net-Storage behavior

Recommendations for high-performance operation of storage systems

This section describes the following performance aspects relating to storage systems:

- Recommendations for high-performance IO access in applications
- Size and configuration of storage systems
- FC connection
- PAV (Parallel Access Volume) on /390 servers

Recommendations for high-performance IO access in applications

The maximum possible throughputs and IO rates depend greatly on the block size used for accessing the data. Use of an appropriate block size provides the basis for efficient IO accesses of an application, i.e.:

- In the case of transaction-oriented loads with high requirements regarding the IO rate, small block sizes should be preferred. However, the IOs should be large enough to ensure that requested data areas can be supplied in one IO. General recommendations are hardly possible here as this depends very much on the application’s IO profile.
- In the case of throughput-oriented loads, as far as possible large block sizes of at least 160 KB to 480 KB should be used

Where it makes sense and is feasible, asynchronous IOs should be performed. This avoids any additional context changes, and the IOs can also potentially profit from PAV or RSC.

Size and configuration of storage systems

In order to achieve high or optimal IO performance, the following is recommended when storage systems are used:

- As far as possible, new storage systems
- Sufficiently dimensioned cache storage
- Latest firmware status
- High-speed disks (SSDs, preferably NVMe)

To ensure optimal performance, special attention should be paid to the cache configuration, the disk population, and the RAID configuration:

Cache

The cache is essential for the IO performance. It “masks” considerably slower access times of the physical disks and must therefore be adequately dimensioned:

- In OLTP mode it should be possible to achieve a read hit rate of at least 80%, preferably higher.
- The write hit rate should always be around 100%. Write hit rates below 100% either indicate that the cache is too small or are the result of an acute overload of specific RAID groups which are not powerful enough.

FC connection

Type FC channels scale very well. In other words, n channels can permit an almost n-fold throughput. A prerequisite for this is that the BS2000 CPUs and the peripherals are sufficiently powerful.

Special features for /390 servers

With SE servers SE710 / SE730, Fibre Channels with a data rate of 16 Gbit/s are supported. The following can be ascertained from the measurements in section "Measurements for storage systems":

- The maximum channel throughputs are achieved only in the case of high parallelism, the throughput of each individual task being considerably lower than without a competitive load. An individual task alone can already claim 200-300 Mbyte/s. If, for example, three or four backup jobs are being performed over the same channel, they already constrain each other, even though their accumulated throughput is still lower than the measured maximum throughput of the channel. This also concerns the IO rates in the case of transaction-oriented loads.

This results in the following recommendations:

- For important productive loads and time-critical batch loads, the IO requirements should be known. If a number of IO-intensive tasks are operating simultaneously, a correspondingly higher number of channels should be provided.
- When multiple channels with a low utilization level are used, parallel loads can be handled better than with a few heavily used channels, even if no overload exists in the latter case.
- Parallel IOs to the same volume can be expedited by PAV. If the software IO times measured with openSM2 are significantly higher than the hardware IO times, the use of PAV should be considered (see section "PAV (Parallel Access Volume) on /390 servers"). This increases the channel utilization.

When migrating to a new channel technology, one should therefore consider:

- The 16-Gbit technology provides double throughput for parallel, throughput oriented loads. However, especially for transaction oriented loads with small IOs, the gain is much lower, so that in this case, the number of channels must not simply be reduced by half!

Special features for x86 servers

A function which is similar to PAV is provided for x86 servers in the form of the function Remote System Call (RSC) for disks. This permits up to six simultaneous I/O operations to/from a volume or device.

Serialization takes place in the storage system. RSC is used by default by the BS2000 operating system components; the user does not need to specify any special setting in the configuration to use RSC or to permit parallel I/O operations.

PAV (Parallel Access Volume) on /390 servers

This section provides the following information on the use of PAV:

- Advantages of PAV
- Effects of PAV

Advantages of PAV

The BS2000 function PAV is used to compensate for an idiosyncrasy of the /390 architecture, namely that IOs are serialized ahead of the device (which corresponds here to a logical volume). In other words, when a /390 server without PAV issues multiple IOs to the same volume, these must wait until each preceding IO has been concluded.

The consequence: increased software service times because of the wait time ahead of the device. Loads with the following are particularly prone to this:

- high parallelism
- and/or low cache hit rates.

In the case of the latter, cache hits which are actually quickly available can be significantly delayed by a preceding cache miss which has not yet been processed; the negative repercussions of the cache misses are thus multiplied.

PAV enables parallel disk access to be implemented on /390 servers. This permits the advantages of current channel and storage technologies to be utilized.

With PAV, multiple virtual devices (alias devices) can be assigned to one logical volume (base device). If a device is occupied, the IO can be started via one of the other virtual devices. The alias devices must be generated in BS2000; no intervention in the storage system is necessary.

Historically, PAV can be used as a static PAV or as a fast dynamic PAV (FastDPAV). Since the introduction of FastDPAV, there is no reason for the use of the older static PAV technique performancewise. **Therefore, FastDPAV is the only recommended method.**

Static PAV

In the case of static PAV, corresponding alias devices are generated and assigned in advance manually for heavily used devices. The decision regarding how many alias devices are assigned to which logical volumes is the responsibility of the system administrator.

Static PAV requires an analysis of the current usage of the volumes and foresighted planning with regard to the future device workload. While it can be applied targeted at r performance-critical productive data, , this is also covered by FastDPAV.

FastDPAV

Beginning with SU710, BS2000 supports FastDPAV: For a set of Volumes, connected via a set of paths with n channel paths per Volume, one FastDPAV base device is generated for each volume. Additionally a pool of FastDPAV alias devices without fixed mapping is generated for all volumes .

Handling an IO request, the BS2000 IO system (BS2IOS) uses first the base device (and ,if generated, static alias devices). If there are already active IOs for the base device and the static alias devices of this volume, BS2IOS will search for a suitable FastDPAV device and start the IO.

The combination of FastDPAV and static PAV like in this description is possible, but only mentioned for technical correctness. In practice the use of only FastDPAV, without additional static PAV devices is regarded enough and recommended.

More in-depth information on PAV can be found in the “Introduction to System Administration” [10 (Related publications)].

Effects of PAV

The use of PAV has effects on both the users' perception and on the system and hardware.

Effects from the user viewpoint

PAV primarily reduces the IO times (software I/O time):

- In the case of synchronous IOs, copy processes, batch mode, etc., the throughput then also improves correspondingly. The users perceive shorter runtimes (if the IOs had been constricted before due to parallel load).
- In OLTP mode, the throughput is specified by the user-generated work and will at most increase to a small degree. The users perceive shorter transaction times.

Basically the user perception depends on how high the proportion of the IO wait times is in the total runtime and transaction time.

Effects on the system and hardware

With PAV, more IOs can be processed in the same time, thus condensing certain loads. This can have an effect on the utilization of the CPU, disk peripherals, and channels.

CPU

The shortened IO wait times enable more data processing to take place in the same time; the CPU utilization can thus increase. This is merely a consequence of the compressed load. The increased CPU utilization by PAV itself is negligible; the CPU utilization as a whole is as a rule not influenced.

Disk peripherals

The requirements with respect to the storage system and the RAID group increase, and in consequence the hardware IO times also increase. Slight increases are an unavoidable consequence of PAV and constitute the expected behavior as long as the increase in the hardware I/O time is significantly lower than the reduction in the software I/O time, see also the measurements in section "Performance when PAV is used".

However, if the corresponding RAID group does not have sufficient reserves, a new bottleneck can occur here (above all when the cache hit rate is low), which means that the performance enhancement can be lower than expected. This can be recognized from the sharply increasing hardware IO times. In such a case, the following measures can make sense in order to increase the volume's raw performance:

1. distribution of the data to one or more RAID groups with a lower workload,
2. reconfiguration of the RAID group to a RAID level with a higher performance,
3. striping over more disks, or
4. in extreme cases switching to faster hard disks or a larger cache size.

In all cases the reason for the bottleneck should be understood beforehand. The first indications for this can be provided, for instance, by the cache hit rate, the IO rate, the IO size, and the ratio of write to read IOs. It should above all be clarified whether the RAID group is fully utilized by a particular load on one volume, or if the overload is a result of multiple loads (possibly also of other systems) on various volumes in the RAID group.

Channels

The channels are also utilized better by PAV. After PAV activation, the increase in the channel utilization should also be checked. Possibly further paths should be made available, above all in the case of throughput-oriented loads.

Measurements for storage systems

The results of IO measurements with the SE servers SE710, SE730, SE300B and SE310 are presented in this section.

Measurements for storage systems connected to /390 servers SE710 and SE730

This section contains measurement results for the performance of the /390 server SE710 when connected to a ETERNUS DX600 storage system via FC. The results may also apply to a SE730 server.

The following configuration was used for the measurements:

- SU710-70 server with 3,600 RPF in native mode.
- Connection of the server to the storage system via FC controllers each operating at 16 Gbit/s. Each volume could be reached over 2 paths.
- The ETERNUS DX600 storage system with a 128-GB cache and physical disks each with a capacity of 300 GB and operating at 15,000 rpm
- Up to 16 volumes on disks of the type D3435 with data format NK2 without mutual interference (1 volume each on a RAID group RAID 1/0(3+3)).
- All the tasks performed only synchronous IOs.

Performance of one task

The tables below show the performance which one user task can achieve with its disk accesses without a competitive load:

- I/O rate in the number of IOs per second (IOs/s)
- Throughput in Mbyte/s

The performance was measured for the standardized load types with different block sizes and a read hit rate of 100%:

Load type	Number of IOs/s for one task with block size				
	2 KB	16 KB	32 KB	160 KB	480 KB
sequential read	6,536	5,359	4,535	1,753	707
sequential write	4,813	3,927	3,348	1,243	500
random25	6,070	4,995	4,139	1,583	641

Load type	Throughput for one task in Mbyte/s with block size				
	2 KB	16 KB	32 KB	160 KB	480 KB
sequential read	12	83	141	274	331
sequential write	9	61	104	194	234
random25	11	78	129	247	300

Performance of one 16-Gbit channel

The /390 servers SE710 and SE730 contain FC channels working at 16 Gbit/s. Here, the throughputs via one channel path with a cache hit rate of 100% are collocated.

The results for two cases are shown below:

- IOs/s with 4KB blocks and
- Mbyte/s when using 480-KB blocks.

FastDPAV-Aliases	Vol.	Tasks	seq. read		seq. write		random25	
			IO/s (4 KB)	Mbyte/s (480 KB)	IO/s (4 KB)	Mbyte/s (480 KB)	IO/s (4 KB)	Mbyte/s (480 KB)
0	1	3	6,253	323	4,698	249	5,785	299
	1	8	6,255	323	4,543	81	5,781	299
	6	12	25,100	1,014	22,481	1,130	24,311	1,105
6	1	3	14,869	775	11,690	600	13,675	672
	1	8	24,688	1,012	21,443	402	23,776	1,103
	6	12	27,603	1,015	27,060	1,299	27,230	1,200

With 12 Tasks in parallel and using large IO block sizes, more than 1,000 Mbyte/s are achieved for read and up to 1,300 Mybte/s for write operation. Thereby, the limits of the IO system are reached.

If more than 3 tasks are working on 1 volume, waiting times (sw time) will sharply increase, leading to a lower overall throughput. Using FastDPAV, a significantly better throughput is achieved.

The tables below show the IO rates and throughputs for various block sizes for measdurements via 1 channel path, using 12 tasks on 6 volumes and FastDPAV.

Load type	IO rate in IO/s for 12 tasks with block size						
	2 KB	4 KB	8 KB	16 KB	32 KB	160 KB	480 KB
sequential read	27,682	27,603	27,204	26,396	23,091	6,485	2,167
sequential write	27,150	27,060	26,791	26,299	24,899	7,849	2,772
random25	27,322	27,230	26,951	26,275	23,835	7,629	2,561

Load type	Throughput in Mbyte/s for12 tasks with block size						
	2 KB	4 KB	8 KB	16 KB	32 KB	160 KB	480 KB
sequential read	54	107	212	412	721	1,013	1,015
sequential write	53	105	209	410	778	1,226	1,299

random25	53	106	210	410	744	1,192	1,200

For read IOs, maximum throughput is achieved with 128 KB and higher at ~1,000 Mbyte/s. For write IOs, it keeps rising with the block size.

Scaling of 16-Gbit channels

The following results were achieved during measurements with 12 tasks on 6 volumes, using FastDPAV with 6 base davices and 6 alias devices. Hence, for each task one free base and alias devices was available. First measurements were conducted using only 1 channel path. For the second measurement run with the same load, another channel path was added.

Channels	Vol.	Tasks	seq. read		seq. write		random25	
			IO/s (4 KB)	Mbyte/s (480 KB)	IO/s (4 KB)	Mbyte/s (480 KB)	IO/s (4 KB)	Mbyte/s (480 KB)
1	6	12	27,603	1,015	27,060	1,299	27,230	1,200
2	6	12	43,923	2,012	38,070	2,037	42,262	2,085

With 2 channel paths, the throughput and IO rates are significantly higher than with only one. They aren't doubled, as also the IO system of the SU /390 and the storage system limit the performance.

However, depending upon the particular application scenario, IO performance can be increased by connecting further channels.

Performance when PAV is used

The following measurements were conducted with 1 volume via 1 channel path, synchronous IOs and 100% cache hit rate. They show the effect of FastDPAV on IO times and, thereby, IO rates and throughput.

Tasks	FastDPAV Aliases	random25					
		IO/s (4 KB)	SW Time (4 KB)	HW Time (4 KB)	Mbyte/s (480 KB)	SW Time (480 KB)	HW Time (480 KB)
3	0	5,785	0.3	0.1	299	4.4	1.5
	6	13,675	0.1	0.1	672	1.8	1.8
8	0	5,781	1.2	0.1	638	12.2	1.5
	6	23,776	0.2	0.2	1,103	3.0	2.7

FastDPAV shows the same behaviour as static PAV. The parallelization of IO requests leads to significantly lower IO times an therefore to a higher IO rate and higher throughput. In most application cases FastDPAV can work witch less alias devices than a configuration with static PAV would need in total.

The increasing hardware IO times are not caused directly by FastDPAV, but are a result of the increased throughputs, and thus of the higher workload of the channel and storage system.

Measurements for storage systems connected to x86 servers SE310

This section contains measurement results for the performance of the x86 server SE310 when connected to an ETERNUS DX600 storage system via FC.

The following configuration was used to make the measurements:

- SU310-20 server with 430 RPF in native mode.
- Connection of the x86 server to the storage system via 2 FC switches with 2 * 4 FC connections (16 Gbit/s). The various volumes and LUNs were each accessible over four paths.
- The ETERNUS DX600 storage system with a 128-GB cache and physical disks each with a capacity of 300 GB and operating at 15,000 rpm
- Volumes on disks of the type D3435 with data format NK2 with low mutual interference (1-2 volumes each on a RAID group RAID 1/0(3+3)).
- All the tasks performed only synchronous IOs.

Performance of one task

The tables below show the performance which one user task can achieve without a competitive load in the case of disk accesses to the ETERNUS DX600 storage system:

- I/O rate in the number of IOs per second (IOs/s)
- Throughput in Mbyte/s

The performance was measured for the standardized load types with different block sizes and a read hit rate of 100%: The Mbyte figures are rounded to 1 Mbyte/s.

Load type	Number of IOs/s for one task with block size				
	2 KB	16 KB	32 KB	160 KB	480 KB
Sequential read	5.559	4.257	3.273	1.140	433
Sequential write	4.399	3.612	2.964	1.106	438
Random25	5.150	4.052	3.186	1.132	433

Load type	Throughput for one task in Mbyte/s with block size				
	2 KB	16 KB	32 KB	160 KB	480 KB
Sequential read	10	66	102	178	203
Sequential write	8	56	92	172	205
Random25	10	63	99	176	203

Performance at high load

The tables below show the performance which can be achieved with disk accesses to the ETERNUS DX600 storage system using many tasks:

- I/O rate in inputs/outputs per second (IOs/s)
- Throughput in Mbyte/s

The throughput was measured with 8 tasks operating in parallel on 4 volumes for the standardized load types with different block sizes and a read hit rate of 100%: The The Mbyte figures are rounded to 1 Mbyte/s.

Load type	**Number of IOs/s with 32 tasks and block size**				
	2 KB	**16 KB**	**32 KB**	**160 KB**	**480 KB**
Sequential read	20.037	14.666	10.893	3.456	1.331
Sequential write	20.187	15.111	11.396	3.803	1.433
Random25	19.799	14.599	10.869	3.459	1.363

Load type	**Number of IOs/s with 32 tasks and block size**				
	2 KB	**16 KB**	**32 KB**	**160 KB**	**480 KB**
Sequential read	39	229	340	539	624
Sequential write	39	236	356	594	671
Random25	38	228	339	546	639

The utilization of the BS2000 CPUs was around 100% for all measurements. The values ascertained thus correspond to the maximum achievable throughputs with an SU310-20.

> As long as the BS2000 CPUs do not develop into a bottleneck, with identical RPF performance the IO rate of x86 servers depends to a great extent on the performance of the disk storage system. Both the cache size and the line speed of the FC interface (8 Gbit/s, 16 Gbit/s) also play a role.

Hardware service times

The hardware service times which can be achieved in practice depend very much on the cache hit rate and the load on the hard disks. With a cache hit rate of 100%, the service times are around 0.1 ms with very small blocks, and 3-5 ms with very large blocks.

In the case of an overload, this range increased to approx. 1-16 ms. Above all in the case of access with small blocks (OLTP mode!), the times then increase disproportionately.

Measurements for LAN connections on SE servers

This section analyzes the communication performance of the SE servers.

- Connection of an SE server to a customer network
- General aspects for determining the network performance
- SE710 and SE730
- SE310
- Measurement tools for LAN connections

The following terms and abbreviations are used here:

Abbreviation	Meaning
1 GbE	Gigabit Ethernet (max. 120 MB/s per direction)
10 GbE	Gigabit Ethernet (max. 1200 MB/s per direction)
MTU	Maximum Transmission Unit
Standard frames	The MTU is 1500 bytes long
Jumbo frames	The MTU is 9000 bytes long
8GFC / 16GFC	8/16 Gigabit Fibre Channel
VCA	Virtual Channel Adapter (when using link aggregation, see the "BCAM" manual [2 (Related publications)].
1FC4VCA	Link aggregation over 4 VCAs via one FC
2FC8VCA	Link aggegation over 8 VCAs via two FCs

Connection of an SE server to a customer network

Connection of an SE server to a customer network

SE servers can be connected to a customer network either directly or via the Net Unit (internal LAN switch of an SE server).

As is to be expected, the direct connection provides somewhat higher performance because the transaction time is somewhat lower than via the Net Unit.

LAN connection to a /390 server

A /390 server is always connected to a LAN via an HNC (High-speed Net Connect) channel adapter (see the "HNC" manual [13 (Related publications)]).

An HNC is equipped with the following controllers:

- A Fibre Channel Controller for direct connection to the /390 server
- A Gigabit Ethernet Controller for connection to the customer network

It is possible to operate more than one HNC on a /390 server. Up to two GFC channels are supported between an HNC and a /390 server.

LAN connection to an x86 server

X86 servers are connected to a LAN via a PCIe controller. The I/Os to the controller are - like all I/Os - directed via the I/O processors.

General aspects for determining the network performance

The typical application types transaction-oriented mode and throughput-oriented mode (mass data transfer similar to file transfer) are examined.

- Transaction-oriented mode

  Sending and receiving short messages (ping-pong, 10 bytes, without times).

  Measured in transactions per second (TA/s).

  Properties

  - The maximum transaction rate is reached with a large number of parallel connections. In this case the BS2000 CPUs are very heavily utilized by communication.
  - The best average response time between an SE server and a partner system is determined from the transaction rate for a connection.
- Throughput-oriented mode

  Sending or receiving messages with a length of 8, 16, 32, 64 KB.

  Measured in megabytes per second (MB/s).

  Properties

  - With parallel connections (or in the case of a low line speed already with one connection), the maximum throughput is reached. In this case the workload on the lines is very high.
  - The throughput for a connection is extremely important for applications with high data transfer over the network, e.g. in the case of file transfer.

Under optimum conditions, the results apply for a Sockets application (TCP/IP), where the messages are not processed further in the course of our measurements. The message transfer is from main memory to main memory, so there are no disk accesses. In the case of real applications, their CPU and disk requirements must be taken into account, as must the fact that networks might not be optimized.

Sufficient CPU capacity is required to fully utilize high-performance networks.

Provided the limits of the lines have not been reached, in the case of parallel load the performance increases with the number of CPUs.

When jumbo frames are used, the network, switches, and partner systems must also be configured accordingly.

The maximum transaction rates are appreciably higher than all current normal requirements of, e.g. OLTP applications. The transaction rate is normally limited by full utilization of the BS2000 CPUs.

The data throughput when sending messages depends on the message length used. The following always applies: the longer a message, the higher the throughput.

SE710 and SE730

When an HNC is connected to a 1 GbE customer network, this can be fully utilized thanks to the 16GFC connection. Up to 2x4 1GbE ports are supported.

For a connection to a 10 GbE customer network, the HNC can be upgraded by one or two 10 Gigabit PCIe Controllers (each with 2 ports). A 10 GbE can be utilized by employing link aggregation. This enables multiple Virtual Channel Adapters (VCAs) to be bundled. One device pair [read/write] in BS2000 corresponds to each VCA. The following applies here:

- By means of link aggregation via one 16GFC (1FC4VCA), this one FC channel operates at up to 60% of the theoretical capacity. Therefore, the 10GbE can only be used up to 75%.
- A 10 GbE can be fully utilized by means of link aggregation via two 16GFCs (2FC8VCA).

The results (TA/s, MB/s) for the LAN connection of an SU730-70 (3,600 RPF via an HNC M4 with software version V6.5) are presented below. Measurements were made under BS2000 OSD V21.0A with openNet Server V4.0. The results may also apply to an SU710 with the same performance.

The following variants were measured:

- Connection of the HNC to a 1GbE customer network and a 10GbE customer network
- Use of standard frames and jumbo frames.

The following results are achieved here:

Transaction-oriented load

- Maximum transaction rate of over 271.000 TA/s (here the CPU utilization of the SU730-70 was around 85%)
- Mean response time with one connection around 0.37 milliseconds

Throughput-oriented load over the 1GbE

- The network is fully utilized: over 110 MB/s is achieved.
- The CPU utilization of the SU730-70 is up to approx. 12%.
- The setting of jumbo frames results in enhanced performance: the CPU requirement is lower with a similar throughput.

Throughput-oriented load over the 10GbE

The maximum throughput which can be achieved here depends on whether and how link aggregation was configured.

- Without link aggregation:
  - When sending 280 MB/s
  - When receiving 380 MB/s.

  Here the CPU utilization of the SU730-70 is up to 13%. The configuration limits the performance (i.e. the device pair in BS2000). This throughput is already archieved with one connection.

- With link aggregation via one 16GFC and 4 VCAs per FC (1FC4VCA):
  - When sending more than 900 MB/s
  - When receiving more than 900 MB/s

  Here the CPU utilization of the SU730-70 is around 30%.
- With link aggregation via two 16GFCs (2FC8VCA):
  - When sending 1,100 MB/s
  - When receiving 1,100 MB/s

  Here the CPU utilization of the SU730-70 is around 40%. The 10GbE limits the performance.

These values are not attained for communication with less powerful partner systems, nor are they possible when other components such as disks or a less than ideal network slow data transfer down.

With the 10GbE the setting of jumbo frames results in only a very low performance enhancement in the CPU requirement compared to standard frames.

SE310

The following was measured on an SU310-20 (430 RPF, with BS2000 OSD/BC V11.0B and openNet Server V4.0):

Transaction-oriented load

- Maximum transaction rate more than 23,600 TA/s with 100% CPU utilization
- Best mean response time approx. 0,3 milliseconds

Throughput-oriented load for parallel connections

With throughput-oriented mode the CPU utilization depends on the message length. The longer the message, the lower the BS2000 CPU requirement. This dependency does not exist when receiving. Both when sending and receiving, the CPU requirement increases with the number of parallel connections.

The table below shows the maximum throughput and the CPU utilization for 64 KB messages on an SU310-20 for a 1GbE and for a 10 GbE, each with 4 parallel connections and both with standard frames and with jumbo frames.

MTU[bytes]	**10 GbE 1500**		**10 GbE 9000**		**1 GbE 1500**		**1 GbE 9000**	
	Send	Receive	Send	Receive	Send	Receive	Send	Receive
MB/s	880	1,060	810	1,025	112	113	115	118
CPU% for 64KB	100	100	100	100	31	91	24	80

For both the 1 GbE and the 10 GbE a very high utilization can be achieved. For a 10 GbE the BS2000 CPU limits the performance.

Throughput with one connection

- The 1 GbE is fully utilized with one connection both when sending and when receiving: 110-120 MB/s is achieved. The CPU utilization is up to 45% when sending (depending on the message length) and 70% when receiving.
- The following applies for a 10 GbE with one connection:
  - When sending, the throughput depends on the message length. Approximately the following throughput was achieved with both standard frames and with jumbo frames:
    - 260 MB/s with a message length of 8 KB (CPU utilization 75%)
    - 600 MB/s with a message length of 32 KB (CPU utilization 87%)
    - 925 MB/s with a message length of 64 KB (CPU utilization 89%)
  - When receiving, the throughput depends on the message length and amounts to around 1.000 MB/s with a CPU utilization of 87% both with standard frames and with jumbo frames.

Measurement tools for LAN connections

The following measurement tools exist for LAN connections:

- BCMON commands
- NETSTAT command
- openSM2 reports

BCMON commands

Using the BCMON command, you start cyclic BCAM monitoring and output the desired values at regular intervals. The values are displayed on the console and also written to the log file ($SYSAUDIT.SYS.CONSLOG.<date>.<counter>), so that the values can be evaluated later. See also the “BCAM” manual [2 (Related publications)].

Simple example for BCMON

This example shows how the throughput of a specific connection is measured. The total throughput can be measured by not specifying LINE=.

/BCMON MODE=ON,RECORD=L2,LINE=LINEFCGB,SEC=30 (command on the console)

Every 30 seconds the measured values accrued in this time are output, e.g.:

```
<C %BCAM-000... %  BCA0B15 Line LINEFCGB:
BYTES=(I=929.088.154/O=930.140.116); I/O'S=(I=50.259/O=33.216);
PACKETING=(I=4,60/O=6,98)                                              1.
<C %BCAM-000... %  BCA0B16 Line LINEFCGB: UNICASTs=(I=231.372/O=231.981);
MULTICASTs=(I=0/O=0)                                                   2.
<C %BCAM-000... %  BCA0B17 Line LINEFCGB: I/O ERRORs=(I=0/O=0);
WAIT+RETRIES=0; DISCARDS=(I=0/O=0); PROTOCOL ERRORs=(I=0/O=0);
L2 PROTOCOLs=(I=0/O=0)                                                 3.
```

1. Line LINEFCGB : Name of the line
   BYTES=(I=): Number of bytes received
   BYTES=(O=): Number of bytes sent
   I/O'S=(I=): Number of inputs
   I/O'S=(O=): Number of outputs
   PACKETING=(I=): Number of messages per input
   PACKETING=(O=): Number of messages per output
2. Line LINEFCGB : Name of the line
   UNICASTs=(I=): Number of UNICASTs received
   UNICASTs=(O=): Number of UNICASTs sent
   MULTICASTs=(I=): Number of MULTICASTs received
   MULTICASTs=(O=): Number of MULTICASTs sent
3. Message BCA0B17 is merely an error counter

NETSTAT command

The NETSTAT program can be used to request information about applications, connections, routing data and network interfaces. The functionality of the NETSTAT program is also available via BCAM command SHOW-NET-STATISTICS/NETSTAT. The NETSTAT program can be used under any user ID without any privileges with the SHOW-NET-STATISTICS/NETSTAT command. See also the “BCAM” manual [2 (Related publications)].

Simple example for NETSTAT

This example shows how the total throughput is measured using NETSTAT.

/NETSTAT INTERFACE-SUM-RATE=*YES,WAITTIME=30 (user command)

After the load messages and a display of all the values accrued to date, the standardized values for the inputs and outputs are output every 30 seconds (in each case in packets and bytes):

```
INTERFACE sum rate                          Date: Fri Nov 23 10:20:35 2012
  PacketsIn       BytesIn    ErrorsIn  PacketsOut      BytesOut  ErrorsOut
   58255627 1979306135058           0    70849955  516799149494          0
INTERFACE sum rate                          Date: Fri Nov 23 10:21:05 2012
  dt(s)    PktsIn/s   KByteIn/s   ErrIn   PktsOut/s  KByteOut/s  ErrOut
 30.007      363.35       24.35       0      265.17      552.28       0
 30.005      345.44       22.70       0      229.20      476.90       0
 30.005      319.58       20.66       0      194.33      404.37       0
```

openSM2 reports

The following SM2 monitoring programs supply important information on networking (see the “openSM2” manual [18 (Related publications)]):

- BCAM-CONNECTION
  Monitored data on connection sets
- CHANNEL-IO
  Monitored data on the channel load and channel transfer rates.
- PERIODIC TASK
  Monitored data on tasks: utilization of the BCAM tasks (e.g. BCAM and BCA0)
- RESPONSETIME
  Monitored data on response times, think times, transactions times, and wait times in the BCAM pool
- SAMPLING-DEVICE
  Monitored data on I/Os, data volume and utilization of devices
- TCP-IP
  Monitored data on TCP/IP connections

Net-Storage behavior

BS2000 permits UNIX file systems to be accessed via NFS.
This enables BS2000 files to be saved and processed by file servers and Net-Storage in released directories.

From the BS2000 viewpoint, on the storage medium Net-Storage, too, data is saved to volumes which are, however, permanently assigned to a pubset. These serve as a storage extension for the pubset. It is thus possible to exceed the maximum possible storage space of an SF pubset (4 TB).

Basic information on Net-Storage can be found in the “Introduction to System Administration” [10 (Related publications)].

Net-Storage is used primarily to save (large) files which do not need to be accessed with maximum performance. Performance-critical data (e.g. database files) should still be saved to pubsets.

There are several reasons for this:

- The performance capability of state-of-the-art networks is constantly increasing, but the bandwidth must be shared with other participants in the network. Consequently, in contrast to FC peripherals, no exact forecasts of throughput and latency periods can be made.
- Read access to Net-Storage is at a higher performance level than write access. A considerably higher throughput can be achieved by reading and writing large I/O blocks than with small I/O blocks.
- The throughput increases with the number of parallel jobs and with the block size until a bottleneck is reached. Thus with sequential processing of large blocks (HSMS/ARCHIVE, COPY, File Transfer), very high throughput values can be achieved which extend to the limits of the bandwidth provided in the network.
- If the configuration and the hardware of the network permit, jumbo frames should be used (MTU 9,000 bytes). The MTU must also be configured or supported on all the network components involved (e.g. switches).

Data backup

The following aspects are relevant to performance when local data backup takes place:

- Hardware performance
  - Throughput of storage systems
  - Throughput of tape storage systems

  The performance of the hardware determines the throughput for data backup provided that the settings for the software used ensure optimal performance.
- Software-based backup concepts
  - Logical data backup with HSMS/ARCHIVE
  - Physical data backup with FDDRL

Recommendations for performance optimization are provided for the software.

Hardware performance

Balanced and combined application of hardware components is required for optimum performance in data backup. This principally requires:

- storage systems such as ETERNUS JX, DX, AF or Net-Storage
- tape storage systems such as ETERNUS LT or MTC archive systems

The backup performance is determined by the lowest throughput value of the hardware components involved.

The configuration of such systems must be geared to the scope of the backup and the time windows available for data backup.

The general recommendations for the peripherals (see section "Recommendations for high-performance operation of storage systems") also apply for data backup.

A certain share of the CPU performance of the BS2000 CPU is also required for data backup.

Throughput of storage systems

For data backup, the data from one or more storage systems is read and written to tape; for data recovery, data is read from tape and written to disk.

Storage systems are connected to the server via Fibre Channel using multiple paths. This permits full use to be made of their performance capability.

If the options of parallelization (suitable RAID groups, PAV, multiplexing) are used, the degree of connection utilization can be extremely high in the case of throughput-oriented data backup loads.

Performance recommendations

- Use of state-of-the-art, high-speed storage systems with FBA disks D3435 and data format NK2. These disks can be operated with an IO length of up to 480 KB.
- Use of storage systems with as large a cache as possible. The size selected should ensure that the read hit rates in OLTP mode are 60 - 80%.
- Multipath connection of the storage systems to the server.
- Use of RAID level RAID 1/0, RAID 5, or RAID 6 (see section "Replication: volume-based mirroring for storage systems"). This also enables “large” volumes (more than 32 GB in size) to be used.
  Fewer parallel IOs can be executed with RAID 1. RAID 1 is therefore less suitable when a high throughput is to be achieved.
- Use of FastDPAV on /390 servers, see "Advantages of PAV".

Characteristics of the data backup load

For data backup, very large quantities of data are read and written sequentially from and to the storage system in several parallel, asynchronous data streams. Current storage systems can can read from the physical disks using read-ahead mechanisms and write to the physical disks using Delayed Fast Writes. The limits of the caches can be reach when doing this. Consequently, an appropriate cache size must be ensured.

> ¡ The performance of the physical disks and of the RAID groups is, however, also decisive for the throughput which can be achieved. With a data backup load, today's storage systems achieve a very good throughput rate with read-ahead or Delayed Fast Write.

Measurements

> ¡ The measurement results in this section assume favorable conditions, e.g. no parallel load on the measured configuration, sufficient CPU performance.

To determine the maximum throughput of backup volumes of the storage system, only reading from the storage system to BS2000 main memory is performed using a test program based on ARCHIVE without writing to tape.

The measurements were made with an SE730-20D and a DX8700 storage system. The storage system was connected via 4 FC paths with 16Gbit/s each. The backup data consisted of large files (1000 MB) on one single volume. Under these circumstances a throughput of 800 MB/s was measured.

The difference in throughput between large files (1000 MB) and medium files (10 MB) is small. With small files (1 MB) it is significant.

Throughput of tape storage systems

For data backup the data is written onto one or more tape storage systems and read from these when the data is restored.

Tape storage systems are connected to the server via Fibre Channel. These connections may not be used with the storage systems.

The FC channels of the BS2000 servers to the tape storage systems can be almost fully utilized by just a few devices when data backup takes place.

Performance recommendations

- Use of ETERNUS CS

  Compared to a real tape archive system, ETERNUS CS has the advantage that the data is stored physically on an MTC only when it has been transferred completely to the virtual MTC device. This enables the parallelism of the data backup to be increased without increasing the number of real MTC drives.
- LTO tape devices with a direct connection to the server:

  Use only preferred quality cartridges

Throughput and compression with directly connected LTO drives

The data written from the server is initially transferred to the LTO device’s cache. Here it is compressed as far as possible by the LTO device. The compressed data is then written to the physical tape in the LTO device. An analogous procedure is used for reading data.

The following performance values apply for LTO devices (without compression of the data in the drive):

Device Type	Maximum Throughput	Capacity
LTO-4	120 Mbyte/s	800 GB
LTO-5	140 Mbyte/s	1,500 GB
LTO-6	160 Mbyte/s	2,500 GB

These performance values relate to the device / the physical cartridges.

According to information from the LTO vendors, the compression up to LTO-5 is 2:1, and with LTO-6 2 and higher 5: 1. With compression in the LTO drive, the following throughput is thus achieved:

- LTO-4: up to 240 Mbyte/s
- LTO-5: up to 280 Mbyte/s
- LTO-6: up to 400 Mbyte/s

Measurements with LTO-6 tape devices

The tables below show the possible throughput with an LTO-6 tape device. The load corresponds to data backup using HSMS/ARCHIVE with large tape blocks. The I/Os are performed with a test program from the BS2000 main memory (not from disk) to the MTC device.

The throughput specified is for data which is difficult to compress (compression factor 1) and for data which can be compressed normally (factor 2-3).

The throughput rates specified refer to one task during the actual backup phase, in other words without mounting the tape, without starting and ending the programs, and without creating the report files.

> The measurement results in this section assume favorable conditions, e.g. no parallel load on the measured configuration, sufficient CPU performance.

SE700 server, direct connection of an ULTRIUM-TD6 over Fibre Channel (8 Gbit/s)

Compression factor	Throughput (Mbyte/s) Write	Throughput (Mbyte/s) Read
1	160	160
2 - 3	356	372

At 160 Mbyte/s, the maximum throughput of LTO-6 is reached in the case of non-compressible data. When the data is compressed in the tape device, the throughput is near the limit of the maximum throughput of LTO-6.

Measurements on an SE300 server with a comparable tape system yield similar results to when performed on an SE700.

Backup performance

Two types of data backup exist on BS2000 systems:

- Logical data backup with HSMS/ARCHIVE
- Physical data backup with FDDRL

The following recommendations apply irrespective of the backup method:

- Use of state-of-the-art, high-speed storage systems with FBA disks D3435 and data format NK2. These disks can be operated with an IO length of up to 480 KB.
- Use storage systems with as large a cache as possible. The size selected should ensure that the read hit rates in OLTP mode are 60 - 80%.
- Multipath connection of the storage systems to the server.
- Use of RAID level RAID 1/0, RAID 5, or RAID 6 (see section "Replication: volume-based mirroring for storage systems"). This also enables “large” volumes (more than 32 GB in size) to be used RAID 1 is not very suitable for data backup.
- Use of FastDPAV (fast dynamic PAV) in BS2000 on /390 servers.

Logical data backup with HSMS/ARCHIVE

In the case of logical data backup with HSMS (see the "HSMS" manual [14 (Related publications)]), BS2000 files, POSIX files, and job variables of the local BS2000 system are read from one or more data media and saved consecutively to a backup file on one or more data media.

Only BS2000 files and job variables can be saved with ARCHIVE (see the "ARCHIVE" manual [1 (Related publications)]).

Backup volumes

The backup volume for logical data backup depends on the application profile in the Data Center. For example, the following values are available for the various backup types and quantities of backed-up data:

- Full backup: complete active data set
- Incremental backup: only files that changed since the last backup are backed up
  -

The number of backup media required depends on the volume of the full and incremental backups and must be determined separately for each application.

Performance recommendations for HSMS/ARCHIVE

The main aim of the performance recommendations is to decrease the backup time for logical data backup with HSMS. When the backed-up data is reconstructed, somewhat different throughput rates apply than for data backup. However, they are usually only critical if a full backup is to be loaded. With partial restoration, which runs in parallel with normal operation, the throughput plays a subordinate role.

The recommendations presented can be used individually and independently of each other. They are a,ways beneficial in all situations.

It is not possible to specify the improvement in performance for each recommendation precisely. The general situation and usage type (e.g. file size) must be taken into consideration. The throughput can be significantly lower with a large number of small files than in the ideal case with a small number of very large files.

1. Limiting the scope of the backup through file selection and the incremental process

   Using the FROM-/EXCEPT-FILE parameters (positive and negative specification with partial qualification) is recommended.

   A maximum backup class can be predefined: only files whose class is less than or equal to the specified backup class are saved (as a file attribute with the default backup class = A).

   Temporary files and paging files are never saved.

   Management classes in SM pubsets (freely definable file attribute for group formation in SM pubsets) can be used for file selection when backup and migration take place.

   You can select whether migrated files should also be backed up (backup operand SAVE-DATA). By default only catalog entries of migrated files are also backed up.

   Two processes are available for incremental backups between the periodical full backups:

   - MODIFIED: Save files only when these do not already exist in the archive
   - BY-CATALOG-MODIFICATION: Save files if these were modified only after a specified date or after the date of the last backup for the same archive
2. Decentralized organization (archives per pubset) also for large SF pubsets
3. Reducing the backup time through parallel I/Os (increased I/O load!)

   HSMS enables multiple jobs to be performed in parallel. The degree of parallelism can be specified by the number of HSMS server tasks.

   In turn, parallel data streams (to multiple tape devices) can be used in a job. This is controlled using the PARALLEL-RUNS operand in the action statement or an archive attribute. Under ARCHIVE this is controlled using the DRIVES operand in the SAVE statement.

   The cartridges with a high capacity should always use the Several-SVID-Format for continuing (CONTINUE) tapes. This is practically a prerequisite for parallel operation, otherwise the tape consumption is high.

   > Increasing parallel operation reduces the runtime, and the number of inputs/outputs per second is increased.
   > When a large number of tape devices are used (e.g. with ETERNUS CS), these can be used in parallel by HSMS/ARCHIVE.
   >
   > Backup jobs which relate to the same archive directory or same HSMS archive cannot run in parallel.

4. Increasing throughput of disks and tapes

Using large tape blocks increases the performance and tape utilization. By default, 256 KB tape blocks are used when saving to tape cartridge using HSMS and ARCHIVE. To permit this, the parameter BLOCK-SIZE-T-C is predefined to the value BIG in the ARCHIVE parameter file. It may then be necessary to specify BLOCKING-FACTOR=*STD / *MAX in the HSMS statements and archive definitions.

> ¡ LTO drives are always written to using a block size of 256 KB irrespective of predefined parameters and archive definitions.

HSMS/ARCHIVE places an additional load on the disk side through catalog and file accesses (e.g. to the HSMS or ARCHIVE directory) and the fragmentation of small files.

To reduce this load as far as possible, PLAM libraries should be used instead of a large number of small files.

In the case of ARCHIVE RESTORE, the files are deleted on the disk before being rewritten, and then created anew (SPACE=REORG parameter). If no reorganization of the disk is required, the SPACE=KEEP parameter must be specified for RESTORE. This causes the files on the disk area to be overwritten, and the number of catalog operations and the number of IOs compared to the default setting are reduced.

5. Multiplexing with PAV (/390 servers)

   HSMS/ARCHIVE generates asynchronous I/O operations These are executed in parallel when the storage system's Parallel Access Volume (PAV, 390 server) function is used together with a suitable RAID level (see "RAID levels and their performance").

6. Reducing the downtime for online operation (application restart before the end of the actual backup)

   HSMS and its CCOPY (Concurrent Copy) function enable consistent backups to be created simultaneously and the data which is to be backed up to be processed in any required way. To permit this, the application which processes the files to be backed up must be terminated briefly before backup begins or be guided to a synchronization point. The backup job can then be started. After an initialization phase which can be monitored with a job variable, the application can be enabled again. Backup then runs parallel to file processing.

   Backup with CCOPY uses either a temporary scratch file or a replication function of the storage system. Further information on this subject is provided in the "HSMS" manual [14 (Related publications)].

   Use of a replication function of the storage system permits load splitting between the application on the original disk and the backup on the mirror disk, and as a result operation of the application is disturbed considerably less by the backup I/Os.

   > ¡ In these cases a task job variable shows the status for the application restart before the task is completed.

7. Performance of the catalog accesses in the case of DMS backups

   In shared pubset operation the normal backup requests are sent to the master side in HSMS to utilize the quicker local catalog accesses without MSCF traffic.

   Backup requests with CCOPY of mirror disks of the storage systems are always performed locally in shared pubset mode because the catalog accesses to the split mirror disks of the storage systems are performed locally. This enables the I/O load to be split (e.g. with two systems for application and backup).

8. Backup server

The concept of the backup server in the SE environment uses a different strategy. Here a dedicated BS2000 system is used as the backup server. Backup jobs which are started on BS2000 systems for pubsets to which the backup server also has access are then backed up by this server. This increases the MSCF workload with respect to accesses to the catalog when the productive system is the master in the SPVS network, but relieves the productive system of the load of data backup.

Here, too, backup jobs are processed locally by mirror disks with CCOPY and not sent to the backup server.

9. When a very large number of files are to be backed up, summary reports (only in the event of errors are entries made on a file-by-file basis) should be used instead of full reports (entries for all files) in order to keep the trailer time short.
10. Reducing RESTORE-FILES run-time from K- to NK-Pubsets
    If RESTORE-FILES has to be performed from a K-Pubset to a NK-Pubset the respective files should be carefully chosen rather than restoring the whole Pubset. Every file will be individually converted from K to NK, resulting in a dramatic increase in run-time when restoring large amounts of files in this specific case.
11. Database backups during ongoing database operation

    Files which can be backed up but are open for writing can be backed up using the backup option SAVE-ONLINE-FILES = *YES, e.g. for UDS databases with log files and for Oracle database files or single tablespaces.

    SESAM database backup by SESAM/SQL via the HSMS program interface is performed in the same way, also in combination with CCOPY of mirror disks of the storage systems. This results in minimal refreshment phases for the database.

Measurements with HSMS/ARCHIVE and LTO-6 tape devices

The tables below show backup throughputs which can actually be achieved with LTO-6 tape devices under HSMS /ARCHIVE on /390 and x86 systems. In each case one volume with files permitting average compression (factor 2-3) was backed up.

The throughput rates specified refer to one task during the actual backup and restoration phase, in other words without mounting the tape, without starting and ending the programs, and without creating the report files.

SAVE and RESTORE scenarios with different file sizes were measured:

- Large files: 1.2 GB (1200 MB)
- Medium-sized files: 12 MB
- Small files: 1.2 MB

The throughputs were measured using a number of configurations, with the following hardware components being used:

- Servers: SE700-20 with 1,120 RPF and SE300-80F with 1,200 RPF
- Storage systems: ETERNUS DX600 and ETERNUS DX440
- SAN connection of the storage systems: 2-3 Fibre Channels operating at 8 Gbit/s
- Volumes: Different-sized NK2 volumes on RAID 1/0 (3+3)
- Tape devices: IBM ULTRIUM-TD6 connected to a Fibre Channel operating at 8 Gbit/s

In practice, the throughputs which can be achieved depend on the sum total of the components involved. For example, in the case of small files the throughput is limited by the accesses to the volumes. No generalization is therefore possible for throughputs in specific individual cases. Instead, the bandwidth of the throughputs measured is specified.

The throughputs achieved with servers with /390 and x86 architecture are at a comparable level when the same performance is available.

File size	Throughput with ARCHIVE in Mbyte/s SAVE	Throughput with ARCHIVE in Mbyte/s RESTORE
Small files (1.2 MB)	160 - 190	80 - 110
Medium-sized files (12 MB)	270 - 350	170 - 220
Large files (1.2 GB)	340 - 370	310 - 320

> The measurement results in this section assume favorable conditions, e.g. no parallel load on the measured configuration, sufficient CPU performance.

In order to display the CPU requirement on a comparable basis, the CPU utilizations measured were converted to RPF per Mbyte/s. The CPU requirement for backing up and restoring large and medium-sized files is between 0.2 and 0.7 RPF per Mbyte/s. Consequently, a fictitious system with 500 RPF, for instance, would have a workload of 10-35% with a data throughput of 250 250 Mbyte/s.

When small files are backed up and restored, the overhead increases because of the increasing number of catalog operations required.

The CPU requirement when backing up a large number of small files rises to a factor of 2, and when restoring with default settings, to a factor of 7.

In such cases you should take into account the optimization tips in section "Performance recommendations for HSMS/ARCHIVE".

> Particularly in multiprocessor systems it must be borne in mind that the CPU performance required for a backup task must be provided by just one BS2000 CPU.

The I/O processors of BS2000 servers in no way present a bottleneck.

0.0.0.1 Measurements with HSMS/ARCHIVE and ETERNUS CS8200 VTL

Measurements regarding data backup performance with the backup file located on the Logical Volumes of a ETERNUS CS8200 VTL were performed on a Server SE710. The results also apply to an SU730. The storage system volumes that were used with HSMS BACKUP- and RESTORE-FILES contained files varying in size (regarding PP, PAM Pages):

- Small files: 35510 * 621 PP (1,2 MB)
- Medium-sized files: 3630 * 6075 PP (12 MB)
- Large files: 36 * 621600 PP (1,2 GB)
- Systemmix: 11872 files of different sizes

The following hardware configuration was used:

- Server: SE710-20D (1220 RPF)
- Storage-System: ETERNUS DX8700, 16 Gbit/s FC, 2,5" SSD-M Drives
  Volumes: NK2 format, in 2 RAID-Groups (RAID1), 4-way connection
- ETERNUS CS8200 VTL: FC-connection with 16 Gbit/s via 2 channels, Logical Drives IBM 03590E1A (BS2000: DEVICE-TYPE=TAPE-C4)

Backup throughput with ETERNUS CS

The backups were performed on one Pubset with 1 task on 1 Logical Volume as well as using parallel backups of 2 Pubsets with 2 tasks on 2 Logical Volumes. With FastDPAV the following average throughputs were measured:

file size	HSMS/ARCHIVE throughput (MB/s)			
	BACKUP 1 task	RESTORE 1 task	BACKUP 2 tasks	RESTORE 2 tasks
small files	317	198	620	282 + 105
medium files	317	253		
large files	317	254	633	473

The throughputs using RESTORE are lower than those achieved using BACKUP. The Cache Hit Rate Write of the storage system occasionally dropped far below 100% during RESTORE operations. RESTORE subtask 1 with 2 small files and 2 tasks used about double the time of subtask 0. Hence in the table above values for 2 parallel tasks as well as the run-time of subtask 1 are pointed out. The many access operations on the BS20000-catalog lead to a significant overhead.

Comparison to ETERNUS DX

Comparison measurements with the HSMS backup file located on a Pubset of the storage system were conducted. The Pubset was connected via a different 4-way connection than the backed up Pubsets.
The throughputs achieved with FastDPAV as well as without PAV were as follows:

ARCHIVE 1 task file size	HSMS/ARCHIVE throughput (MB/s)			
	BACKUP FastDPAV	RESTORE FastDPAV	BACKUP without PAV	RESTORE without PAV
small files	290	158	170	111
medium files	471	515	176	187
large files	463	477	177	197

In this scenario with FastDPAV the BACKUP throughput using medium and large files is about 50% higher compared to using Logical Volumes. For RESTORE operations the throughput is about twice as high as before. The parallel Read- and Write-IOs on the backup file impact the results heavily.

Scaling with several tasks

When connecting the Logical Drives via 1 channel to the server the BACKUP operations with several parallel tasks scale well. When using RESTORE the storage system ist the limiting factor regarding throughput. HSMS-Backups (FastDPAV, volumes with large files) with a single task and 2/3/4 parallel tasks resulted in the following BACKUP and RESTORE throughputs:

ARCHIVE	HSMS/ARCHIVE throughput (MB/s)			
	BACKUP TAPE-C4	RESTORE TAPE-C4	BACKUP TAPE-U4	RESTORE TAPE-U4
1 task	317	254	316	253
2 tasks	625	467	621	465
3 tasks	917	493	911	491
4 tasks	1089	464	1104	464

The Logical Drives were configured as both IBM 03590E1A (BS2000: DEVICE-TYPE=TAPE-C4) and LTO Ultrium 4 (BS2000: DEVICE-TYPE=TAPE-U4).

There was no significant difference between DEVICE-TYPE TAPE-C4 and TAPE-U4.

Comparison K- and NK-Pubsets

Conducting further measurements the difference between backing up K-Pubsets and NK-Pubset with the backup file located on Logical Volumes was examined. The Pubsets each contained files of different sizes (Systemmix). Both BACKUP and RESTORE were used and tested several times. The following table describes the average throughputs for the different scenarios:

	HSMS/ARCHIVE throughput (MB/s)

Pubset BACKUP - RESTORE	BACKUP FastDPAV	RESTORE FastDPAV	BACKUP without PAV	RESTORE without PAV
K - K	287	175	124	93
K - NK	287	11	124	11
NK - NK	291	176	254	162
NK - K	290	178	249	109

The results varied siginificantly between runs, most of all while using RESTORE. With BACKUP K and a following RESTORE NK all files were converted using PAMINT. This lead to a significant throughput decrease to 11 MB/s. This decrease in performance is unrelated to PAV, but is the reason for increased RESTORE run-times.

Physical data backup with FDDRL

In the case of physical data backup with FDDRL (see the "FDDRL" manual [11 (Related publications)]), whole data media are saved (individual disks or entire pubsets (home pubset, exported pubset, split disk mirror). Here all the data on a data medium, including the data labels, is written block by block in the physical order without catalog access to MTC.

FDDRL (DUMP function) by default transfers only the areas marked as occupied in the F5 label. As a rule, therefore, in a physical backup with FDDRL no more data is transferred than in a logical backup with HSMS. In the case of physical backups, however, neither incremental backups can be created nor can individual files be restored.

Performance recommendations for FDDRL

The recommendations presented can be used individually and independently of each other. They are a,ways beneficial in all situations.

1. Reducing the backup time through parallel I/Os (increased I/O load!)

   Backing up multiple volumes in parallel under their own FDDRL subtasks. The degree of parallelization can be specified with the FDDRL parameter TASK-LIMIT.

2. Setting the task priority for the FDDRL subtasks
   - If an FDDRL run is to be executed as quickly as possible and without taking into account the response time behavior during BS2000 operation while the workload is low, the TASK-LIMIT parameter of the job class should have the value of the available tape devices (at most 16) in order to achieve a good throughput through the h igh degree of parallelization. The FDDRL parameter RUN-PRIORITY should be set to “high”.
   - If an FDDRL run is to be executed during normal BS2000 operation without impairing the performance of the main application, the TASK-LIMIT parameter of the job class should be set relatively low to take into account the other applications. In the event of multiple parallel runs a noticeable worsening of the response time behavior of the other applications must be expected. The RUN-PRIORITY parameter should be set to “low”.

3. Increasing throughput of disks and tapes

   Using large tape blocks increases the performance and tape utilization. By default, large tape blocks are written when saving to MTC with FDDRL.

4. Multiplexing with PAV (/390 servers)

   FDDRL generates asynchronous I/O operations when reading from disk (DUMP) or writing to disk (RELOAD). These are executed in parallel when the storage system’s Parallel Access Volume (PAV, 390 server) function is used together with a suitable RAID level (see "RAID levels and their performance").

5. Multiplexing with multiple disks

   In the case of very high-speed tape storage systems, FDDRL can read blocks from up to 4 disks in parallel and write alternately to tape. This process makes sense particularly when disk processing is slower than tape processing. Depending on the tape storage systems available and the size of the tape volume used, the following backup strategies can be employed:

   - Large number of tape drives and small tape volumes

     With ETERNUS CS a large number of tape drives are as a rule available. The volume size is configurable and should be configured in relation to the disk volume size to avoid too high tape VSN consumption or frequent tape changes occurring. Here it is recommendable to use the FDDRL statement //DUMP-PUBSET ..., SAVE-ENTITY=*SINGLE-DISK to back up pubsets.

     One tape set is written for each disk volume. This ensures that maximum performance is achieved. The tape and disk sizes should be coordinated to achieve a good tape occupancy level.

- Few tape drives and large tape volumes

  When LTO drives are used, the following FDDRL statement is recommended: //DUMP-PUBSET ..., SAVE-ENTITY=*DISK-SET(NUMBER-OF-DISK-SETS=n)

  Multiplexing enables multiple disk volumes to be backed up on a small number of tape volumes and tape drives. The number of drives used in parallel can be set using the NUMBER-OF-DISK-SETS parameter. The TASK-LIMIT must be coordinated with this using //MODIFY-FDDRL-PARAMETERS. Since multiple disk volumes are backed up on one tape set, large tape volumes can be used effectively.

  The creation of disk sets using //DUMP-PUBSET also has the advantage that complete and consistent restoration of the pubset is possible with //RELOAD-PUBSET. If disk sets are not used, the pubset must be restored with //RELOAD-DISK. In this case the system administrator has the task of guaranteeing the completeness and consistency of the various restored volumes.

> In addition to the operand values mentioned, TAPE-FORMAT=*STD should also be set (//MODIFY-FDDRL-PARAMETERS). FDDRL then determines the ideal tape format itself. A different format specification is only provided for data interchange with systems using earlier FDDRL versions.

Measurements with FDDRL

The tables below show backup throughputs which can actually be achieved with LTO-6 tape devices under FDDRL on /390 and x86 systems. In each case a DUMP and RELOAD to one volume was performed with files permitting average compression (factor 2-3).

The throughput rates specified refer to one task during the actual backup and restoration phase, in other words without mounting the tape, without starting and ending the programs, and without any other overhead.

The throughputs were measured using a number of configurations, with the following hardware components being used:

- Servers: SE700-20 with 1,120 RPF and SE300-80F with 1,200 RPF
- Storage systems: ETERNUS DX600 and ETERNUS DX440
- SAN connection of the storage systems: 2-3 Fibre Channels operating at 8 Gbit/s
- Volumes: Different-sized NK2 volumes on RAID 1/0 (3+3)
- Tape devices: IBM ULTRIUM-TD6 connected to a Fibre Channel operating at 8 Gbit/s

In practice, the throughputs which can be achieved depend on the sum total of the components involved. No generalization is therefore possible for throughputs in specific individual cases. Instead, the bandwidth of the throughputs measured is specified. The throughputs achieved with servers with /390 and x86 architecture are at a comparable level when the same performance is available.

Throughput with FDDRL in Mbyte/s DUMP	Throughput with FDDRL in Mbyte/s RELOAD
270-310	310-320

> ¡ The measurement results in this section assume favorable conditions, e.g. no parallel load on the measured configuration, sufficient CPU performance.

In order to display the CPU requirement on a comparable basis, the CPU utilizations measured were converted to RPF per Mbyte/s. The CPU requirement with an FDDRL task is between 0.1 and 0.15 RPF per Mbyte/s. Consequently, a fictitious system with 500 RPF, for instance, would have a workload of between 5% and 7.5% with a data throughput of 250 Mbyte/s.

> ¡ Particularly in multiprocessor systems it must be borne in mind that the CPU performance required for a backup task must be provided by just one BS2000 CPU.

The I/O processors of BS2000 servers in no way present a bottleneck.

0.0.0.1 Measurements with FDDRL and ETERNUS CS8200 VTL

The FDDRL backups on Logical Volumes of a ETERNUS CS8200 VTL were performed with the same configuration used in the previous HSMS/ARCHIVE backup chapter:

- Server: SE710-20D (1220 RPF)
- Storage-System: ETERNUS DX8700, 16 Gbit/s FC, 2,5" SSD-M Drives
  Volumes: NK2 format, 2 RAID-Groups (RAID1), 4-channel connection
- ETERNUS CS8200 VTL: FC-connection with 16 Gbit/s via 2 channels, Logical Drives (LD) IBM 03590E1A (BS2000: DEVICE-TYPE=TAPE-C4)

The following throughputs were achieved with DUMP-DISK and RELOAD-DISK with single disk to single LD, 2 disks parallel to 2 LDs, and 2 disks multiplex (MUX) to 1 LD, both with FastDPAV and without PAV:

FDDRL	FDDRL throughput (MB/s)			
	DUMP FastDPAV	RELOAD FastDPAV	DUMP without PAV	RELOAD without PAV
1 disk	297	245	275	202
2 disk	588	464	553	398
2 disk MUX	295	247	294	250

In all cases with FastDPAV as well as with 2 disks and Multiplexing without PAV the virtual drives posed the bottleneck.

VM2000

This chapter contains aspects for the use of VM2000 which are relevant to performance:

- Guest system planning
- Optimizing a guest system
- Measurement tools for VM2000

Guest system planning

This section describes the following aspects of guest system planning:

- Number and multiprocessor level of the VMs
- Setting the CPU quotas
- Recommendations for an optimal configuration

Number and multiprocessor level of the VMs

The total number and multiprocessor level of the VMs affect the VM2000 overhead:

- Total number of VMs and total number of virtual CPUs
  - The more virtual CPUs are started on a real CPU, the less ideal the effect of the CPU caches is.

    By means of CPU affinity (see "VM2000 scheduling (/390 servers)") VM2000 (/390 servers) reduces the number of CPU switchovers in which a virtual CPU is executed on a different real CPU than before. CPU pools enable the number of active virtual CPUs per real CPU to be restricted.
  - The VM2000 hypervisor overhead for managing the CPU queues on /390 servers depends directly on the number of virtual CPUs in a CPU pool.

    > **i** The relation of the number of active virtual CPUs (of the guest systems with production load) to the number of real CPUs should be less than or equal to 3 in all CPU pools. The number of guest systems should be correspondingly low. When the performance requirements for the individual guest systems are lower, a larger number of VMs (up to 15) can also be configured. In all cases it must be checked whether the available main memory is sufficient for the guest systems required.

- Multiprocessor level of a VM (= maximum utilization of the VM)

  The load of the virtual CPUs of a VM has an effect on the utilization level of the time slice and consequently on the frequency of the scheduling operations. A high utilization level of the time slice and a low frequency of scheduling operations are favorable for performance.

  The multiprocessor level of a VM should be oriented according to the peak requirement of the guest system. As low a value as possible should be selected. If the load is low, virtual CPUs in the guest system should be detached.

Setting the CPU quotas

The CPU quotas for the VMs should be set in several stages:

1. Rough setting of the CPU quotas

In order to make a first approximation, it is recommended that you set the CPU quotas in the same proportions as the expected CPU utilization of the individual guest systems.

Example

> Two guest systems (VM1 and VM2) are planned for a biprocessor system. These are estimated to require 50% and 20% of the computer capacity respectively.
>
> This results in the following CPU quotas:
>
> VM1: CPU-Q=50 (standardized EFF-Q=71,43)
> VM2: CPU-Q=20 (standardized EFF-Q=28,57)
>
> Since guest system VM2 will only place a load of around 20% on the server, it should be run as a monoprocessor guest system. Guest system VM2 will certainly require both CPUs.

2. Performance limit

The limit for the CPU performance for a VM (see "VM2000 scheduling (/390 servers)") must be selected so that the utilization is around 75% of the performance limit.

3. Measurement

The VM and the guest system are observed using /SHOW-VM-STATUS and SM2 (see "Measurement tools for VM2000"). The measurement values must be used to check whether the actual CPU time consumption and the performance behavior are in line with expectations.

If the guest systems consume approximately the expected capacity, the settings can be left as they are.

4. Checking the performance requirement

If the guest systems do **not** consume approximately the expected capacity, you must check the actual CPU requirements of the individual guest systems. To do this the guest systems should only be started individually, i.e. without competition from another VM. The guest systems should at least be prioritized one after the other with a very high effective CPU quota.

It can be assumed that the measured VM-ACTIVE percentage quotation corresponds to the actual CPU requirement of the guest system. The CPU quotas can now be recalculated on the basis of these values.

> With regard to the workload placed on the server by all guest systems, it is recommended that you avoid CPU workloads > 75% on /390 servers in OLTP mode.

5. Fine-tuning the CPU quotas

Even after the approximate CPU quotas have been set correctly (i.e. the CPU quotas correspond closely to the capacity requirements of the guest systems), it is possible that the capacity requirements of individual guest systems will deviate sharply from the expected values. Guest systems with a high proportion of I/O operations generally fall below the expected values, whereas guest systems with high CPU requirements generally produce values above those expected.

In addition, the response times of important applications sometimes fail to meet requirements. In cases like this, the CPU quotas must be tuned, i.e. individual CPU quotas must be set higher (“overplanning”) or lower than the recommended values from the approximation.

It is not unusual for the CPU quotas to be corrected by a factor of 2. The performance requirements of the guest systems determine whether corrections are necessary and the extent of such corrections. This phase requires close cooperation with the administrators of the guest systems.

6. Forming VM groups (/390 servers)

VMs can then be combined to form VM groups.

When VM groups are used, fine-tuning of the MEMBER-CPU-QUOTA is also recommended for the group members.

Overplanning of the CPU-QUOTA for the VM group is not recommended. If overplanning is necessary, individual VMs should be planned instead of VM groups.

Recommendations for an optimal configuration

The constantly increasing performance capabilities of the BS2000 servers mean that the timing for scheduling the VM2000 guest systems has also changed. This leads to the following recommendations:

- Reducing the scheduling operations
- Reducing the multiprocessor level
- Using dedicated CPUs
- Adapting CPU pools to the server architecture
- Avoiding shared disks

Reducing the scheduling operations

- In the case of applications which are not CPU-intensive (e.g. OLTP mode or batch mode with a large number of I /O operations of other interruptions), there is a need for a large number of generally only very brief intervals in which CPU performance is required ("CPU time slices").
- When systems are utilized significantly less than 100%, normally only a few of these short CPU time slices are used immediately one after the other; the CPU then switches to the wait state (idle).
- In the case of TP loads severe fragmentation of the CPU requirement into small CPU time slices must therefore be reckoned with, as well as frequent, brief wait states of the CPU.
- The length of these CPU time slices decreases as the CPU performance increases (faster processing) and as the multiprocessor level increases (shorter queues).

As a result of this, the number of VM2000 scheduling operations rises as the CPU performance increases and as the multiprocessor level of the VMs increases. The increased throughput and reduced response times thus lead to a slight increase in the VM2000 overhead.

For OLTP applications it is therefore recommended that the priority controls between the DB server and DB client tasks be set in such a way that the job log of the DB server tasks is always full. This avoids overhead through additional communication.

Reducing the multiprocessor level

Keep the number of the VMs' virtual CPUs (the multiprocessor level) as low as possible (see section "Number and multiprocessor level of the VMs"):

- Specify the multiprocessor level of a VM. Do not simply use the multiprocessor level of the server.
- When selecting the multiprocessor level of a VM, ensure that it is only large enough to permit the required (peak) load to be processed.
- If possible, distribute the load over more than one VM with lower multiprocessor levels.
  In measurements on a /390 server with 4 CPUs a load was initially distributed over 2 guest systems each with 4 CPUs, and then over 4 guest systems each with 2 CPUs. It became apparent that the load distribution over 4 guest systems each with 2 CPUs caused around half the VM2000 overhead.
- Do not start any guest systems "in reserve". Such guest systems cause VM2000 overhead even when they bear no load.
- You can configure guest systems with an appropriate multiprocessor level to handle load peaks. However, detach the CPUs in the guest system in times when there is no load peak (/DETACH-DEVICE).

Using dedicated CPUs

On /390 servers also make use if possible of the option of dedicated CPUs (see section"VM2000 scheduling (/390 servers)"):

In VM2000 operation without additional CPU pools, all available real CPUs are used for scheduling the (virtual) CPUs of all guest systems. For systems for which particularly high performance requirements apply (e.g. productive systems in contrast to test or development systems), it is consequently recommendable to use a mode which is very similar to native mode.

- Make as many real CPUs available to such systems as there are (in total) attached virtual CPUs in the guest systems concerned.

The result of this is that each virtual CPU has its own real CPU to use exclusively ("dedicated CPU"). This prevents unnecessary overhead resulting from avoidable cache losses and wait times for a free real CPU or multiprocessor locks.

The following configuration, for example, is suitable for this purpose:

- In addition to the standard CPU pool, configure another CPU pool for critical productive systems with dedicated real CPUs.All other guest systems use the standard CPU pool with the remaining real CPUs (or additional CPU pools which they each share).
- On /390 servers, also use the VM attribute VM-ACTIVE-IDLE for VMs with dedicated CPUs.

  This function prevents virtual CPUs in the wait state (idle) from leaving the real CPU. There is hardly any hypervisor overhead.

  A measurement on a /390 server with 4 CPUs and only one VM showed that when dedicated CPUs and the VM attribute VM-ACTIVE-IDLE are used, the VM2000 overhead is reduced by 75% compared to when dedicated CPUs are used **without** the VM attribute VM-ACTIVE-IDLE.

  > **i** No evaluation of the VM-ACTIVE times (/SHOW-VM-STATUS or VM report of openSM2) is possible for VMs with VM-ACTIVE-IDLE.

Adapting CPU pools to the server architecture

When setting up CPU pools, pay attention to the physical server architecture.

Larger servers with a higher multiprocessor level may have multiple processor modules. These generally have their own cache. In NUMA architectures, the main memory is separated into several domains, which are each assigned to a specific processor module from an administration perspective. Memory access times from a CPU core to a memory module assigned to its “own” processor module is more performant than access to “more remote” memory modules.

If a task or a VM moves between different processor modules during processing, the performance is impaired by cache losses and memory access times. When using CPU pools, you have the opportunity to reduce these effects:

- Only assign CPUs from the same processor module to your CPU pools:

  In an optimum situation, each **CPU pool** will be limited to just **one module**.
  However, such strict delineation is not always possible, e.g. as soon as a CPU pool has to have more CPUs than a single module provides due to the performance requirements of a VM. You should then attempt to distribute your CPU pools among the CPUs in such a way that
  - at least the most performance critical VMs can use as many CPUs as possible from a particular module.
  - as few CPU pools as possible are split between multiple modules.

The general conditions to be considered depend on the specific server unit:

SU /390

An SU /390 has one or two system boards. These each house a processor module with up to 16 BS2000 CPUs and the associated IO processors and main memory.

- An SU710 or SU730 has one to two system boards.

  Up to and including the respective -70 models, all CPUs are located on one board, which means there is nothing to consider.

  The models -100 to -160 have two system boards. The first 8 CPUs are located on the first board and all others on the second board. Ideally, your CPU pools should therefore only include either the CPUs 0-7 or 8-15.
- An SU740 has one system board.

  On this system board, there is one processor containing 16 cores. However, this processor is internally split into to sections with 8 cores each. Considering performance, this causes effects similar to the older models with two system boards:

  Up to and including SU740-80 there is nothing to consider.

  On the models from SU740-100 to SU740-160, your CPU pools should ideally only include either CPUs 0-7 oder 8-15.

> ¡ A detailed description of CPU pool administration can be found in the “VM2000 user manual” [34 (Related publications)].

SU x86

An SU x86 has two or four Intel processors. Each of these correspond to a processor module. To guarantee thorough separation, CPU pools exist at several hierarchical levels, some of which cannot be influenced by configuration settings.

More in-depth details can be found in the “ Fujitsu SE Server - Administration and Operation” manual[4], “Visualization on x86 server unit” section. But for the understanding in this context , only the following information is relevant:

- Each physical CPU (corresponding to an Intel core) is pre-assigned to one of two internal system pools (X2000, BS2000). This classification cannot be influenced.
- The pool containing all BS2000 CPUs is located in one or two process modules depending on the server model. With VM2000, a further subdivision can be carried out using VM2000 CPU pools.

Therefore, from a VM2000 perspective, the assignment of the physical CPUs follows essentially the same logic as for /390 servers, i.e. the first BS2000 CPU is always assigned to the first CPU of a particular module. If there are more BS2000 CPUs than a module provides, all remaining CPUs are located on the other module. The following applies here:

- An SU310 has Intel processors, each with 16 cores. Therefore, all BS2000 CPUs are located in the same module. Although there is nothing to consider concerning the server architecture, you are advised to configure CPU pools only as large as necessary.
- An SU330(B) has Intel processors, each with 16 cores. Therefore, all BS2000 CPUs are located in the same module. Although there is nothing to consider concerning the server architecture, you are advised to configure CPU pools only as large as necessary.
- An SU340 has Intel processors, each with 16 cores. Therefore, all BS2000 CPUs are located in the same module. Although there is nothing to consider concerning the server architecture, you are advised to configure CPU pools only as large as necessary.

Avoiding shared disks

With SU /390 avoid 'shared' disks as much as possible.

Refer to the information on exclusive and "shared" assignment in section "Peripherals".

Measurements have shown that with more than one guest system the VM2000 overhead increases by 0.4% for every 100 I/O operations per second on disks which are assigned as “shared”.

- Limit the use of disks assigned as “shared” under VM2000 to the necessary degree.

Optimizing a guest system

VMs and guest systems with batch-oriented, CPU-intensive load are preferred by VM2000 scheduling. Dialog-oriented guest systems with frequent idle times are hampered in their response times by the IO increase. If necessary fine tuning of the CPU quotas must be performed, see "Setting the CPU quotas".

The task priorities in a dialog-oriented guest system have less effect. If required, the priorities must be set anew, see the following section.

A guest system that only makes little use of its time slice is hampered in comparison with native mode by the I/O increase on the one hand and the worse cache hit rates on the other. In particular on /390 servers, execution in a CPU pool with dedicated CPUs and the VM attribute VM-ACTIVE-IDLE can help, see "VM2000 scheduling (/390 servers)".

When PCS is used the increase in a category can be influenced in both directions, see section "Using PCS (Performance Control System)".

Setting priorities

If you distribute an IT workload across several different guest systems, the degree of concurrency of the individual components of the load changes. This means that priorities sometimes need to be changed. The following example illustrates this.

The following load components are running on a server in native operation:

- one UDS/UTM application
- multiple batch database applications
- a number of dialog tasks involving program development with no access to the databases

Up to now priorities and CPU utilization have been as follows:

	UDS tasks	UTM tasks	Batch	Dialog1	Dialog2
Task number	1	8	2 - 6	3	10
Priority	150	150	255	180	210
Workload (%)	20	30	10	5	10

The IT load is now to be distributed over two guest systems as follows: The developer activities are to be outsourced to a separate guest system. All other components should run on another guest system on account of their shared databases

VM	Name	CPU quota	Load
VM1	TP system	80	UDS/UTM + batch
VM2	Dialog system	20	Dialog1 + Dialog2

The fact that the dialog load has been relocated means that the situation for batch tasks improves, since these can now always run when the TP load is waiting for inputs/outputs. The throughput of CPU-intensive batch tasks improves, since they tie the real CPU to the guest system.

Systems used purely for dialog tasks generally have a high proportion of idle time. The low CPU quota means that the dialog system will have to wait for a relatively long time after each idle state. This means that the response times for the dialog tasks may increase.

The preference given to dialogs with a high priority compared with those with a low priority also decreases. The following measures are necessary if the same results are to be maintained after the load has been split:

- If the dialog response times increase dramatically, the CPU quota of the dialog system must be raised. The TP response times and the batch throughput must be constantly monitored.
- The difference in priorities between Dialog1 and Dialog2 must be increased.
- The priorities of the UDS and UTM tasks must be increased in order to restore the original difference to the batch tasks, which now have a greater priority.

Using PCS (Performance Control System)

If PCS (Performance Control Subsystem) is running under VM2000, the CPU quota of the corresponding guest system is taken into account internally. In other words, within a guest system the sum of the S-Q MAX value for response time-critical categories should be 100 (corresponds to 100% of the guest system capacity).
Compared to native operation, a correction generally needs to be made to the parameters of the SERVICE ranges (range between SERVICE-QUOTA MIN and MAX, controlled by the dilation parameters REQUEST-DELAY MIN and MAX).

The dilation (REQUEST-DELAY) is defined as follows:

Dilation = Runtime in multiprogramming mode / Single runtime

Under VM2000, it is possible that both runtimes will increase. The single runtime can increase, for instance, as a result of increased I/O times caused by other guest systems or as a result of a reduction in hardware performance because of a lower cache hit rate.

Depending on the load profile of the individual categories in a guest system and the load profiles of the other guest systems, it is possible that the PCS dilation for individual categories may increase in comparison with native mode (which is to be expected) or decrease.

Checking or correction of the SERVICE-QUOTA and REQUEST-DELAY parameters should be done in the following sequence:

- SERVICE-QUOTA MIN
  This sets the service quota a category requires in normal operation, i.e. without taking workload peaks into account.
- SERVICE-QUOTA MAX
  This sets the maximum percentage of the capacity of the guest system which can be assigned to a category during peaks in the workload.
- REQUEST-DELAY MIN
  The system administrator should take a series of measurements to identify the value for REQUEST-DELAY ACTUAL as of which a bottleneck or unsatisfactory behavior in a category arises, i.e. the value at which the service rate should be increased. REQUEST-DELAY MIN is then set to this value.
- REQUEST-DELAY MAX
  This defines the degree of response to a workload peak. The nearer this value is set to that of REQUEST-DELAY MIN, i.e. the smaller the value, the stronger the reaction to peaks in the workload.

Operation via KVP

Using Virtual Consoles via $VMCONS is generally not recommended any more, not only because of their higher VM2000 overhead. KVP Consoles are the preferred way of operating your guest systems.

Measurement tools for VM2000

Two measurement tools are available for VM2000 operation:

- In the monitor system /SHOW-VM-STATUS performs measurements at VM2000 level for VMs, VM groups, CPU pools and real CPUs. For /390 servers, here some internal VM2000 performance values are also output. The hypervisor activities, planned and actual loads for CPU utilization and VM2000 scheduling data can be monitored. This command can be used to output both periodic measured values and also measured values from the immediate past.
- The openSM2 software monitor with online reports (INSPECTOR) and long-term analyses (ANALYZER).

The following ANALYZER report groups are recommended for observing the utilization of the guest systems and of the hypervisor (/390 servers) using openSM2.

- In the guest systems

  CPU: CPU utilization
  CPU utilization in the processor states TU, TPR, SIH. These values are the utilization values determined by the guest system concerned (in relation to the number of virtual CPUs available on the guest system!).
- In the monitor system

  VM2000: VM2000 utilization/VM2000 hypervisor
  CPU utilization, current CPU quota and current maximum performance utilization per VM. CPU consumption of the hypervisor and hypervisor idle (/390 servers).

  VM2000: CPU pool
  CPU utilization and dimensioning of the CPU pools.

  VM2000: VM groups (/390 servers)
  CPU utilization, current CPU quota and current maximum performance utilization per VM group.

> ¡ Some SM2 values must be interpreted differently for VM2000 mode. Refer to the section entitled “SM2 together with VM2000” in the “openSM2” manual [18 (Related publications)].

System and application control

This chapter describes the following topics concerning system and application control:

- System parameters relevant to performance
- Size of the user address space
- Performance criteria when creating files
- Working with HIPERFILEs and DAB
- Optimizing an OLTP application
- High-performance loading of programs

System parameters relevant to performance

With the help of the BS2000 system parameters (which are defined in the SYSOPT-CLASS2 parameter set of the SYSPAR.BS2.version parameter file), the system behavior can be adjusted to suit the relevant requirements.

This section describes the following system parameters which are relevant for performance:

- BMTNUM and CATBUFR
- CONSDDE7, EACTETYP, L4MSG, MSGCENTL, MSGCENTN, MSGDLAM, MSGFILii,MSGNOFL
- DEFLUID
- DESTLEV
- DMPRALL, DMSCALL, DMMAXSC
- EAMMEM, EAMMIN, EAMSEC, EAMSIZ
- ETMFXLOW
- L4SPDEF
- SSMAPRI and SSMASEC
- TEMPFILE

For a complete list of parameters with detailed descriptions of each, see the te command SHOW-SYSTEM-PARAMETERS in the manual “BS2000 - Commands (volume 1 - 7) [15 (Related publications)].

BMTNUM and CATBUFR

Management of and accesses to the file catalog TSOSCAT are carried out internally by the Catalog Management System (CMS). For a functional description and further suggestions for tuning, see section "Creating system files".

The system parameters BMTNUM and CATBUFR should always be considered together. When deciding which values to assign to these parameters, you must take into account the number of pubsets with which the system is to be run (cf. CATBUFR).

Parameter name	Value range	STD value
BMTNUM	0 ... 255	32

This parameter defines the number of I/O buffers and Buffer Management Tables (BMT) for the catalog management system (CMS). This number of buffers is created once for each pubset and once globally for all private disks and Net Storage volumes.

Each block of the TSOSCAT file can contain max. 13 file entries or 18 JV entries. For each CMS I/O buffer (4096 bytes für pubsets, 2048 bytes for private deisks and Net Storage Volumes) a table (BMT) is created (approx. 178 bytes). It is used for managing the current status of the block in the buffer.

The default value for BMTNUM is 32. For pubsets, this is also the lowest possible buffer number. Values from 0 to 31 are only applied for private disks.
Each increase also increases the probability that CMS will find a file entry or JV entry without a physical input operation being carried out. The positive effect of this strategy is particularly noticeable in the case of frequent access to adjacent catalog entries and thus to one and the same catalog block.

Parameter name	Value range	STD value
CATBUFR	N/Y	N

If this parameter is set to Y, the CMS buffers and the BMTs will be created as resident (in class 3 memory).
If it is set to N (default), they will be created as pageable (in class 4 memory).

The pubset-specific setting of the number of CMS buffers and their memory class is carried out using the commands

```
/ADD-MASTER-CATALOG-ENTRY
/MODIFY-MASTER-CATALOG-ENTRY
/IMPORT-PUBSET
```

and the operands NUMBER-OF-BUFFERS and RESIDENT-BUFFERS.

Defined values are sought in the following order:

1. Explicit parameter specification in /IMPORT-PUBSET
2. Specification for /ADD-MASTER-CATALOG-ENTRY or /MODIFY-MASTER-CATALOG-ENTRY
3. Specification via the system parameters BMTNUM and CATBUFR

If neither private disks nor Net Storage volumes are used, buffer allocation for private disks can be prevented with BMTNUM=0. For Pubsets, the minimum number of 32 buffers stays effective. With /MODIFY-MASTER-CATALOG-ENTRY (for home and paging pubsets) or /IMPORT-PUBSET the number of CMS buffers and their storage class can also be specified differingly.

The size of the address space requirement per pubset or of the private volumes for the CMS buffers can be calculated using the following formula:

CMS buffer (bytes) = (x + y) * Number of buffers + 384

x = 4096 for pubsets; x = 2048 for private volumes

y = 178 for pubsets; y = 200 for private volumes

If, for example, with smaller systems a large number of memory-resident buffers are created, the catalog operations will be faster but this advantage may be offset by an increased paging rate. If the catalog is accessed frequently, use of the software product SCA (Speed Catalog Access) is imperative.

CONSDDE7, EACTETYP, L4MSG, MSGCENTL, MSGCENTN, MSGDLAM, MSGFILii, MSGNOFL

These parameters control message output and consequently belong together:

Parameter name	Value range	STD value	Meaning
CONSDDE7	N/Y	N	Defines whether the message `DMS0DE7 SAM FILE CLOSED` is also to be output on the console: Y Output on the console and to SYSOUT N Output only to SYSOUT *Note:* If a magnetic tape is assigned as the output device for SPOOL, this message will appear at the end of every SPOOL output.
EACTETYP	0/1/2/3	0	Defines which of the following messages are to be output via the console: `BLS0500`, `BLS0517`, `BLS0519, BLS0523, BLS0524, BLS0526, BLS0539`, `BLS0551`, `BLS0552`. 0 None of the messages 1 Only message `BLS0519` 2 All the messages specified above 3 All the messages except `BLS0519`
L4MSG	0/1	0	Controls the output of the question `EXC044E "ACCEPT ALLOCATION REQUEST..."`. 0 Message is not output 1 Message is output
MSGCENTL	36...2500	200	Defines the length of an entry in class 4 memory for the message processing. The value must be a multiple of 4. This area of the class 4 memory is used to buffer the most recently and frequently used messages in order to save on file accesses
MSGCENTN	0...32767	32	Defines the number of entries in class 4 memory (see above) for the message processing. *Note:* The class 4 memory requirement can be calculated by multiplying the values of MSGCENTL and MSGCENTN
MSGDLAM	0...99	0	Number of message files to be processed via DLAM

MSGFIL01 MSGFIL02 MSGFIL03 MSGFIL04 : MSGFIL15 *)	filename(n)		Number and names of the message output files. The following default values apply for the MSGFIL01 and MSGFIL02 parameter: MSGFIL01: `SYSMES.BS2CP.version` MSGFIL02: `SYSMES EKP.01` There are no standard names for the parameters MSGFIL03 through MSGFIL15, i.e. these files are not created. The file name must be fully qualified and can only be cataloged with the user ID $TSOS.
MSGNOFL *)	0...15	2	Number of message files that were specified via the MSGFILxx parameter.

*) for details see the section "Message files"

DEFLUID

With BS2000 V21.0B the default user ID can no longer be changed. If the default user ID is changed in the parameter file it will automatically be changed back to $TSOS and the system will inform the operator with a console message.

DESTLEV

Parameter name	Value range	STD value
DESTLEV	0/1/4/5/6	0

DESTLEV defines whether file extents are to be overwritten with binary zeros when released from the system (/DELETE-FILE or /MODIFY-FILE-ATTRIBUTES with memory release).

With large files this can be a very lengthy process (lasting minutes).

DMPRALL, DMSCALL, DMMAXSC

These three system parameters must be defined in coordination with each other. They are used as defaults for the space allocation when creating and extending files. The values apply to all private volumes. For pubsets they only apply if nothing else is defined in the pubset's MRSCAT entry. In addition, the allocation unit specified when the pubset was created with SIR is considered depending on the data format (K, NK2, NK4).

Defined values are sought in the following order:

1. Direct specification in the /CREATE-FILE or /MODIFY-FILE-ATTRIBUTES command
2. Entry in MRSCAT
   - If specified with /MODIFY-MASTER-CATALOG-ENTRY (ALLOCATION operand), the modification will not take effect until after the next import of the pubset.
   - If /MODIFY-PUBSET-SPACE-DEFAULTS (operand PRIMARY-ALLOCATION / SECONDARY-ALLOCATION / MAX-ALLOC) is used, the definition takes effect immediately.
3. Specification via system parameters

If the values ascertained in this way are not a multiple of the allocation unit defined when the pubset was configured using SIR, they are rounded up to the next multiple.

Primary allocation

Parameter name	Value range	STD value
DMPRALL	3...65535	3

Primary allocation for a file in PAM blocks if the SPACE operand was not given a value in /CREATE-FILE oder /MODIFY-FILE-ATTRIBUTES (or in the FILE macro).

Secondary allocation

Parameter name	Value range	STD value
DMSCALL	3...65535	3

Secondary allocation for a file in PAM blocks if this was not supplied in the SPACE operand in /CREATE-FILE oder /MODIFY-FILE-ATTRIBUTES (or in the FILE macro).

After each file enlargement, the amount of the increase is doubled as compared to the value at the previous enlargement. Secondary allocation (SA) at the (i+1)th enlargement equals twice the secondary allocation at the i-th enlargement: SA(i+1) = 2 * SA(i)

This formula continues to be used until double the value of the current secondary allocation exceeds the value DMMAXSC. Thereafter the formula SA(i+1) = SA(i) applies.

Maximum value

Parameter name	Value range	STD value
DMMAXSC	3...65535	48

When double the value of the current secondary allocation exceeds the value of DMMAXSC, the value for the secondary allocation is no longer modified.

Example

File enlargements with the values: DMPRALL 15, DMSCALL 3, DMMAXSC 48. In this example, DMMAXSC could contain any value from 48 to 96 without the behavior changing.

- 15+3=18 (1st file enlargement)
- 18+6=24 (2nd file enlargement)
- 24+12=36 (3rd file enlargement)
- 36+24=60 (4th file enlargement)
- 60+48=108 (5th file enlargement)
- 108+48=156 (6th file enlargement, no change to the secondary allocation because the value would then be greater than DMMAXSC))

> ¡ When a file is created on disk or its size changes, within DMS the smallest amount that can be changed is an allocation unit.
> The smallest allocation unit is 3 PAM blocks for private volumes and for pubsets 3 PAM blocks (K and NK2 format) or 4 PAM blocks (NK4 format).

Each dynamic file enlargement leads to a series of management I/O operations in the system (updating the entries in the user ID, the file catalog, the F1 label with private volumes) and thus incurs a considerable system overhead. To minimize this overhead, the DMSCALL parameter should be set to at least 4 allocation units.

To reduce the number of dynamic file enlargements (or avoid them), the operand SPACE=*RELATIVE(PRIMARY-ALLOCATION=xx,SECONDARY-ALLOCATION=yy) is available to all users in /CREATE-FILE oder /MODIFY-FILE-ATTRIBUTES.

> ¡ Every user should try to give the file its expected final size already when creating it, by means of a corresponding primary allocation.

Example

A program writes 456 blocks sequentially with the PAM macro call. The AINF macro call determines the number of I/O operations.

/CREATE-FILE ... , SPACE=REL (...)		**No. of physical I/Os according to AINF**	**No. of extents acc. to /SHOW-FILE-ATTR**
PRI-ALLOC=	**SEC-ALLOC=**		
3	3	480	20
6	6	476	18
30	30	467	9
192	192	459	3
456	30	456	1

The results can vary depending on the respective disk occupancy levels.

EAMMEM, EAMMIN, EAMSEC, EAMSIZ

Parameter name	Value range	STD value
EAMMEM	0...2730	0
EAMMIN	4...64512	3000
EAMSEC	1...64512	200
EAMSIZ	4...64512	64512

These parameters define the size of the SYSEAM file. They are valid for all :catid:SYSEAM files if they are created on several pubsets and if nothing else is defined in the MRSCAT entry for each pubset. In addition, the allocation unit specified when the pubset was created with SIR is considered depending on the data format (K, NK2, NK4).

The SYSEAM file is managed in EAM allocation units. When a DMS allocation unit is 3 PAM pages, the EAM allocation unit is also 3 PAM pages; in the case of all other DMS allocation units the EAM allocation unit is 4 PAM pages. The value of EAMMIN defines the minimum size. If necessary, it is extended by the value of EAMSEC (number in EAM allocation units) until the maximum size of 64512 EAM allocation units is reached. Should the SYSEAM file shrink (e.g. in the case of /DELETE-SYSTEM-FILE SYSTEM-FILE=*OMF or /EXIT-JOB), it is reduced in increments of 8 allocation units, but does not go below the value of EAMMIN.

The value of EAMMIN must be set such that in normal operation 80% of all requests to save blocks in the system file SYSEAM can be satisfied without it being necessary to enlarge the file. Experience shows that a size of 4000 allocation units for the initial setting is advisable.

If the value chosen for EAMMIN is too low, this will lead to a higher load on the system due to many dynamic file enlargements and uncontrolled growth of the SYSEAM file.

By choosing suitable values for EAMMIN and EAMSEC or the EAM operand (MINIMAL-SIZE, SECONDARY-ALLOCATION) in /ADD-MASTER-CATALOG-ENTRY und /MODIFY-MASTER-CATALOG-ENTRY, you can keep the variation of the file size of SYSEAM to a minimum or suppress it completely.

It is important to keep the number of required secondary allocations per pubset as low as possible.

EAMSIZ is the maximum value in allocation units within the $TSOS.SYSEAM file that are available to one user alone.

Standard values for the EAM system parameter are heavily dependent on the expected load. The values for EAMMIN and EAMSEC should both be set to a multiple of 8, as this produces optimum adjustment to the internal table structure of EAM.

The value of EAMMEM defines the size of the class 4 memory (in units) used for EAM. It should be a multiple of 8. The pubset-specific definition is implemented using the EAM=*PARAMETERS(VIRTUAL-MEMORY=...) operand in /ADD-MASTER-CATALOG-ENTRY or /MODIFY-MASTER-CATALOG-ENTRY.

This area is created for the home pubset only and is part of the SYSEAM file. First the class 4 memory area is filled, and only once it is full does writing to disk begin (no cache).

Beneficial use (speeding up compilation runs by saving on I/O operations) is only possible if there is sufficient main memory available. Otherwise the benefit of cutting down on physical I/Os on the SYSEAM file will be offset by increased paging of the class 4 memory pages.

ETMFXLOW

Parameter name	Value range	STD value
ETMFXLOW	127...256	256

Tasks with an external priority greater than or equal to ETMFXLOW are not affected by aging in processor state TU. This means that for these tasks the dynamic setting of the internal priority (see section "Prior concept") is deactivated. Any increase in priority acquired in processor state TPR becomes invalid on transition to TU.

Tasks with an external priority greater than or equal to ETMFXLOW are suitable for definition of a background load. For the main application(s) a correspondingly large external priority difference must be selected. This function must be used very carefully and its effects must be monitored.

L4SPDEF

Parameter name	Value range	STD value
L4SPDEF	66...n	2500

The L4SPDEF parameter determines the limit value for achieving saturation level 4 in pubsets. The level is regarded as achieved when fewer PAM blocks are free than defined here. The default value is recommended.

The value is valid only as long as no other pubset-specific definition is specified using /ADD-MASTER-CATALOG-ENTRY or /MODIFY-MASTER-CATALOG-ENTRY with the ALLOCATION=*PARAMETERS(SATURATION-LEVEL4=...) operand.

0.0.1 NBESSIZE

Defines the maximum size in PAM pages of the central system event stream file $SYSAUDIT.SYSLOG.ESS.SYSTEM.

If the system event stream service is not used and not needed, the value can be set to 0. In that case, the file won't not created and the system event stream services will not be available.

SSMAPRI and SSMASEC

Parameter name	Value range	STD value
SSMAPRI	3...65535	24
SSMASEC	3...65535	24

These parameters define the size of the primary and secondary allocation for system output files. The parameters are valid for files created with the following commands:

`/ASSIGN-SYSOUT`

`/ASSIGN-SYSLST`

Whenever one of these system output files is used for the first time, a cataloged file is created with the size defined by SSMAPRI and SSMASEC. The file names are:

`S.OUT.tsn.yyyy-mm-dd.hhmmss.cnt`

`S.LST.tsn.yyyy-mm-dd.hhmmss.cnt`

SSMAPRI

> Primary allocation in PAM blocks for the task-specific spoolout files or for reassigned system files which were created with an ASSIGN ... command.

SSMASEC

> Secondary allocation in PAM blocks for the task-specific spoolout files or for reassigned system files which were created with an ASSIGN ... command.

The optimum values for primary and secondary allocation of these SPOOL output files are heavily dependent on the type of the load. Unless experience indicates otherwise, the default values of 24 PAM blocks for each should be used.

TEMPFILE

With BS2000 V21.0B the system-parameter value for TEMPFILE is fixed to '#'. If the value in the parameter file is changed to 'NO' or '@', the system will automatically change it back to '#' and inform the operator with a console message.

Size of the user address space

User address space for x86 servers

The user address space for x86 servers is generated with a size of 2 GB and cannot be modified.

User address space for /390 servers

For /390 servers the total address space consists of the user address space (including the space requirement for shared products; see the SHRSIZE parameter) and the system address space. For the size of the total address space on /390 servers there is a choice of the values 1 or 2 GB.

On S servers the size of the user address space automatically determines the size of the system address space as the difference to the next higher value of the total address space. It is no longer necessary to specify the SYSSIZE parameter in the parameter service during system startup.

By default, the user address space for /390 servers is defined with a size of 1,808 MB. This size can be changed using the SYSPRC.BS2000-EXEC.<version> procedure (see the “System Installation” manual [32 (Related publications)]).

With configurations which require a larger system address space, the user address space must be reduced correspondingly (e.g. to 1,792 MB when SYSSIZE=256 MB, with all intermediate values being possible from 240 through 512 MB at intervals of 16 MB).

One of the reasons the subdivision of the total address space was introduced was to reduce the main memory requirement for the address space management (segment table). Depending on the size of the total address space (1 or 2 GB), the following main memory per task is occupied for the segment tables:

- 4 KB for an address space of 1,024 MB
- 8 KB for an address space of 2,048 MB

With a small number of tasks, selection of the largest possible address space (2 GB) is not critical. With a large number of tasks, you must check whether the extra main memory required by some large applications is justified.

Page-table management requires 3 KB main memory per megabyte of occupied address space and task. If, for example, 10 tasks completely occupy the maximum available user address space of 1,808 MB, the following main memory (resident class-3 system address space) is required for managing the address space alone (segment table and page table)

10 * 8 KB = 80 KB for segment table

10 * 1.808 MB * 3 KB/MB = 54.240 KB or page table

Sum 54.320 KB

With large virtual address spaces, you must make sure that the paging areas are large enough. The user-specific size (/ADD-USER, /MODIFY-USER-ATTRIBUTES) must be defined carefully. If many large address spaces are to be used, a correspondingly large sized-main memory is required.

Performance criteria when creating files

BS2000 users and system administrators can generally only influence the placement of the files with respect to logical volumes. At physical level a number of logical volumes can be located on the same physical disk. The resultant effects on performance and the performance measures are described in section "Disk organization and accesses from the BS2000 viewpoint".

Information on the performance of Net-Storage is provided in section "Net-Storage".

In the following sections in this chapter the (previous) term “disk” is still used synonymously for the (current) term “logical volume”.

> i As a rule the explicit definition of a file location is not to be recommended. In unfavorable cases this can even have a negative influence on the performance.

Logical operating modes of disks

In BS2000, disks are addressed logically via the volume serial number (VSN). Volumes can be operated as:

- Public volumes: Public volumes are operated in SF pubsets (Single-Feature pubsets), SM pubsets (System-Managed pubsets) or shared pubsets.
- Private volumes

Irrespective of which logical operating mode is in use, the time taken to perform a single data access operation is approximately the same (if the disks are of the same type).

There are, however, serious differences regarding the number of I/O operations to be performed when calling functions which require catalog access. This is due to the differing degrees of complexity with regard to memory management or the co-ordination of multiprocessor operation.

To increase availability, the following operating modes are also provided:

- DRV: Dual Recording by Volume: Data mirroring controlled by BS2000 (software product DRV, for details see the "DRV" manual [7 (Related publications)])
- Replication: volume-based mirroring for storage systems
  Disk mirroring, controlled by the ETERNUS DX/AF storage systems. In the ETERNUS DX/AF and Symmetrix storage systems, the RAID-1 level is available on the hardware level. Dual data recording takes place without the need for any further software support.

Public volumes

Memory management is performed by the system and offers optimal support for calls using catalog access:

- a high degree of user convenience by means of considerable simplification of the definitions for creating, extending and deleting files
- a lower number of management I/Os compared with private volumes (management of F5 labels in main memory, support from the software product SCA, see "Accelerating catalog accesses with SCA").

The location of a file can be specified explicitly for all pubsets by systems administration. and for selected pubsets by the user as well, where the authorization for physical allocation is entered in the pubset-specific user catalog (/MODIFY-USER-ATTRIBUTES, PHYSICAL-ALLOCATION operand).

Pubsets are the strategic platform for further developments in the disk area. Private volumes do not offer full functional support of new functions, e.g. in the HIPERFILE concept.
Migration from private volumes to pubsets is thus recommended.

SF pubsets

As far as memory management is concerned, the same applies as for public volumes. In addition to an increase in availability, the following advantages can be seen as far as performance is concerned (particularly important for interactive mode):

- Splitting the file catalog TSOSCAT over separate pubsets improves catalog performance if the catalog functions are used intensively.
- It is possible for systems support to control input/output distribution by means of the user catalog entries. Users with files subject to particularly frequent access can be transferred to separate pubsets.

SM pubsets

SM pubsets consist of a number of volume sets that may have different formatting, availability and performance attributes specified by systems support.

In comparison to SF pubsets, which consist of a single volume set, SM pubsets offer better control of input/output distribution to systems support and greater convenience to users:

- By specifying direct attributes or a storage class when creating a file, a user can specify where the file is to be stored on an SM pubset.
- Systems administration assigns a volume set list to each storage class and defines the desired performance requirements for each volume set.

You will find detailed information on SMS (System Managed Storage) and SM pubsets in the “SMS” manual [33 (Related publications)].

Shared pubsets

Two or more write-authorized servers (or guest systems under VM2000) can access shareable pubsets simultaneously (except in the case of the home and paging pubsets). The systems are interconnected via a BCAM link and work according to the master/slave principle. All catalog operations (RDCAT, WRCAT, etc.) are always performed by the master system. It is thus only practical to use SCA in the master.

Taking into account the following points, the performance behavior is virtually the same as for “normal” public volumes:

- the system from which most accesses are performed is set as the master (to minimize cross-system actions)
- to keep the overhead for cross-system actions to an acceptable level, no more than two OPEN/CLOSE operations per second should be carried out on the slave systems for each RPF (Relative Performance Factor).

> Files on shared pubsets can be supported from any system with DAB and local caching.

Private volumes

Memory management is the responsibility of the user. The user can avoid access conflicts in the case of files which are used regularly by specifying the exact location of the files in the relevant volumes.

The number of management I/Os in the case of file processing calls with catalog access is considerably higher than with public volumes owing to additional accesses to the VTOC (Volume Table Of Contents) area. There are, for instance, 6 logical accesses to the F1 labels per OPEN/CLOSE pair, each of which requires one or more physical I /Os.

Thus efficient performance behavior can only be achieved if the number of management I/Os is small compared with the number of “productive” I/Os.

DRV: Dual Recording by Volume

DRV increases the availability of information stored on disk by recording the data on 2 physical disk drives. The duplication takes place in the IO system of BS2000. For a synchronous write operation, first of all an asynchronous output to the first disk is initiated, then a synchronous output to the second disk drive.

The disk drive with the shorter queue is selected when a read operation is to be performed, thus resulting in an unsymmetrical load on the original disk and mirror disk. Selecting the least heavily used disk drive reduces the I/O time (software duration) for read accesses in comparison to SRV (Single Recording by Volume) mode.

Approximately the same performance behavior can be expected as for SRV mode.

Copy speed

The copy speed can be set dynamically in three steps for each pubset using the operand COPY-SPEED=*LOW /*MEDIUM/*HIGH of /SET-DRV-PARAMETER at both restoration and equalization.

With COPY-SPEED=*LOW, the DRV copy operations are slowed down considerably so that the pubset for the I/O operations of applications is more accessible. The default is *MEDIUM.

Extent of copying

Not all the disk's blocks are copied at restoration; only the blocks identified as occupied in the F5 label are copied. After a system failure or an abnormally terminated export of a pubset, DRV carries out equalization in order to match the disk pairs again.
In addition to the existing variants of copying either all the blocks or only the system data, DRV also allows only the system data and the open files to be equalized. This option is only possible for SF pubsets or volume sets of an SM pubset. It can only be set by the following command before the pubset or volume set is imported:

```
/SET-DRV-PARAMETER UNIT=*PUBSET(...)/*VOLUME-SET(...), EQUALIZE-DATA= *OPEN-FILES
```

Rapid restoration after separate processing

In the case of separate processing (e.g. backup of the original data, batch processing of the mirror data), DRV remembers the changed areas. This allows the restoration time to be dramatically reduced when only the changes since the beginning of separate processing have to be copied (since this can be measured in terms of Mbytes) rather than all the files (Gbytes of data).

Accelerating catalog accesses with SCA

SCA (Speed Catalog Access) is a recommended software product for accelerating catalog services.In searches for file entries and job variable entries, SCA replaces sequential searches for catalog blocks with direct accessing. This is accomplished using two tables. One table provides information about free space in the catalog blocks, and a second (reference table) assigns the logical block number of the catalog block to the file names or job variable names.
Both tables are created by the SCA task.

SCA can only be called for SF pubsets. In the case of SM pubsets, accelerated access to catalog entries is already implemented internally.

Two versions of SCA are available: with and without task switching. The use of SCA without task switching is recommended.

- SCA without task switching

  The reference table is created in class 4 memory and is thus available to all tasks. All SCA routines (except for table creation) run under the accessing user task. There is no overhead for task switching, and the overhead for serialization measures is minimal.

- SCA with task switching

  The reference table is managed by the SCA task in its class 5 memory.
  Each catalog service request from a task results in a task switch.

You can store the variants that are to be used for a pubset or specify whether SCA is to be started automatically (standard for BS2000/OSD V8.0 and higher) when the pubset is imported in the MASTER catalog MRSCAT (START-SPEEDCAT operand in /ADD- and /MASTER-CATALOG-ENTRY).

The “SCA without task switching” version may be expected to require approx. 1 to 2 % less CPU time, but somewhat more class 4 memory than does the “SCA with task switching” version.

Automatic startup of SCA

If SCA is installed SCA is started in *SPEEDCAT-TASK mode when a pubset is imported in order to guarantee high-performance catalog access with SF pubsets.

The default value START-SPEEDCAT=*AUTOMATIC is therefore used in the ADD-MASTER-CATALOG-ENTRY command.

Advantages of using SCA

1. Increased total throughput rate for the system when the workload involves an intensive use of the catalog.

   The reduction in the number of physical accesses of TSOSCAT per logical catalog call (OPEN, CLOSE, ...) results in a lowering of the system workload and a marked reduction in the catalog access time.

   In addition, the burden on the disk drives containing the catalog files is reduced. If SF pubsets are used, these are generally the xxxx.0 volumes. As far as the whole system is concerned, this means that the throughput of the accessing user tasks can be increased.
   Runtime for programs with a high frequency of catalog access (e.g. ARCHIVE) as well as response times for commands which need to access the catalog regularly (e.g. /CREATE-FILE, /MODIFY-FILE-ATTRIBUTES, /IMPORT-FILE) can be improved considerably.

   SCA is indispensable for fast systems operating in interactive mode. Its use leads to a marked increase in throughput.

2. Greater flexibility in computer center organization

   The use of SCA makes one of the previous organizational methods of improving catalog performance redundant. It is therefore no longer necessary to place frequently used catalog entries at the beginning of a catalog block chain.

3. Full compatibility

   SCA does not change the structure of the catalog. As a result, both the user interfaces and the internal system interfaces to the catalog management system are retained.

> **i** Significant improvements in performance can only be expected when the user has a large number of catalog entries (more than 60-100 file entries per user ID).

SCA can be started and/or terminated for different catalogs simultaneously, either automatically, when the pubset is imported (determined by the MRSCAT entry), or by procedure call (/ENTER-JOB FROM-FILE=SPEEDCAT.ENTER.START) after the pubset has been imported.

Creating system files

System files are required by the operating system to complete its tasks. Because of their crucial importance, these files are amongst those that have the greatest effect on the performance of an IT system. Of the many system files, the following are the ones accessed most regularly:

Function	DMS name
File catalog (SF pubset) File catalog (SM pubset)	`$TSOS.TSOSCAT` `$TSOS.TSOSCAT.<volset-id>`
Paging areas	`$TSOS.SYS.PAGING.<vsn>`
Background storage for access method EAM	`$TSOS.SYSEAM`

Priority must be given to **avoiding access conflicts** when creating these files.

The usual way of reducing the probability of access conflicts is to distribute the files over several volumes. If, for reasons of economy (e.g. utilization of disk capacity), user files and system files are placed on the same volume, then it is desirable to select user files which are not frequently used.

The characteristic features of system files are the **file size** and the **access frequency**.

> If a system has already been set up, the location of the paging areas can only be changed with great difficulty. Therefore particular attention must be paid to the location and size of this system file at configuration.
> If the system is being used for program development, then cataloged module libraries (TASKLIBs) and macro libraries are often accessed so regularly that their location must be taken into consideration at configuration.

File catalog (SF pubset)

The file catagog (this is the file TSOSCAT) is only accessed from CMS (Catalog Management System) routines. The PAM blocks in this file contain:

- file entries or
- job variable entries or
- global management data
  (MRS catalog and entries describing the current state of TSOSCAT)

The file is accessed in blocks. The system parameter BMTNUM (see "BMTNUM and CATBUFR") or /IMPORT-PUBSET or /ADD- and /MODIFY-MASTER-CATALOG-ENTRY are used to specify how much I/O buffer is available to the CMS. Simultaneous user requests to the same catalog block are synchronized with the aid of a Buffer Management Table (BMT) and a bourse (one BMT and one bourse per catalog I/O buffer).

The catalog file can be divided into several extents. For technical reasons, the first extent of the file must be on the volume xxxx.0. If there are several pubsets, there is a catalog on each pubset. Placing parts of TSOSCAT on heavily used disk drives must be avoided. All volumes with paging areas and SYSEAM area fall into this category.

Catalog format

The only still supported catalog format EXTRA LARGE with a maximum catalog size of 64.016 PAM pages is incompatible to older formats. Possibly existing catalogs in other formats are automatically converted when importing the pubset.

Up to 100 subcatalogs each with 64,016 PAM pages can be created for SM pubsets with catalogs in EXTRA LARGE type for each of the special catalogs #MIN, #JVC and #PTV (/ADD-CATALOG-FILE).

CMS recognizes when a catalog is 90% full and automatically extends it. If none of the existing subcatalogs can be extended, a new subcatalog is then created.

The file can also be expanded manually using the /MODIFY-FILE-ATTRIBUTES command (SPACE operand). The change becomes effective immediately and not only after the next time the pubset has been imported.

Creating TSOSCAT

The size of TSOSCAT is determined by

- the number of users permitted to use it
  (represented by the user IDs entered by means of /ADD-USER)
- the number of files and job variables in the system
- the size of the catalog entries; with files this is dependent on the number of extents they have, and with job variables on the size of the JV value.
- the splintering of catalog entries, which is caused by selective deletionThe space which becomes free by deleting “individual” entries is available for creating new entries, but is not used for anything else. Only after all entries have been deleted is a catalog block released.

The entries in TSOSCAT are stored in catalog blocks (each 4 KB in size):

- A catalog block for files, the so-called “primary block”, is reserved for each user even if a user creates no files.
- A catalog block contains either only file entries of one user or only JV entries of one user.

- 1 to 11 file entries fit into a catalog block.
  An average of 10 file entries per catalog block can be expected on a pubset which is sufficiently large and maintained using SPACEOPT.
- 8 to 18 JV entries fit into a catalog block.
  It is difficult to specify an average because the size of the JV values can be freely selected by the user. For example, in the case of MONJVs with a standard size of 128 bytes, 11 JV entries fit into a catalog block.

The average number of the catalog blocks which are occupied by “primary blocks” and file entries can be estimated using the following formula:

Number of catalog blocks + Number of user IDs + (Total number of files / 10)

The formula takes into account the splitting up of catalog entries and the decreasing capacity as the size of the entries increases. Additional catalog blocks are required for job variables.

Example

Calculation of the catalog size for 150 user IDs with 65 files per user ID:

TSOSCAT = 150 + (150 * 65 / 10) = 1125 catalog blocks

If the software product SCA (Speed Catalog Access) is not used, no more than 65 files or job variables should be cataloged per user ID . If SCA is used, then a greater number of files or job variables is possible.

Location of TSOSCAT

A TSOSCAT file of 2000 PAM blocks is automatically created on volume xxxx.0 at system generation.

Assuming that the paging area created on the system residence (xxxx.0) is **not used during subsequent operation**, TSOSCAT should be stored on volume xxxx.0 in its required size (in the case of installation using SIR).

If the load on volume xxxx.0 essentially consists only of user activities (max. 15% capacity utilization) and catalog accesses on the part of the system (also max. 15% capacity utilization), it will manage 50 catalog I/O operations per second at a 70% read hit rate. It would be advantageous to reduce this by increasing the number of CMS input /output buffers and using SCA.

If several pubsets are used, a copy of TSOSCAT is automatically placed on each pubset. The recommendations are then true for volume xxxx.0 for each pubset.

Once normal operation has been started, it is advisable to monitor the number of catalog I/O operations per second using the software monitor SM2 (reports “CMS Queues”, “CMS IO'S” in the report group CATALOG-MANAGEMENT).

As well as the frequency of catalog I/O operations, it is also necessary to consider the address space requirements for tables and buffers and delays resulting from synchronization when calculating CMS performance for productive operation. If multiple pubsets are used, catalog I/O buffers and their management tables (BMT) are created for each pubset. If there is a large number of pubsets, then the address space required by CMS can increase considerably.

It is possible to define the number of CMS buffers for each pubset with /ADD-, /MODIFY-MASTER-CATALOG-ENTRY or /IMPORT-PUBSET.

Example

The system parameters BMTNUM=32 and CATBUFR=N were specified in the startup parameter service. For 2 pubsets CMS needs the following amount of work area:

```
(33 * (4096 + 178) + 388) * 2 = 282.860 bytes,
  |     |      |       |    |   i.e. 277 KB of class 4 memory
  |     |      |       |    |
  |     |      |       |   Number of pubsets
  |     |      |      Central CMS work table
  |     |     BMT + DMS access tables
  |    Catalog block I/O buffer
 Number of CMS I/O buffers + 1
```

An additional I/O buffer for the static MRS catalog (approx. 4 KB) is required for pubsets with EXTRA LARGE catalogs.

To protect simultaneous requests, one bourse is created per catalog buffer. In addition, further bourses are used as protection against competing calls from CMS routines.

From time to time, it is necessary to revise the organization of files on disks and of file entries and job variable entries in catalog blocks. This is done in order to keep path lengths as short as possible when searching for file entries and job variable entries, and to ensure that as few catalog buffers as possible are locked for simultaneous requests from other users during this search. During this process, all files and job variables must be saved, deleted and recreated. When recreating, two requirements - which may possibly be contradictory - must be fulfilled as far as possible.

1. On the one hand, large files should be transferred into the system first, then this allows them to be distributed according to considerations of practicability. Small files can be placed in gaps.
2. On the other hand, CMS capacity is used efficiently if file entries for heavily used files are as near as possible to the beginning of the series of catalog blocks for each user ID. CMS starts searching for file entries or job variable entries in the first block of the series of user catalog blocks in each case.

The second requirement should be given priority.
If there are too many user IDs and file and job variable entries per user ID, SCA should be used (for details, see section "Accelerating catalog accesses with SCA").

File catalog (SM pubset)

An SM pubset consists of a control volume set and a number of “normal” volume sets.

The file catalog of an SM pubset is distributed to the individual volume sets in such a way that every subcatalog contains the information for the files on the corresponding volume set.

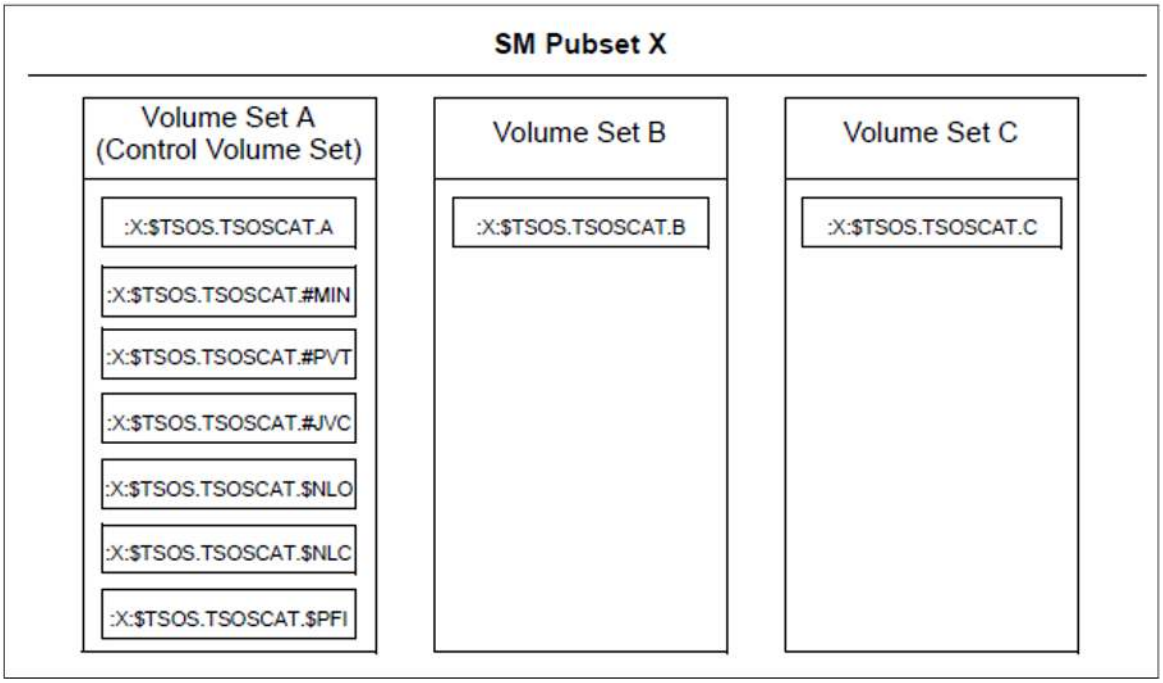


Figure 10: Structure of the file catalog of an SM pubset

When several SF pubsets are merged to form an SM pubset, you must remember that the values for the CMS buffers are adapted in the MRSCAT.
When the values are simply added, better catalog performance generally results on account of the better equalization of load fluctuations on the catalog.

The following other global subcatalogs are on the control volume set and should be divided up there among several volumes.

- Subcatalog for migrated and empty files
- Subcatalog for files on private volumes
- Subcatalog for job variables

Other subcatalogs on the control volume set ($NLO, $NLC, $PFI) are used exclusively for data security.

The central user catalog is also on the control volume set.

The structure of the individual subcatalogs corresponds to that of the TSOSCAT file of an SF pubset. Therefore, when creating these subcatalogs the same recommendations apply as for creating TSOSCAT on SF pubsets.

Use of SCA is not possible; accelerated catalog access is already implemented internally.

Paging areas

Paging areas are cataloged as files with the name:

`:<catid>: $TSOS.SYS.PAGING.<vsn>`

Using the parameter service, a choice can be made from all available paging files every time the system is run. This enables modifications made necessary by various workload conditions to be carried out in each session.

All pubsets containing at least one paging area for the imminent session must be online before system initialization (before startup). These pubsets are used exclusively by the startup procedure, but are not implicitly imported (i.e. the paging areas are used even if the pubset is not imported). During the session, these pubsets cannot be used by other BS2000 systems (as “shared pubsets”).

Paging areas may also be contained on SM pubsets.

Creating paging areas

When a pubset is created, the paging files are reserved on each of the volumes of this pubset intended for paging, and an entry is made for each one in the file catalog of this pubset. A volume can only contain one paging file (although the file can have several extents).

The size of a paging area depends on the number of programs executing simultaneously and their virtual size. To this must be added the system address space currently generated.

The following formula is recommended for creating paging areas:

Size = (Number of users * Virt. program size + System address space) * 2

After normal operation has started, a check should be carried out (report “Paging Area Utilization” in the report group MEMORY of SM2) in order to ascertain the actual file size required.

For programs executing simultaneously, the size of the paging area should be several times that of average requirements, but should not total more than twice the maximum requirement.

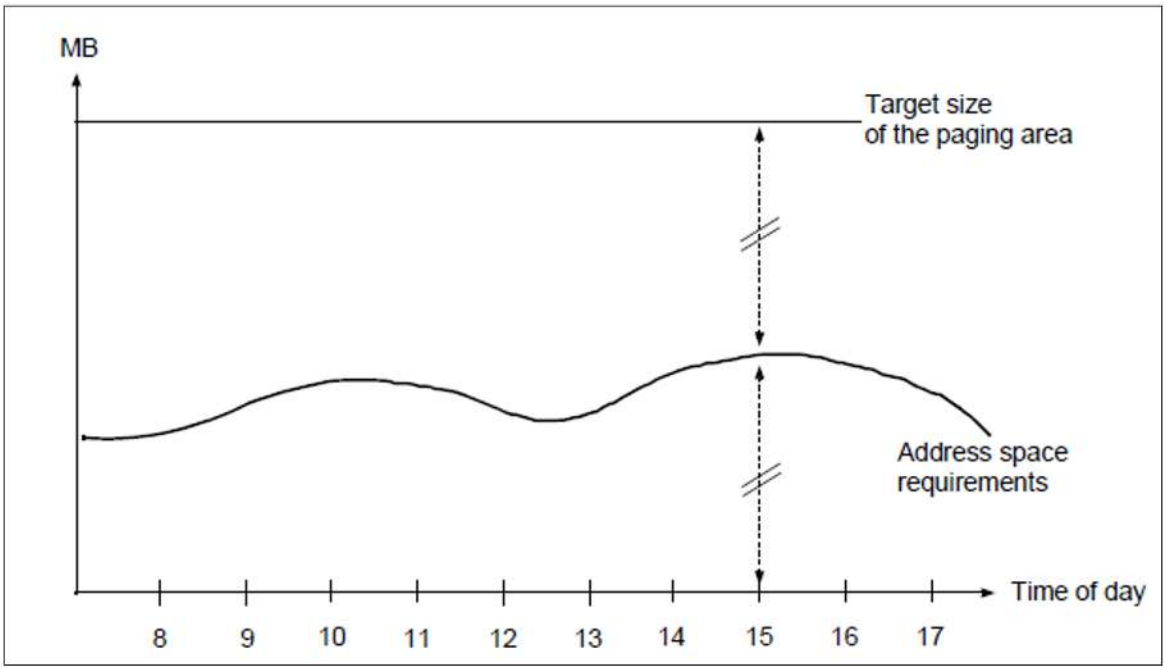


Figure 11: Allocation of the paging area

The **access frequency** of the paging area depends on:

- the operating mode (TP, interactive, batch mode)
- the size of main memory
- the speed of the server

Interactive mode

Interactive mode is characterized by a large number of tasks which, however, are relatively small as regards working set requirements (50 - 150 4 KB pages).
Since few page requests are normally made to the individual user tasks per dialog step (5 - 10), there is no significant drop in performance visible to the user, even when the total paging rate is high.

However, a paging operation not only involves a time delay, it also places considerable demands on the system.

Therefore, for reasons of efficiency, paging should take up no more than 3% of the performance capacity of the server. With high-powered servers even this percentage is too high, and requires an uneconomically large number of volumes with paging areas.

TP mode

TP mode is characterized by a small number of tasks with relatively large working set requirements (several MB in size).

In order to fulfill a particular user function, a large number of pages are addressed, often over several tasks (especially in the case of data base/data communication applications).

Increasing the paging rate usually leads to a severe drop in performance as viewed by the user. Due to the heavy page requirements for fulfilling a function, the paging rate climbs dramatically with even a slight shortage of main memory. At this point, an insignificant additional increase in workload is sufficient to cause a drop in performance (exponential increase in paging rate: the system “capsizes”).

TP mode is considerably more sensitive to main memory shortages than the other operating modes. Consequently, the reference values for the upper limit of the paging rate must be correspondingly lower (see section "Standard values for BS2000 servers").

From an economic point of view, it is always desirable to reduce the paging rate to values less than those suggested above.

> In order to be better able to react to sudden overloads, utilization of a volume which is used exclusively for paging activities should not exceed 10%. With a read hit rate of 70% this volume will handle approx. 30 paging I/O operations per second. As normally no more than 10 - 20 paging I/O operations should be performed (see the section "Standard values for BS2000 servers"), it is generally not necessary to distribute the paging area over more than 3 to 6 physical devices.

Location of the paging areas

Paging files should be the same size on all volumes (if possible only one extent) so that a constant I/O workload can be achieved on all volumes. As a rule the I/O rate correlates to the size of the paging file.

If possible, paging areas should never be activated on volumes containing the file catalog or a SYSEAM extent. This also applies to volumes which are heavily utilized by user files.

Further details can be found in the “Introduction to System Administration” [10 (Related publications)] (under “Memory management”) and in the “System Installation” manual [32 (Related publications)] (under “Handling important system files”).

Sample calculation of size and distribution of the paging areas

Assumptions

- TP mode 1200 user in over 30 UTM tasks and 5 database tasks each with an average program size of 120 MB
- System address space: 240 MB
- Paging rate: Max. 100 I/O operations per second
  As explained on "TP mode" above, a paging volume should not perform more than 30 paging I/O operations, i.e. (3) - 4 volumes with paging areas are required

Calculation

(35 * 120 MB + 240 MB) * 2 = 8880 MB

In order to achieve an even distribution of workload over the volumes, the paging area must be divided into extents of equal size, i.e. 2220 MB per volume.

Selection of specific volumes for paging during startup

If the paging areas are not selected via the parameter service, the areas created on the home pubset are used.

Specifications in the system parameter file determine which paging areas are actually to be used for the following session. If, after planning the workload of an IT system, the load profile is already known before system initialization, the paging parameters can be set in a realistic fashion according to this profile using the parameter file.
As a rule, no paging area should be reserved on the first volume of each pubset (VSN=xxxx.0), as the file catalog relevant to that pubset is always stored here and, if applicable, a high I/O rate can be expected with regard to CMS.

Extending and reducing the paging area

The paging area can be extended as well as reduced in size. This means that it is possible to relocate the paging area from the performance-critical home pubset to a pubset with a lower workload. For more information, refer to the section entitled “Paging area” in the “Introduction to System Administration” [10 (Related publications)].
You will find information on increasing the paging area or reducing the paging activities to selected volumes in the descriptions of /EXTEND-PAGING-AREA and /MODIFY-PAGING-AREA-ATTRIBUTES in the “Commands” manual [15 (Related publications)]).

SYSEAM file

The SYSEAM file is used to store temporary data (access method EAM). SYSEAM can be created on any pubset.

Users access the SYSEAM file which resides on their user default pubset. If there is no SYSEAM file available on the user default pubset, the SYSEAM file of the home pubset is accessed.

The ideal size of SYSEAM can only be determined after observing the mean utilization over a sufficiently long period. For information on the initial setting, see section "EAMMEM, EAMMIN, EAMSEC, EAMSIZ". Observation is carried out with the software product SPCCNTRL (Space Control). Heavy demands are placed on the SYSEAM file, especially in the case of interactive program development in interactive mode: frequent compilation runs have a considerable effect on both size and access frequency.

For monitoring access frequency, use of SM2 is recommended (file monitoring).

Location of SYSEAM

To avoid SYSEAM becoming a bottleneck when there is a large number of users in conjunction with powerful CPUs, it makes sense to distribute it across several pubsets.

If possible, no SYSEAM extents should be stored on volumes with paging areas.

Since all I/O requirements are subject to caching (write hit = 100%) in storage systems, it is not necessary to distribute the SYSEAM file across several volumes in a pubset.

Any increase in the size of the SYSEAM file workload is generally reflected by a corresponding increase in the size of the file. If this is the case, it is recommended that the volumes required are reserved exclusively for SYSEAM and that, with regard to the I/O rate, the upper workload limit of 30% is approached. In this case, a volume manages around 300 I/O operations per second.

SYSEAM peak loads

The main users of SYSEAM are the compilers, the software product LMS and the linkage editors (binders) BINDER and DBL.

Systems support should know the times when these program services are called. Then decisions can be made whether additional pubsets need to be created and when they should be imported.

If a high proportion of the workload consists of program services, the TSOS macro libraries, the user macro libraries and the MODLIB files should not be placed on volumes which also contain SYSEAM extents.

If sufficient main memory is available, it makes sense to place part of the SYSEAM area in class 4 memory (this applies only for the home pubset) using the system parameter EAMMEM (see section "EAMMEM, EAMMIN, EAMSEC, EAMSIZ"), thus saving on SYSEAM I/O operations. This permits compilation and binder runs to be accelerated correspondingly.

Measures which can be taken after SYSEAM secondary allocation:

When the console message is issued in a session:

```
DMS0852 (&00): (&01) PAM PAGES AND (&02) EXTENTS
```

(where &00 is the name of the SYSEAM file), the file size exceeds the value of EAMMIN or the pubset-specific entry for MINIMAL-SIZE in the MRSCAT.

This message, which is issued whenever the file is extended or shortened, should appear only during peak workloads. If it is issued frequently, it means that the value selected for the system parameter EAMMIN or for MINIMAL-SIZE in the MRSCAT entry is too small. In order to avoid secondary allocations, the size of EAMMIN must be adjusted and the file extended accordingly. After system initialization, the files can be erased and then newly created again. During this time, no users may access the SYSEAM files. A renewed startup is not necessary afterwards.

The SYSEAM files are reset to the size of the system parameter EAMMIN via a system cold start.

Message files

Two areas must be taken into consideration if system messages are to be created and output efficiently in BS2000:

- message management

  All the messages which could be needed while the BS2000 system is in use are stored in a number of ISAM files. Message files which are essential for the system run and ones which have been specified by means of parameters are made available at startup (system parameters MSGFILii and MSGNOFL, see section "EAMMEM, EAMMIN, EAMSEC, EAMSIZ"). Further message files can be specified in the MIP parameter file (SYSPAR.MIP.<version>). However, they are enabled only after the MIP subsystem has been loaded. Further message files may be added or removed during the session. See the “Introduction to System Administration” [10 (Related publications)].

- System-internal message handling

  Each message file created with the utility routine MSGMAKER contains the message text and a number of attributes (for searching for the message text) as well as the meaning and the response text (for /HELP-MSG-INFORMATION).

When the system is being set up, the files is opened and the management section is read into a work area in class 4 memory.
If users make requests for system messages to be output in the course of processing, the following actions are taken when handling these message outputs:

- on the basis of the accompanying message code, a check is made as to whether the message is already in the message buffer or whether it needs to be read using the ISAM access GETKY
- on the basis of the weight code, a check is made as to whether this message is to be output on the output medium console or whether it is to be suppressed
- variable sections of the text are updated and output is initiated (if suppression has not been specified). At least one change of task is necessary for this.

Output media can be any of the following:

- the main console and/or the subconsole
- and/or SYSOUT/SYSLST of the user task
- and/or a user area (buffer)

Every request for output of a message to the master console and/or subconsole (even if output is to be suppressed) is also written as a record to the CONSLOG file.

Saving message files

Message files which are only required for special applications should not be placed on the home pubset disks (where they merely occupy space unnecessarily), but rather on the pubset belonging to the particular application. For current requirements, disks can be attached with /IMPORT-PUBSET or the files can be restored from backup data (e.g. using HSMS). The message files are then transferred into the active system using /MODIFY-MSG-FILE-ASSIGNMENT or /MODIFY-MIP-PARAMETERS.

Individual messages (or the contents of complete message files) are integrated into the message files required for the session.

Number of message files

The messages of special software products can remain in the software product's message file. They are automatically installed with IMON.

The number of message files managed as an integral part of the overall system, and which of these should be made available in which session and perhaps removed after they are no longer needed, is left to the discretion of systems support.

> **i** A smaller number of additional message files facilitates management and requires less system address space. A larger number of message files allows the system to be tailored more exactly to the workload conditions.

Suggestions for limiting or speeding up the output of system messages

- Suppression of certain system messages

  Preparing, updating and outputting messages is necessary for the system to run properly, but also simultaneously increases its workload. The way in which messages are handled should be tailored to the particular requirement. For example, the request to output a message can be revoked if the loading of a program (/START-EXECUTABLE-PROGRAM) has no relevance for the operator's activities and no log of the event is evaluated (set system parameter EACTETYP to 0, see section "EAMMEM, EAMMIN, EAMSEC, EAMSIZ").

  If different applications are processed sequentially during data processing, the number of outputs to the consoles which are to be suppressed should be chosen to suit each individual application.

  This means that:

  - All messages which need to be handled differently with regard to output suppression are given different weight codes in the message files. /ADD-CONSOLE-FILTER is used to set the filter level most suitable for each individual type of workload. Productive use, program development, test operation, diagnosis following faults or after version changeover etc. and any mixture of these are all different types of workload.
  - An alternative method for suppressing output on the basis of the weight code is to use the message subscription or message suppression function (/MODIFY-MSG-SUBSCRIPTION, /SET-MSG-SUPPRESSION). This makes it possible for a message-receiving entity to subscribe to or suppress individual messages. Both possibilities can also be used in combination:
    1. First you roughly define the message set by allocating routing codes and specifying weight codes.
    2. Then you carry out the fine-tuning via subscription/suppression.

- Selective output of system messages

  The following applies to the assignment of routing codes (parameter service, statement SET-CODE):

  - With the appropriate routing codes, consoles can receive operating messages.
  - The passing of messages is performed by a system-internal communication mechanism (with the aid of bourses) and incurs a noticeable system overhead.
  - When assigning the routing codes, you should always check which messages or commands must be available on the consoles, so as to avoid superfluous messages by restricting assignment.
  - The message delivery can be optimized by means of the message subscription/suppression mechanism.

- Speeding up message output with DLAM (Dynamic Loadable Access Method)

  Systems administration can set the DLAM flag for any message.These flags are stored in the directory of each message file. At startup, the directory of all message files which, according to the start parameters, should be available at “system ready” (system parameter MSGFILii, see section "EAMMEM, EAMMIN, EAMSEC, EAMSIZ") is evaluated and placed in the range assignment lists (in the message handler tables). Following this, all messages with the DLAM flag are read from the appropriate files.
  (MSGMAKER utility routine, statement/mask: ADD-MSG and MODIFY-MSG)

  Advantage

  The address space requirements for buffered message texts do not change during the system run (static format from startup). The path lengths are shortened if DLAM flags are set as it is then no longer necessary to check whether a message text which is currently required is already buffered. The physical I/O operation is brought forward from the system run to the startup phase.

  Messages are created in class 5 memory at startup. No physical I/O operations are executed when a message is called during the system run.

  Effort

  Selecting and flagging the individual messages takes up expensive processing time. The use of DLAM is recommended on large servers with a high throughput rate.

- Speeding up message output with LOCAL-DLAM (Local Dynamic Loadable Access Method)

  If a message has the access method LOCAL-DLAM (assigned with the MSGMAKER utility routine), it is copied into class 4 memory at activation.

  Advantage

  As with access with DLAM, no physical input/output is necessary. In addition, there is no access to the central MIP task - all actions are executed under the job-source task. This not only reduces the system overhead, but also prevents possible lock situations between the central MIP task and other job-source tasks.

- Speeding up the message output using the system parameters MSGCENTL and MSGCENTN (see section "EAMMEM, EAMMIN, EAMSEC, EAMSIZ")

  MSGCENTL defines the maximum length of messages and MSGCENTN the number of messages which can be buffered in class 4 memory in order to reduce the number of file accesses.

  The amount G of class 4 memory required is calculated as follows:

  G = MSGCENTL * MSGCENTN [Byte]

Creating user files

User files are generally created and processed by users in the course of their session. It must be noted that each execution of the following operations increases the workload on the file catalog: creating and deleting a file, opening and closing a file, changing the size of a file.

The main criteria for choosing the data volume and the logical operating mode are the size of the file, the access frequency and the frequency of accesses to the catalog during file processing.

File size and access frequency

When data resources are planned, the frequency with which files are accessed has a considerable influence on system performance.

- To keep the load on the volumes as low and uniform as possible, (large) files which are not accessed frequently should be placed on volumes on which frequently accessed files have already been installed and vice versa.
- Frequently accessed files should always be placed on separate volumes (especially in the case of write-intensive applications). This is only practical if a number of tasks are involved.
- Input/output files which are accessed alternately within the same processing run should be on separate volumes.
- It makes sense to use frequently accessed files belonging to time-sensitive applications as HIPERFILEs (see section "Working with HIPERFILEs and DAB").
- As a rule loads with normal to high write rates are managed efficiently in the disk controllers using familiar caching resources. However, bottlenecks can occur (write delay) in the case of loads with very high write rates to particular hotspots/files.

  To counteract this effect, it is recommended to distribute the hotspots across several logical volumes or physical devices:

  1. Hardware measures:
     Use of RAID 1/0 or RAID 5 (see section "RAID levels and their performance") with additional software support through the use of PAV (see "PAV (Parallel Access Volume) on /390 servers").
  2. Software measures:
     Distributing the hotspot or file over multiple files, extents or volumes. An example of this is the distribution of KDCFILE from UTM to multiple files (see "Optimizing the various phases").

Frequency of catalog access

As already mentioned in section "Logical operating modes of disks", the internal outlay for processing calls with catalog access is at its smallest for public volumes or pubsets, followed by private volumes. Throughput for the individual logical operating modes is therefore dependent on the ratio of management IO operations to “productive” I /O operations.

Public volumes or pubsets are most suited to files subject to frequent catalog access (procedures, ENTER files, a large number of OPEN/CLOSE operations in quick succession).

Files with a low frequency of catalog access can be stored on private volumes.

The high frequency of catalog accesses is only acceptable for shareable public volume sets when there is joint access by several servers.

In order to reduce the number of management I/O operations, the following points should be borne in mind:

- select the primary and secondary space allocation in /CREATE-FILE to suit the file size expected

- avoid erasing and creating a file unnecessarily (job command sequence: /CREATE-FILE, /DELETE-FILE, /CREATE-FILE for the same file name)
- use OPEN/CLOSE operations sparingly with the commands /COPY-FILE, /CALL-PROCEDURE, /INCLUDE-PROCEDURE, /ENTER-JOB, /ENTER-PROCEDURE, /START-(EXECUTABLE-)PROGRAM, LOAD-(EXECUTABLE-)PROGRAM and /ASSIGN-SYSDTA also trigger OPEN/CLOSE operations.)

Reorganization

With larger systems, it is advisable to reorganize all files on a regular basis (e.g. every three weeks). This reorganization counteracts the splitting of files into a number of small extents, which cause an increase in positioning times. The reorganization is considerably simplified by the software product SPACEOPT.

SPACEOPT clears up the fragmentation through optimum relocation of the file extents on the volumes of a pubset. Files can also be reorganized if the free space available is smaller than the size of the file, i.e. SPACEOPT also permits local enhancements on the basis of the free disk space available.

SPACEOPT can adjust the size of BS2000 volumes to the size of the LUNs (Logical Units) with which they are implemented in ETERNUS DX/AF storage systems. Such an adjustment can be required after disk migration using DRV, in which the source and target disks must have the same capacity. Information on disk migration, e.g. from D3475 to D3435, is provided in the “DRV” manual [7 (Related publications)].

A further application is the expansion of a BS2000 volume following the expansion of a LUN through the inclusion of additional disks in the ETERNUS DX/AF storage system. This function is only available for the disk format D3435 (FBA data format).

SPACEOPT works on a volume basis, i.e. tasks can be specified for individual or all volumes of a pubset. SPACEOPT offers options for assessing the occupancy and fragmentation status and can be used during operation.

SPACEOPT can also move files if they are opened.

It is advisable to use SPACEOPT in off-peak times, as it can incur a high I/O workload depending on the degree of fragmentation of the volumes.

> The performance of other BS2000 guest systems or of other servers which execute I/O operations to/from the same physical disk can be negatively influenced by reorganizing using SPACEOPT.

You will find detailed information on SPACEOPT in the “SPACEOPT” manual [31 (Related publications)].

Working with HIPERFILEs and DAB

The objective of the HIPERFILE (High Performant Files) concept is to increase the speed of file accesses and to avoid or eliminate I/O bottlenecks by buffering the data in the main memory using the software product DAB (Disk Access Buffer).

The purpose of DAB (Disk Access Buffer) is to provide faster access to disk files by means of caching in the fast main memory (MM).

Main memory is a volatile storage medium that is suitable primarily for the caching of read accesses as well as for write jobs to temporary files. Unsaved write data in the buffer is lost after a system crash if it is on a volatile storage medium.

With BS2000 V21.0A, there is only one variant of the HIPERFILE concept, the ADM-PFA caching (administrator-controlled Performant File Access), in which the systems support uses commands to specify which files or volumes are to be buffered in which cache medium and which caching mode (read, write or read/write mode) is to be used.

Caching modes

When the data requested is available in the cache, a cache hit is said to have occurred; otherwise, the attempted access is referred to as a cache miss. The ratio of cache hits to the total number of accesses is referred to as the cache hit rate. The higher the hit rate, the greater the benefit provided by the cache. The cache in the main memory is examined here; the cache integrated in the storage system works “invisibly” in the background.

The following caching modes are available (in the diagrams below, accesses are depicted by arrows, with read accesses on the left and write accesses on the right):

Read cache

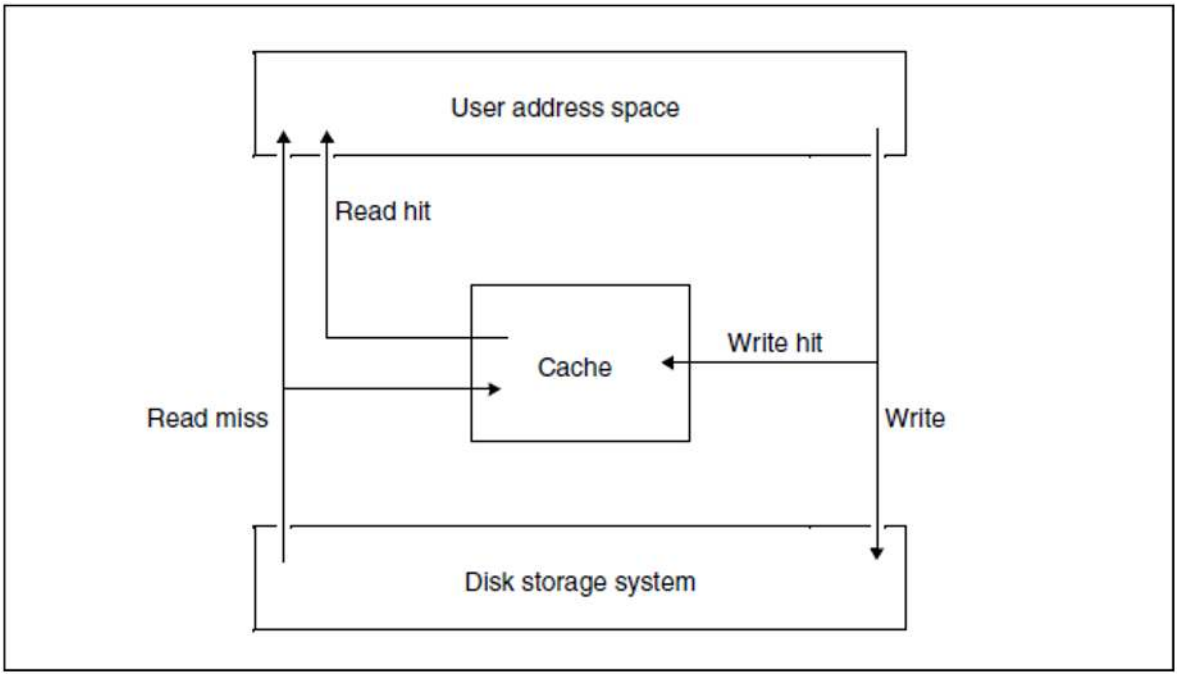


Figure 12: I/O processes for a read cache

Read hit:

The data is available in the cache when accessed and is read from there.

Read miss:

The data is read from the storage system and entered into the cache at the same time so that subsequent read requests can be resolved from there.

Write:

Write accesses are always to the storage system. If the relevant data block is already in the cache (“write hit”), the contents of the block are updated without making the write operation any faster.

Write cache

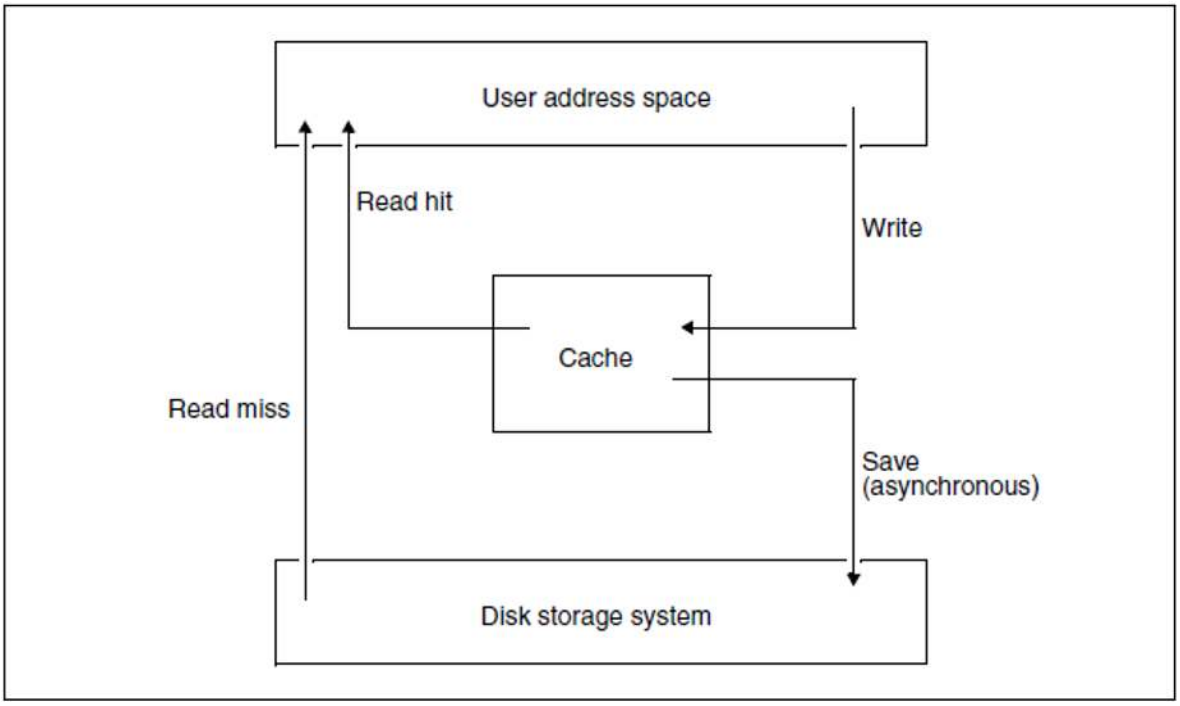


Figure 13: I/O processes for a write cache

Read hit:

The requested data is read directly from the cache (as in a read cache)

Read miss:

The data is read from the storage system but not entered in the cache.

Write:

The data is always written to the cache irrespective of whether or not the data block exists there. The application writing the data considers the output process to have been concluded as soon as the data has entered the cache. The writing of the data to the storage system is generally initiated immediately afterwards.

Read/write cache

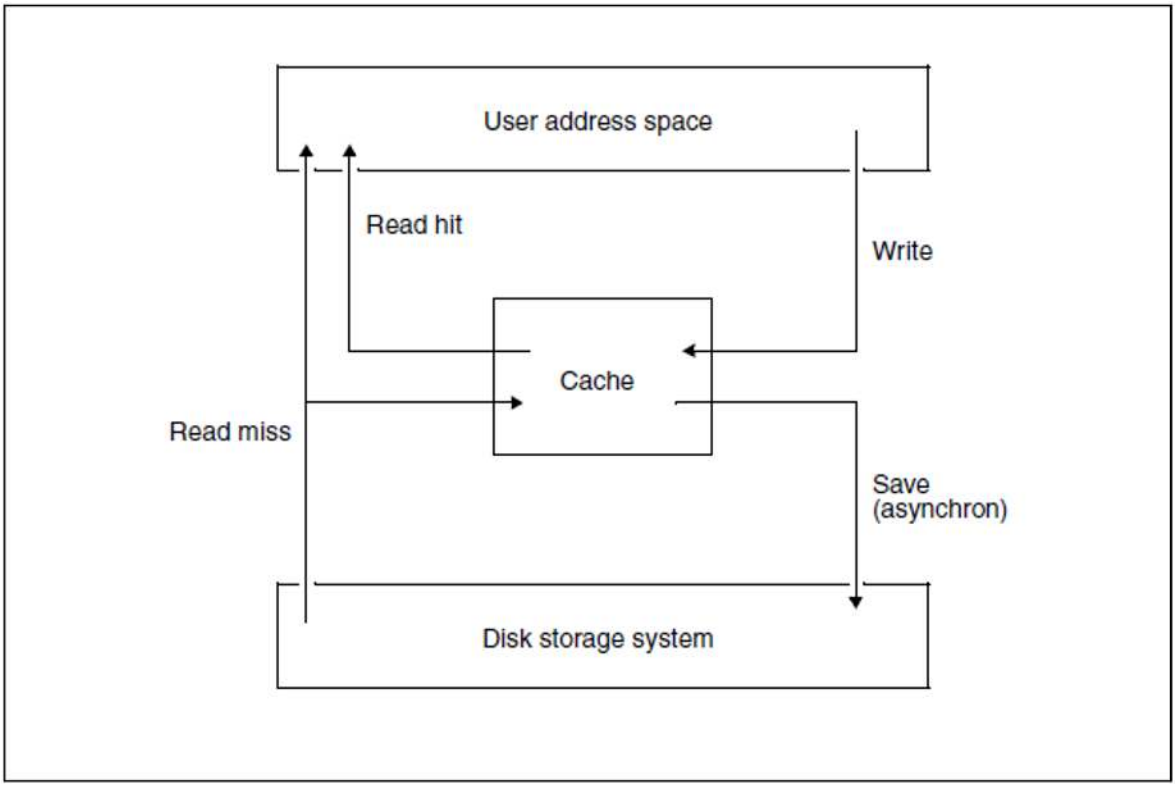


Figure 14: I/O processes for a read/write cache

Read hit: Like read cache

Read miss: Like read cache

Write: Like write cache

ADM PFA caching

In **ADM PFA caching**, systems support controls the creation of cache areas and the allocation of selected data areas (a cache area is always a subarea of a cache that represents a separate administration unit). Systems support can use /START-DAB-CACHING to create independent cache areas and thus to separate selected data belonging to different applications. A cache area can also be created for entire volumes. You will find a detailed description of this in the manual “DAB” [5 (Related publications)].

The term **AutoDAB** refers to a variant of DAB caching. It involves automatic and intelligent procedures which, on the one hand, greatly reduce the work of systems support when administering the caching and, on the other, allow the efficiency of the DAB caching to be further increased.
A DAB cache area can be set up as either an “automatic” or a “non-automatic” cache area.

Any of the following three settings is possible when using the cache buffer:

1. The authorized user can specify which files are to use the cache area created in which cache mode.
2. Systems support can specify that all user files of a pubset are to use the cache area.
3. Systems support can create an automatic cache area in which the optimum selection of the files to be supported occurs automatically. This selection is permanently monitored and, if necessary, corrected. AutoDAB enables entire pubsets to be covered by a cache area.

DAB manages each cache area in the form of so-called cache segments whose default size is 32 KB for automatic cache areas. The default size of a cache segment in automatic cache areas is 32 KB. For non-automatic cache areas it can also be specified by systems support (4 KB, 8 KB, 16 KB or 32 KB). Automatic caching must be stopped during the modification.

The first time an area (file) supported by DAB is read-accessed, DAB reads from the storage system page-chained, adjacent data blocks, the number of which depends on the set cache segment size.

Spatial locality

In read accesses with good spatial locality (particularly for sequential accesses), large segments (= 32 KB) allow such a high throughput of I/O jobs that in this case, there are a large number of read hits and the accesses to the storage system are reduced to a minimum.
In read accesses with low spatial locality (random access, the most frequently used access type in database mode) it is advisable to select small segments (= 4 KB).
AutoDAB recognizes good locality and, when it finds it, automatically performs a prefetch (even if a smaller cache segment was set).

Chronological locality

The blocks read into the cache remain there until DAB is terminated or this memory area is used for another cache segment.
Preemptions are managed on the LRU (least recently used) principle, which means that a high read hit rate can be achieved given good chronological locality of the accesses even with a relatively small cache area.
For performance-critical applications whose read accesses have low chronological locality, DAB offers the possibility of resident buffering.

In the case of a read/write (or write) cache, write jobs are always entered in the cache directly, regardless of whether or not the data block is already in the cache. As far as the application is concerned, the output operation is completed as soon as the data is written to the cache. DAB saves the data on the storage system later.

It must be noted that DAB creates the system buffer in main memory on a resident basis (i.e. main memory must be large enough).

Systems support uses the following commands to control the use of ADM PFA caching with DAB:

Command	Function
`START-DAB-CACHING`	Creates and assigns attributes to cache areas
`SHOW-DAB-CACHING`	Outputs information on the current DAB configuration and access counters (hit rates)
`STOP-DAB-CACHING`	Saves cache areas on disk and deletes them
`MODIFY-DAB-CACHING`	Dynamically modifies parameters of a DAB cache area
`FORCE-STOP-DAB-CACHING`	Deletes a DAB cache area without backing up the write data

Recommendations on use

- General use of AutoDAB

  The selection of the data set to be buffered and the setting of the prefetch factor should occur through the automatic caching. It is therefore recommended that you operate the performance-critical volumes or pubsets with AutoDAB.

  As a rule, the following procedure is simple and effective:

1. Determine all performance-critical applications
2. Determine all volumes/pubsets that are accessed by these applications
3. Combine all these volumes/pubsets to form an automatic ADM PFA cache area (useful if nothing is known of the access frequency of individual pubsets or Define an automatic PFA cache area for each individual pubset.

- Defining the size of the cache area

  The size of the cache area should be at least 1% of the size of the data set to be supported (files processed in parallel). With predominantly sequential processing, this value can be lower, while with predominantly random processing it may be too small. If the cache size is not sufficient, this will be reported by AutoDAB as a console message.

- Speeding up of all write accesses:

  Regardless of the locality, when a read/write (or write) cache is used, DAB speeds up all write accesses as long as there are free buffers available.
  As a result, short-term peak loads can be dealt with and fast file access times achieved. However, since DAB has to save on the storage system the write data buffered in the cache, in the case of write applications the input /output system can only be relieved when the locality of the accesses is sufficiently high.

- Home pubset

  The home pubset cannot be buffered in the main memory. Although this is possible with ADM PFA caching, it should only happen with a pure read cache.

- SM pubsets

  The data of an SM pubset cannot be buffered in a DAB cache medium with ADM PFA caching (since the storage location, i.e. the volume set, of the file can be changed dynamically).

- Database pubsets

  With pure database pubsets it may be possible to improve the “chronological locality” by selecting a smaller segment size: e.g. 8 KB instead of the default value 32 KB (/START-DAB-CACHING, parameter CACHE-SEGMENT-SIZE).

> In practice the reduced I/O times usually lead to a considerable increase in throughput and thus to preemptions of the load and the CPU utilization.
> In the case of applications which cause a higher I/O rate additional disks should be used to increase the throughput. (The use of DAB for write IOs is not always recommended.)

You will find more detailed information on using DAB in the “DAB” manual [5 (Related publications)].

Optimizing an OLTP application

This section describes the measures for response time optimization in transaction mode and with BCAM.

> In many cases the network quality has the greatest influence on the response time behavior. Lost and corrupted packets and packet transposition force the transport system to take error recovery measures, which result in packet repetitions and reduced transmission performance. A network analysis must include not only the end systems, but also all the other network components, such a switches, routers, firewalls, gateways, etc. SNMP agents can integrate the measuring components of BS2000 into such an infrastructure.

Phases of an OLTP transaction

The phases of an OLTP transaction which are relevant to performance are:

1. Network runtime up to the server
2. Waiting for acceptance of the transaction by BCAM
3. Processing by BCAM and forwarding to the TP application (e.g. openUTM)
4. Waiting for acceptance by openUTM
5. Processing time of the transaction, controlled by openUTM with the following detail phases:
   a. CPU requirement of the application / of openUTM
      (with wait for CPU allocation)
   b. I/O processing in the application / in openUTM
      (with wait for the I/O operation to end)
   c. Waiting for job acceptance by database tasks
   d. CPU requirement through database activities
      (with wait for CPU allocation)
   e. Processing the I/O operation in the database task

   These detail phases can occur several times
6. Sending the response to BCAM
7. Waiting for auf acceptance of the response by BCAM
8. Processing by BCAM
9. Transfer of the output data for network transfer
10. Network runtime up to the client

Only phases 3 - 9 can be measured using BS2000 resources (principally using openSM2, see the “openSM2” manual [18 (Related publications)]).

For phase 2, BCAM supplies an indicator of possible delays. openSM2 presents the times collected by BCAM in graphical form in the monitoring routine BCAM-Connection:

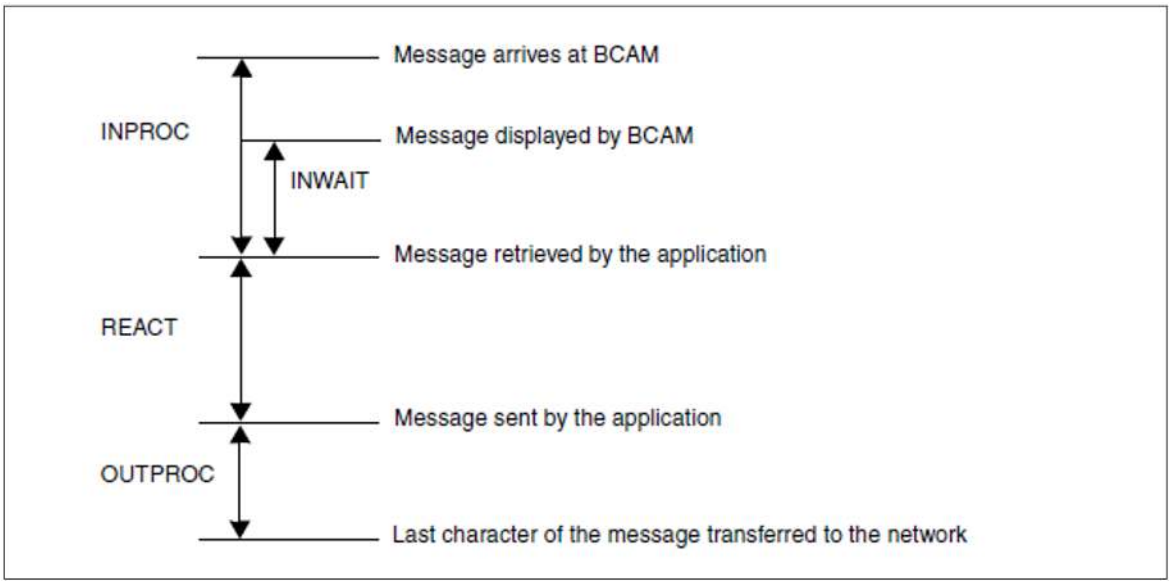


Figure 15: BCAM times

The following times are correlated:

INPROC: Phases 3 and 4

INWAIT: Phase 4

REACT: Phases 5 - 7

OUTPROC: Phases 8 and 9

Optimizing the various phases

This section deals with bottlenecks in the openUTM and BCAM environments which can lead to longer response times in the various phases. It indicates how these bottlenecks can be detected by means of measurements and how they can be eradicated or alleviated.

Phases 1 and 10: Network runtimes

General information on the LAN connection is provided in section "Network connection".

Phase 2: Waiting until the incoming message has been accepted by BCAM (Inbound Flow Control)

Mechanisms for controlling the data flow protect the transport system from being flooded by data from the partners. These ensure, on a connection-specific basis, that a (high-speed) sender is matched to the processing speed of a (low-speed) recipient.

In this way BCAM protects its memory pool, which is available to all applications and connections. During phases in which heavy use is made of the BCAM pool, BCAM prevents its partners from sending messages into the BS2000 system. This can result in increased response times.

Longer or persistently high utilization of the BCAM pool can be caused by applications which fetch their messages slowly. A pool size which is too small for the processing performance required can also result in a heavy workload on the BCAM pool.

openSM2 provides messages for analyzing the utilization of the BCAM pool, see the RESPONSETIME monitoring routine in the “openSM2” manual [18 (Related publications)].

BCAM itself enables you to see whether a bottleneck has occurred by entering the console command /BCMON and by analyzing the values of the console message BCA0B21.

Brief description of the procedure:

- Determine the maximum values set (optional) using /SHOW-BCAM-PARAMETERS PARAMETER=*LIMITS
- Activate BCAM monitoring (output of the measurement values every <n> seconds): /BCMON MODE=ON, RECORD=RES-MEM,SEC=<n>
- Describe the main output values in console message BCA0B21:
  - USED-BCAM-MIN-I: Minimum BCAM buffer allocation for incoming messages in KB
  - LIMIT-ACT: Current upper limit for the buffer allocation in KB
- Indicator for high utilization of the BCAM buffer by incoming messages:(USED-BCAM-MIN-I) * 4 > LIMIT-ACT
- As an alternative to or to complement the above-mentioned points: /BCSET THRESHOLD-MSG=ON can be used to request a warning message.

  Console message BCAB021 is then output if BCAM holds up the incoming messages for more than 5 seconds on account of a pool bottleneck (or if it cannot accept any send requests from the application, see phase 7). This warning message is disabled using /BCSET THRESHOLD-MSG=OFF.
- Optimization measure in the event of a bottleneck in the BCAM pool: The maximum permissible threshold value should be significantly increased using the /BCMOD RESMEM=<n> command (<n>=3*LIMIT-ACT would be ideal).

Phases 3 and 8: Processing the incoming/outgoing message in BCAM

The time values for these phases should be in the low single-digit range (in milliseconds).

Phase 4: Waiting for openUTM

The time which elapses between a job being issued to openUTM and acceptance of the job is referred to as the INWAIT time. It is determined for each application and is included in the INPROC time.

openSM2 records the INWAIT times (and also the INPROC , REACT and OUTPROC times) in 5 time intervals which are known as buckets. Buckets can only be set up globally, not on an application-specific basis. To simplify monitoring, intervals should be classified in such a manner that all times which are not acceptable are assigned to the last interval (overrun value).

Example

The normal response time is 100 - 200 ms (typical for large /390 servers). Short-term fluctuations up to a factor of 2 should be tolerated. However, no lengthy doubling of the response time should be tolerated.

The buckets in openSM2 should consequently be defined so that they ensure that all INWAIT times which are equal to or greater than 400 ms are counted in the overrun interval, e.g. with:

```
/SET-BCAM-CONNECTION-PARAMETER INWAIT-BUCKETS=(50,100,200,400)
```

This statement defines the buckets in such a way that all wait times < 50 ms are counted in the first interval, all wait times between 50 and 100 ms in the second interval, all wait times between 100 und 200 ms in the third interval, all wait times between 200 and 400 ms in the fourth interval, and all wait times > 400 ms in the overrun interval.

The INSPECTOR or ANALYZER component of openSM2 must then be used (for every application) to monitor the measurement values of the overrun interval. The values are output as a percentage (in relation to the time values recorded in the monitoring cycle) of this application. A percentage of 10 or higher indicates that bottlenecks occurred when the jobs were accepted by the UTM tasks.

Measuring option in openUTM

The UTM command KDCINF STAT is used to output a number of useful UTM measurement values, see the openUTM manual “Administering Applications” [20 (Related publications)]. The output value %Load provides important information for the analysis of whether the number of UTM tasks could constitute a bottleneck. This value specifies the average utilization of all UTM tasks in the last monitoring cycle. At least a short-term bottleneck is indicated by a value greater than 90 (%).

Threshold value monitoring of the number of UTM tasks using openSM2

An imminent UTM task bottleneck can be recognized from the UTM application monitoring report of openSM2. For this purpose the values for the duration of a transaction in seconds (DT), the number of transactions per second (TX), and the number of UTM tasks for the application (UT) must be ascertained from this report.

This enables the average utilization of the UTM tasks of an application to be calculated: Load (in %) = 100 * DT * TX / UT.

In INSPECTOR of openSM2 this calculated value can be subjected to threshold value monitoring. If a value of 90 (%) is exceeded, an email to this effect can, for example, be generated for systems support.

Optimization measure

The UTM command KDCAPPL TASKS=<n> enables the number of UTM tasks to be increased on an application-specific and step-by-step basis until the INWAIT times are acceptable. The optimum number of tasks ascertained in this way should be entered in openUTM's start parameter file. It will then be effective the next time the application is started. If <n> exceeds the maximum value defined in the KDCDEF run, this maximum value must be increased and a new KDCDEF run must be started.

When the number of UTM tasks changes, the number of TAC classes and the number of tasks in these TAC classes must also be taken into account. Allowance must also be made for a certain buffer for load peaks.

Note

Only if appreciably more UTM tasks than required are started can this lead to slight performance losses as a result of slightly higher utilization of the main memory and CPU.

Phases 5 and 6: Processing the transaction

The time accumulated in this phase (and in phase 7) is specified as the REACT time in the BCAM Connection Report.

Tuning I/O performance when accessing the KDCFILE

In addition to the measures relating to the hardware and in BS2000 which are described in this manual, the performance of applications with high transaction rates can also be enhanced in openUTM by means of optimized write accesses to the KDCFILE.
To do this, the page pools and/or the restart area can be exported from the KDCFILE in openUTM. These exported areas are then distributed over a number of volumes.

Example:

The page pool is to be distributed to 2 public volumes, the restart area to 4 public volumes:

```
/CREATE-FILE FILE-NAME=<filebase>.P01A, -
SUPPORT=*PUBLIC-DISK(VOLUME=<v1>,SPACE=*RELATIVE(PRIMARY-ALLOCATION=666))
/CREATE-FILE FILE-NAME=<filebase>.P02A, -
SUPPORT=*PUBLIC-DISK(VOLUME=<v2>,SPACE=*RELATIVE(PRIMARY-ALLOCATION=666))
/CREATE-FILE FILE-NAME=<filebase>.R01A, -
SUPPORT=*PUBLIC-DISK(VOLUME=<v3>,SPACE=*RELATIVE(PRIMARY-ALLOCATION=300))
/CREATE-FILE FILE-NAME=<filebase>.R02A, -
SUPPORT=*PUBLIC-DISK(VOLUME=<v4>,SPACE=*RELATIVE(PRIMARY-ALLOCATION=300))
/CREATE-FILE FILE-NAME=<filebase>.R03A, -
SUPPORT=*PUBLIC-DISK(VOLUME=<v5>,SPACE=*RELATIVE(PRIMARY-ALLOCATION=300))
/CREATE-FILE    FILE-NAME=<filebase>.R04A, -
SUPPORT=*PUBLIC-DISK(VOLUME=<v6>,SPACE=*RELATIVE(PRIMARY-ALLOCATION=300))
```

In addition, the following parameters must be modified in the MAX statement in KDCDEF: `PGPOOLSFS=2, RECBUFFS=4`.

The files defined above are then used in the KDCDEF run. In this case the KDCDEF program may modify the values for `PRIMARY-` and `SECONDARY-ALLOCATION`. Without the aforementioned commands, KDCDEF would create the files itself (without volume assignment).

The new files are used after openUTM has been restarted.

Controlling the UTM jobs by means of TAC classes

Similarly to category control using PCS (see section "PCS concept"), transactions in openUTM can be assigned to so-called “TAC classes”. These TAC classes can be controlled using two methods which cannot be combined:

- Priority control
- Process limitation

Details of job control in openUTM are provided in the openUTM manual “Generating Applications” [21 (Related publications)].

Recommendations for use:

- When the TACs are distributed to TAC classes, it must be ensured that higher-priority TAC classes are not held up by TACs from lower-priority TAC classes (use of blocking calls).
- You are recommended not to define more than 3 or 4 TAC classes.
- Priority control is used above all to ensure that long-running TACs from low-priority classes do not hinder short-running TACs from higher-priority TAC classes in the event of (short-term) overloads. As a result, high-priority transactions are given preference using all the started processes.
- The process limitation is used to ensure that long-running TACs cannot hinder short-running TACs. The advantage of this variant is that it always guarantees that enough free UTM tasks are available for new (and important) TACs.
- In addition, TACs can be controlled by specifying the RUNPRIO in the TAC statement. However, this is only recommended for highest-priority TACs and must be coordinated with the priority structure of all tasks which run in BS2000.

Further performance information on openUTM:

- Monitoring the cache hit rate

  The UTM command KDCINF STAT (see "Measuring option in openUTM" in phase 4 above) should be used (at least occasionally) to check whether the hit rate in the UTM cache’s page pool is high enough. When the value is below 90 (%), the UTM cache may need to be enlarged (see the openUTM manual “Generating Applications” [21 (Related publications)]).
- Use of UTM-F

  This UTM variant is suitable mainly for applications with “retrieval” accesses and no or only a few updates. It is used to avoid I/O operations in the case of reduced functionality (with regard to failsafe performance and restart).

Phase 7: Waiting before BCAM

This phase is included in the REACT time in the BCAM Connection Report (see phase 5).

As in phase 2, BCAM can be subject to the partner’s control mechanisms for controlling the data flow. This can be recognized from ZWR values > 0 (ZWR=Zero Window Receive) in the openSM2 report BCAM. In such situations BCAM accepts a certain volume of data to be transferred from the application, but refuses to accept further data when a threshold value is exceeded.

BCAM also delays the acceptance of data in phases in which the BCAM pool is heavily utilized. Lengthy or permanently high utilization of the BCAM pool can be caused by connections which are subject to data flow control. A pool size which is too small for the processing performance required can also result in a heavy workload on the BCAM pool.

As in phase 2, BCAM offers a monitoring option.
If the BCA0B21 message displays high values for USED-BCAM-MIN-O, the BCAM pool should be enlarged, as in phase 2.

Phases 8 and 9: Delay in transfer to the network

The time in phase 8 (100% BCAM processing) is in the single-digit millisecond range. The times for phases 8 and 9 are combined in the OUTPROC time. You are recommended to monitor the OUTPROC time like the INPROC time (see "Phase 4: Waiting for openUTM"). High values in the overrun interval indicate possible network performance problems (e.g. due to malfunctions in routers or switches).

High-performance loading of programs

This section will first provide a general description of the processes involved in loading a program/product.

This forms the basis for a description of the measures which should be considered by users in order to optimize the load time for a program/product.

- General notes on improving loading speed
- Structural measures for reducing resource requirements
- Improving the load speed for C programs
- Use of DAB

It is generally assumed that the programs or modules are stored in PLAM libraries. This means that the programs /modules are linked with BINDER and loaded with /START-EXECUTABLE-PROGRAM.
The information provided here relates only to working with LLMs. No explicit reference is made to the linking and loading of phases, although a number of the measures described for speeding up the loading process can certainly also be used in such cases.

Basic stages involved in linking/loading a program/product

Diagram of the entities involved (the LLMs to be loaded are located in the library specified in the load call):

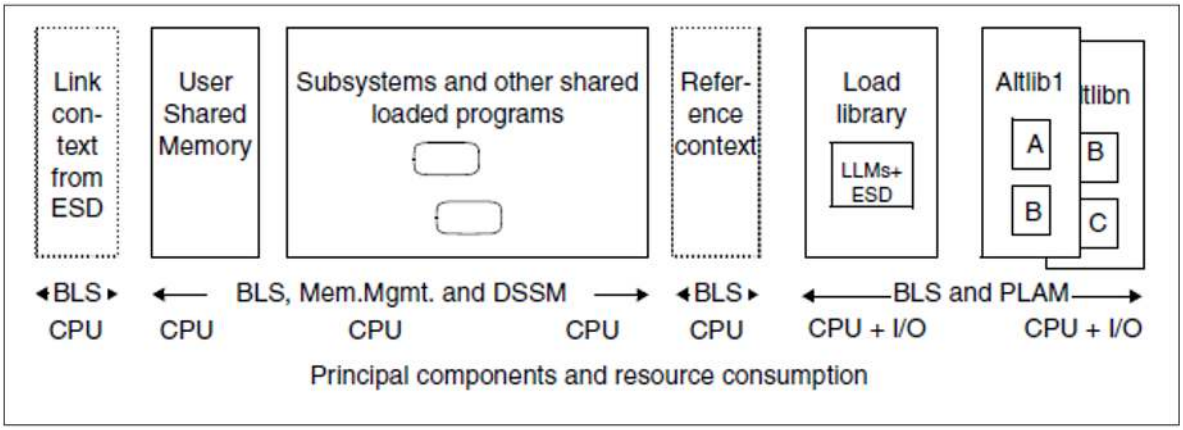


Figure 16: Stages involved in linking/loading a program/product

External references are resolved from left to right, i.e. the system searches the entities in the following sequence:

1. Link context (according to the ESD information in LLM); this is empty if the program is a standalone program (with /START-EXECUTABLE-PROGRAM)
2. User Shared Memory (= memory pools); a number of memory pools can be specified when /START-EXECUTABLE-PROGRAM is issued (default: deactivated)
3. (Pre-loaded) DSSM subsystems and other shared loaded programs (default: activated when /START-EXECUTABLE-PROGRAM is issued)
4. Reference context in memory: this is created dynamically during loading and is used to satisfy external references which could not be resolved in the link context
5. Library specified in the load call
6. alternative libraries (Altlibs) in the specified sequence.

General notes on improving loading speed

- All the PLAM libraries involved should be organized as efficiently as possible, i.e.
  - they should contain no redundant LLMs
  - they should be completely reorganized, so that the LLMs they contain are not split into an unnecessary number of extents
  - the library itself should comprise as few extents as possible
  - the library should possibly be created using (STD,2).

  All these measures reduce the number of I/Os and improve runtime behavior. The effectiveness of these measures is, of course, dependent on the original structure. A library created with (STD,2) will be larger than the original library by about 1 KB per member. The reduction in I/O operations will be greater, the more members the library has, provided that these members are not too small. Runtime improvements can then amount to over 10%.
- The /START-EXECUTABLE-PROGRAM comand should always be used when loading an LLM object. This initiates more effective BLS search procedures, allowing a reduction in CPU time of about 10%.
- If modules have the READ-ONLY attribute, one slice should be generated for read-only modules and one slice for read/write modules. This speeds up the loading process by reducing both CPU requirements and I/O operations.
- The size of the LLMs and thus the PLAM overhead are both reduced if all symbols not required for subsequent BINDER runs are set to “invisible” with the BINDER statement

  `//MODIFY-SYMBOL-VISIBILITY SYMBOL-NAME=...,SYMBOL-TYPE=..., SCOPE=..., VISIBLE=*NO,...`
- If public/private slices are to be used, format 2 should be used in order to reduce the CPU requirements of BLS and DSSM (see the BINDER statement //SAVE-LLM ...). FOR-BS2000-VERSIONS=*FROM-V11).

  The operand SUBSYSTEM-ENTRIES in the BINDER statement //START-LLM-CREATION allows you to specify a symbolic name for the links between the private and public slice. This name must also be defined in the DSSM catalog. (DSSM is called for every link with the old format.)
- You should ensure that the LOGGING parameter in the DSSM parameter file is **not** set to ON (default: OFF). ON would cause DSSM data to be logged when subsystems are loaded.
- RESOLVE-BY-AUTOLINK link from **several** libraries:
  The BINDER statement //RESOLVE-BY-AUTOLINK LIBRARY=(A,B,C) can be replaced by the following specification, which improves performance:

  ```
  //RESOLVE-BY-AUTOLINK LIBRARY=A
  //RESOLVE-BY-AUTOLINK LIBRARY=B
  //RESOLVE-BY-AUTOLINK LIBRARY=C
  ```

  During dynamic loading, this means, for instance, that new links which need to be resolved from library B are only searched for in library B (and possibly library C), but not first in library A (and then in B and possibly C).

Structural measures for reducing resource requirements

Unfortunately, it is rarely possible to quantify precisely the effect of each of the measures described below. The effects depend to a large degree on the structure of the load objects, the subsystems and the libraries. One can, however, generally assume that a gain of at least 10% can be achieved with each of the measures.

Reduce the number of libraries or optimize their contents

- Merge alternative libraries

  If alternative libraries are merged to form a single library (this could even be the specified load library), this results in a considerable reduction in search operations in BLS/PLAM, which in turn leads to a saving in I/O operations and CPU time which is sometimes considerable greater than 10%.

  Merging is generally only recommended if no entries or CSECTs exist with the same name. If this is not the case, care should be taken when the libraries are merged, that the “correct” CSECTs or entries are included.

- Link in modules from libraries permanently

  If the contents of the alternative libraries and the specified library never or only rarely change, it may be advisable to link the modules into the load object permanently. This reduces the CPU requirements on loading. On the other hand, static linking means that the load object will increase in size, thus generally causing more I/O operations, which in turn increases the runtime.
  This measure makes sense, therefore, if only small modules are involved and if the alternative libraries contain a particularly large number of entries.

  If you adopt this strategy for improving performance, you must, of course ensure that the link procedure is repeated each time the alternative library is modified.

- Mixed strategy

  If, for instance, the contents of only one of the alternative libraries is modified regularly, it may be advisable to adopt a mixed strategy using both of the methods described above.

Restricting dynamic loading of shared code

- Do not dynamically load/link shared code

  Many programs do not need to dynamically load/link from a subsystem/shared programs or user shared memory. These programs should be started as follows:

  ```
  /START-EXECUTABLE-PROGRAM ...,
     DBL-PARAMETERS=*PARAMETERS(RESOLUTION=*PARAMETERS(SHARE-SCOPE=*NONE)),
     ...
  ```

  This prevents BLS from calling DSSM to search the subsystems for the entry, resulting in an often considerable reduction in CPU time.

- Only dynamically load from system memory

  This is the default. It is not possible to restrict the search for the entry performed by DSSM to specific subsystems or to shared programs only.

- Only dynamically load from user shared memory

  The following parameter of /START-EXECUTABLE-PROGRAM permits limitation to used shared memory:

  `DBL-PARAMETERS= *PARAMETERS(RESOLUTION=*PARAMETERS(SHARE-SCOPE=*MEMORY-POOL(...)))`

  It is also possible to further restrict the operation to specified memory pools.

- Preload user shared memory

  As far as possible, all shareable sections of the applications in use should be preloaded (e.g. FOR1-RTS). This means that they are loaded only once and only need to be linked (with little overhead) for each further load operation. This strategy generally results in a saving of considerably more than 10% in I/O operations and CPU time.

Eliminating ESD information

- Complete elimination

  This measure only applies for standalone programs. These are LLMs which are not called by other programs and which are completely prelinked. For programs of this type, the load operation can be accelerated if the ESD and other BLS information tables are eliminated using the appropriate BINDER call. Call BINDER as follows:

```
/START-BINDER
//START-LLM-UPDATE LIB=<original-lib>,ELEMENT=<elem-name>
//SAVE-LLM LIB=<new-lib>,TEST-SUPPORT=*NO,MAP=*NO,
           SYMBOL-DICTIONARY=*NO,LOGICAL-STRUCTURE=*NO,
           RELOCATION-DATA=*NO
//END
```

  This parameter specification means that only those BLS structures are created which are required for loading a simple program, thus reducing the size of the object. This saves CPU time and I/O operations when the program is loaded. All the information required for other processing is then no longer available.

  This means that

  - these parameters cannot be used for load objects which are split into a number of slices
  - no LLM updates can be carried out on an object of this type.
  - relocatable objects (e.g. as part of a runtime system) cannot be created.

  The (supplementary) parameter setting REQUIRED-COMPRESSION=*YES should not be selected, since, although up to 10% of I/O operations can be saved, approx 40% more CPU time is required.

- Partial elimination

  When merging LLMs or sub-LLMs to form a prelinked module with a single CSECT, you should specify what ESD information is to remain in the external address book with the BINDER statement //MERGE-MODULES. This can result in a considerable reduction in BLS overhead during loading.

Improving the load speed for C programs

You can improve the load speed of C programs by linking the few modules which cannot be preloaded from the C runtime system to the compiled program. Use the following supplementary BINDER statement:

```
//RESOLVE-BY-AUTOLINK LIB=<Partial-Bind-Lib>
```

The <Partial-Bind-Lib> is installed under the name SYSLNK.CRTE.PARTIAL-BIND. It is also necessary to make external names invisible in order to suppress duplicate symbols:

```
//MODIFY-SYMBOL-VISIBILITY SYMBOL-NAME=*ALL,VISIBLE=*NO
```

These two statements result in a considerable reduction in CPU time and particularly in runtime when loading C programs. The load object is only a few pages (approx. 18) longer than the compiled program.

> The CRTEC and CRTECOM subsystems must be loaded to allow CRTE to be loaded in the shared code area.

Use of DAB

If the linked object and/or the libraries used are very large, with the result that a large number of I/O operations are required during loading, you should check whether it is possible or necessary to use DAB (Disk Access Buffer).

DAB permits such a considerable reduction in I/O operations on frequently accessed files that in extreme cases the runtime depends only on the CPU requirements.

A cache area must be selected for the load object which is the same size as the load object itself. This is particularly simple if the object is the only member of the library. You can then issue te following to reserve a main memory area with the same size as the library:

```
/START-DAB-CACHING AREA=*FILE(FILE-AREA=<lib>),CACHE-SIZE=*BY-FILE
```

This is filled the first time the object is loaded, with the result that no time-intensive I/O operations are required for any subsequent load operations.

The cache areas for the libraries can generally be created smaller than the total size of the libraries, since it is generally the case that not all of the modules contained in the libraries will be dynamically loaded. In this case, the cache size must be specified explicitly (in KB or MB) when /START-DAB-CACHING is issued. It is possible to monitor the cache hit rate for each cache area using openSM2 or /SHOW-DAB-CACHING. The hit rate should be at least 80%. If this is not the case, the size of the cache area must be increased.

System and application analysis with openSM2

This chapter describes how you can execute a performance analysis with openSM2:

Basic procedure

Before discussing bottleneck analysis in detail we should recall the basic problems involved in meeting performance expectations:

Performance expectations of the individual user

The performance expectations of the individual user are temporal in nature (response time, elapsed or dwell time). The time taken to process the user's requests to the IT system has a direct effect on his/her work pattern and hence on his/her “productivity”.

Processing time comprises service time and the time spent waiting for the relevant resources. While service time reflects the immediate resource requirements, wait time depends on the workload placed on the resources. Therefore, the lighter the resource workload, the easier it is to satisfy the performance expectations of the user.

Performance expectations of the organization running the server

For economic reasons, the performance expectations of the organization running the server are directed towards attaining maximum “system throughput” (high transaction or job throughput rate).

Therefore, to meet the performance expectations of both the individual user and the systems support, it is always necessary to reach a compromise.

Understandably, investment planners also demand the fullest possible utilization of resources. However, in actual practice, various standard values have emerged for using resource capacity. These values should never be exceeded unless absolutely necessary. In most cases they do not represent technical limits. If exceeded, they lead to inefficient utilization of the IT system and make it more difficult to meet the performance expectations of the individual users.

Therefore, before carrying out measurements, it is essential to clarify which performance expectations are not being met:

- System-oriented performance expectations (the system throughput leaves something to be desired, i.e. the performance of the **entire IT system** is unsatisfactory).

  Indications include:

  - excessively low transaction rate (as an immediate offshoot of generally poor response time behavior)
  - excessively low throughput rate (dwell time for jobs generally too long).
- User-oriented performance expectations (the performance expectations of **individual users** are not being met).

  Indications include:

  - excessively long response time for certain types of transaction
  - excessive dwell time for particular jobs.

If the performance problem is **system-oriented**, the causes most likely lie in an overload on one or more resources.

If **user-oriented**, it is essential to find out the reasons why delays arise when handling special load requests.

> As the use of additional monitoring programs leads to increased resource requirements (principally CPU time), it should be ensured that the monitoring programs are disabled or removed from the start procedure after a successful measurement.

Selecting monitoring cycles

Normally, measurements for bottleneck analysis are made at the moment of peak workload. To facilitate analysis, as much measurement data as possible should be collected. The measurement period should be from 30 minutes up to a maximum of one hour.

An analysis interval of 15 minutes should be used as the basic reference value for judging the measurement values (average value for 3 measurement value subintervals of 5 minutes each).

The collection of measurement data by SM2 is partly event-related and partly done by sampling. To ensure the reliability of the samples, it is necessary to collect a certain number of measurement values per analysis subinterval (minimum 1500). A sampling period of 400 milliseconds is recommended.

For each monitoring cycle, SM2 forms the mean value of the collected measurement values. In order to illuminate the range of deviation per analysis subinterval, it is helpful if each analysis subinterval is composed of two or more monitoring cycles.

Recommended measurement parameters

- sampling period (SAMPLING-PERIOD): 400 milliseconds
- monitoring cycle (OFFLINE-PERIOD): 60 seconds
- analysis subinterval: 5 minutes
- Monitoring period: 0.5 - 1 hour

With bottleneck analysis measurements, it is preferable to **output the observed values to the measurement file** rather than display them on the screen. Consequently file analysis with ANALYZER is preferred to the online analysis with openSM2 Manager or INSPECTOR.

Rough analysis and detailed analysis

When using ANALYZER (or SM2R1) for analysis purposes, it is best to proceed step by step:

1. Rough analysis

   Bottleneck search by means of pinpointing (analysis subinterval = 15 minutes)

   This locates those analysis subintervals in which the recommended standard values are exceeded.
   The **automatic performance analysis** available in ANALYZER (or SM2R1) provides valuable help in tracking down bottlenecks. This function compares the measurement values collected in the SM2 file with the standard values recommended in the following section, and issues a corresponding message to the user. It is important that only critical periods of productive operation are used for automatic performance analysis.
2. Detailed analysis

   Investigate the critical analysis subintervals (15 minutes) by reducing the analysis subinterval (TIME-STEPS) to the size of the monitoring cycle (OFFLINE-PERIOD) in order to highlight the range of deviation in the measurement values.

To facilitate the assessment of the measurement findings, it should be ensured that the measurement values come from periods in which performance expectations were satisfied as well as from periods in which they were not satisfied. Finding the cause of the problem is much easier when both cases are compared.

The following section attempts to simplify the interpretation of measurement values by suggesting standard values for resource workloads.

Selecting monitored values

The selection of monitored values depends on what you wish to examine. A distinction is made between the following:

- Monitored values for examining the system performance
- Monitored values for examining application performance

Monitored values for examining the system performance

For detecting resource overloads the following data is required:

- workload values: CPU, channels, devices (disks), main memory
  With disks, SM2 sees exclusively logical volumes.
- number of I/O operations per device
- number of tasks per device queue
- Working set requirement for each task category

To make clear the dependence of response time behavior on resource workload, it is helpful to maintain the following statistics in parallel:

- Connection-specific response time statistics (monitoring program BCAM-CONNECTION)

The use of the following monitoring programs is also recommended:

- Extended system statistics (monitoring program SYSTEM)
  Among other things, this monitoring program provides information on dilations and dwell times in the system queues for each category, as well as information on the hardware service time (hardware duration) and software service time (software duration) for DMS and paging I/O operations.
- Extended device statistics (monitoring program SAMPLING-DEVICE) The monitoring program is started automatically when SM2 is called, but the hardware and software service times for each volume are not recorded. If monitoring is also to include these values, the monitoring program must be restarted with the following statements:

  ```
  //SET-SAMPLING-DEVICE-PARAMETER DISK-SERVICETIME=*ON
  ```

  ```
  //CHANGE-MEASUREMENT-PROGRAM TYPE=*SAMPLING-DEVICE.
  ```
- Channel statistics (monitoring program CHANNEL-IO)
  This monitoring program is used to ascertain the number of I/O operations and the number of PAM blocks for each channel.
- Catalog statistics (monitoring program CMS)
  This monitoring program provides information on the number, type and duration of the accesses to catalog entries on each pubset.
- VM2000 statistics (monitoring program VM)
  In VM2000 operation this monitoring program should only be started in the monitoring system (but there by default). In the monitoring system it supplies the CPU share of all guest systems and on /390 servers the share of hypervisor active and idle.

Monitored values for examining application performance

Besides the above-mentioned global system workload values, additional data is required for detecting delays. To begin with, the corresponding task selection can be carried out using the periodic task statistics (monitoring program PERIODIC-TASK).

The following monitored variables are necessary for further analysis:

- resource depletion per task (TASK monitoring program)
  The monitoring program provides:
    - CPU time requirement
    - Number of SVC calls
    - the number of I/O operations and I/O times, broken down according to the volumes defined in the monitoring program
    - the duration of wait states
    - Paging activities
    - the message lengths per terminal I/O.

  An additional user task measurement with subsequent analysis by SM2-PA (program analyzer) is recommended for analyzing the CPU requirements of individual tasks in more detail using program counter statistics and SVC statistics.
- SVC statistics (SVC monitoring program)
  The monitoring program provides the frequency of each SVC call classified by processor states TU and TPR.
- DAB hit rate (DAB monitoring program)
  The monitoring program provides the number of accesses per specified DAB subarea.
- wait times in front of individual devices (SERVICETIME monitoring program)
- ISAM pool hit rate (when NK-ISAM is used; ISAM monitoring program) The monitoring program provides the number of accesses per specified SAM pool.
- number of access operations to particular files (FILE monitoring program)

- UTM statistics (UTM monitoring program)
  The monitoring outputs data for each application, including the dwell time in UTM, the dwell time broken down according to database share, TAC-class wait time and time for distributed processing. In addition, the program provides average consumption values for CPU and I/O operations in UTM and in the database system (only SESAM/SQL and UDS), as well as the number of database calls for each dialog step or asynchronous process.

  UTM values are output under the following conditions:

  - the UTM-SM2 subsystem must have been started
  - UTM-SM2 is started automatically when the operand MAX SM2=ON is specified for the KDCDEF run. UTM-SM2 is started subsequently via the UTM administration interface with KDCAPPL and SM2=ON, together with the readiness to pass values.
  - the SM2 monitoring program UTM must have been activated
  - the UTM application must have been set to supply data to SM2.
    The passing of values is controlled by means of the MAX SM2=ON/OFF/NO operand in the KDCDEF run as follows:
    - if SM2=OFF (default value) is set, the passing of values for each application can be activated at a later point in time via the UTM administration interface with KDCAPPL and SM2=ON;
    - if SM2=ON is set, no further administration is required;
    - if SM2=NO is set, the passing of values is generally prevented and cannot be activated at a later point in time.
  - database-specific consumption values (I/O operations, CPU time) are output only under the following conditions:
    - BS2000 accounting is activated;
    - The UTM accounting record UTMA is enabled (with /MODIFY-ACCOUNTING-PARAMETERS, ADD-RECORD-TYPE=UTMA).
    - UTM accounting is activated (KDCAPPL, parameter ACC=ON);
    - in SESAM/SQL, the recording of statistics is activated with the statement ACC,TP=ON,CPU.

  Further information is provided in the manual “openUTM - Concepts and Functions” manual [19 (Related publications)] and in the manual “Using openUTM Applications under BS2000” [22 (Related publications)].

- Database statistics
  The SESAM-SQL and UDS-SQL monitoring programs supply monitored data about the SESAM/SQL and UDS /SQL database systems.
  - Prerequisite in SESAM/SQL:

    To transfer statistical data from SESAM/SQL to openSM2, start SESMON in batch mode:

    ```
    /START-SESAM-PERF-MONITOR
    //SET-MONITOR-OPTIONS ...,OUTPUT=*SM2
    ```

    When OUTPUT=*SM2, only one DBH can be specified for each SESMON instance. A new SESMON instance must be started for each further DBH for which data is to be output.

    The interval at which SESMON transfers the data to openSM2 is automatically set to approx. 30% of the SM2 monitoring cycle. It cannot be set manually.
  - Prerequisite in UDS/SQL:

    Transfer of the statistical data from UDS/SQL toopenSM2 is initiated either when the UDS monitor is started using the MEDIUM=S,n statement or, during ongoing operation, with the /INFORM-PROGRAM MSG='ADD MEDIUM=S'n' command. It can be terminated again using the /INFORM-PROGRAM MSG='FINISH MEDIUM=S' command.

    n is used to define the cycle in seconds (5 n 999) in which the UDS monitor transfers the data to SM2. It should be set to considerably less than the monitoring cycle set in SM2 so that data can be transferred several times in an SM2 monitoring cycle.
  - The monitored values are supplied asynchronously to openSM2 from the database systems and apply for cycles defined for one or more database systems which do not need to match the SM2 cycle exactly. Here there can be differences in both the length of the cycles and in the temporal displacements between the database system and the SM2 cycles.

    The length of the database system cycle or cycles is used to standardize the monitored values to one second. The values are therefore exact, but they only match the SM2 cycle to a certain degree.
- Information on TCP/IP connections (TCP-IP monitoring program)
  The IP and PORT number is supplied for each connection together with the number of transport service data units transferred.
- Number of accesses and data transferred of network devices
  (monitoring program SAMPLING-DEVICE)
- Communication in the computer network (MSCF monitoring program) Data is supplied on the communication of the local computer with other computers.
- Data on basic functions in the HIPLEX network (NSM monitoring program)
- Statistics on lock requests from TU, TPR and NSM
  (DLM monitoring program)
- POSIX statistics (POSIX monitoring program)
  Supplies measurements on the use of various POSIX functions.

> ¡ Storage systems of type ETERNUS DX/AF can be monitored with the openSM2 Manager.

Examining system performance with openSM2

First, for ease in understanding the suggested standard values for resource workloads, it is helpful to quantify the work of BS2000.

System time (SIH time) is required primarily for handling the following events:

- controlling multiprogramming operation
- carrying out DMS I/O operations
- carrying out paging.

The following standard values apply to a CPU working at full capacity:

The SIH portion should not be more than 20% (/390 servers) or 10% (x86 servers), even in the case of heavy DMS I /O activity.

With normal loads, the time taken up for controlling multiprogramming operation amounts to 5 - 10% (/390 servers) or 3 - 5% (x86 servers) of the full capacity of the CPU involved.

With the usual workload and a balanced configuration, the following SIH portions apply for

- executing DMS I/O operations: 3 - 6% (/390 servers) or 2 - 4% (x86 servers)
- executing paging: max. 2% (/390 servers) or max. 1% (x86 servers)

When interpreting the values measured with SM2, the user is advised to proceed in the order of the sections which follow.

I/O load

The following standard values exist for the I/O load:

- Standard values for the DMS I/O rate
- Standard values for the channel workload
- Standard values for the device workload

Reference values for the DMS I/O rate

In section "Reference values for BS2000 servers" you will find guidelines for the recommended maximum I/O rate as a function of the CPU type. These values are governed by the rule that no more than a certain share of CPU capacity (for /390 servers 15 - 25% or for x86 servers 12 - 16% in processor state SIH+TPR) should be used for initiating and handling I/O operations. In individual cases, the recommended values may be considerably exceeded. This can by all means be a meaningful application. In such cases an individual check should be made to determine whether the violation of the recommended values causes performance problems.

ANALYZER reports for assessing the DMS I/O rate

Report group	Report	Meaning
IO	IOs for device classes	Number of DMS I/O operations/s
CATEGORY-IO	IOs for category	Number of DMS I/O operations/s per task category

Storage systems of type ETERNUS DX/AF can be monitored with the openSM2 Manager.

Tuning approaches for reducing the DMS I/O rate

The points below are not arranged according to general importance.

- Reorganize the ISAM files Lower the number of index levels by means of organizational measures (dividing the collected data, checking for data obsolescence).

- Switch to NK-ISAM files, with the resultant savings on I/O operations through the use of self-configuring ISAM pools or ISAM pools which can be selected on a user-specific basis. Use of the ISAM pool can be monitored with the aid of reports of the report group ISAM-POOL or ISAM-FILE:

Report	**Monitoring variable**	**Meaning**
Fix operations	Number of fix-hit operations	Number of ISAM accesses/s which were resolved from the ISAM pool
	Number of fix-IO operations	number of ISAM accesses per second that resulted in an I/O operation to disk (read access)
	Number of fix-wait operations	number of ISAM accesses per second during which the system had to wait for pool pages to become free
Slot operations	Number of reserve slot wait operations	number of wait states per second resulting from bottlenecks in the internal pool management which were caused by the pool being too small
Used pages	Number of total pages	number of PAM blocks in the ISAM pool which (after deducting the space required for management data) are available for buffering data
Index operations	Number index operations	total number of index accesses
	Number index hit operations	share of index accesses dealt with from the ISAM pool

  The ISAM pool is the correct size (for details of calculating the size, see the “DMS Macros” manual [8 (Related publications)]) when the following applies:

  #FIX-HIT OPERATIONS / (#FIX-HIT OPERATIONS + #FIX-IO OPERATIONS) >= 0,85

  The following must also be true:

  Number of fix-wait operations = 0 Number of reserve slot wait operations = 0

  In order to generate an ISAM pool that is of sufficient size, enough main memory capacity must be available to prevent the effect of reducing the number of DMS I/O operations being compensated for by an increase in the number of paging I/Os.
- Raise the blocking factor (for both disk and tape processing) Increasing the block size is always advisable in the event of sequential access. In the case of random-access processing (PAM, ISAM), this condition must be checked on an individual basis. Increasing the block size reduces the number of I/O operations and thus offloads the device and the CPU, but not the channel.
- Increase the size of the internal data buffer for database applications (please refer to the relevant database manuals) Whenever an increase in the size of the internal buffers leads to a corresponding increase in the size of the user address space, adequate main memory capacity must also be provided.
- Reduce the I/O rate to KDCFILE in UTM applications by increasing the size of the UTM cache area (see the “openUTM Administering Applications” manual [20 (Related publications)]).

- Use the software product DAB (ADM-PFA) By reading in up to 16 PAM blocks at a time (32 PAM blocks with AutoDAB), the number of I/O operations required is significantly reduced when processing is mainly sequential. Prerequisites are sufficient main memory and a corresponding CPU reserve capacity (CPU intensity increases when the physical I/O delay is reduced). See the “DAB” manual [5 (Related publications)] for information on using this function. The hit rate on the specified DAB subareas can be checked with the help of the reports “Reads for internal area” und “Writes for internal area” of the report group DAB. If AutoDAB is used, /SHOW-DAB-CACHING provides an overview of the files which are currently subject to caching. See also the section "ADM PFA caching".
- Use HIPERFILEs (see section "Working with HIPERFILEs and DAB").

If the common value for the DMS I/O rate is less than the suggested standard value (see the section "Reference values for BS2000 servers"), the next step is to check the workload on the channels and the attached peripheral devices.

Reference values for the channel workload

In the case of channels with ETERNUS DX/AF storage systems connected, the workload should not exceed 60%. This applies for both native operation and operation under VM2000 (see also information on the "Report group CHANNEL").

Reports of the report group CHANNEL for assessing the channel workload

Report	Meaning
Data transfer	Number of PAM blocks transferred
IOs	Number of I/O operations per channel

Tuning measure for channel overload: reconfiguration

As a rule the load of one or more LUNs is distributed equally over the channels assigned to them. Temporary overloads (15 minutes) for application-related reasons can be tolerated. In the case of overloads lasting for a longer time, more channels must be provided for the relevant LUNs, see section "FC connection".

Reference values for the device workload

Disks (more precisely: logical volumes)

If volumes are shared by two or more users, the size of the workload is a key criterion for the wait time in front of each volume. With online applications, the wait time should not be more than one third of the hardware service time.

Therefore it is essential to determine whether the volume concerned is being shared by two or more users (or tasks) or whether it is assigned permanently to one user.

Magnetic tape cartridge units

Usually magnetic tape cartridge units are permanently assigned to a single task. Therefore, a workload of 100% is feasible.

The user should make sure here that the workload is not made artificially heavy by the “processing” of an excessive number of inter-block gaps.

ANALYZER reports for assessing device workload (volumes)

Report group	Report	Meaning
CATEGORY-IO	Duration of non paging IOs for category	Hardware and software service time for DMS I/O operations/s per task category
	Duration of paging IOs for category	Hardware and software service time for paging I/O operations/s per task category
DISK	IOs	Number of I/O operations per second and volume
	Queue length	Average length of queues for the individual volumes
	Data per IO	Number of PAM blocks per I/O operation and volume
	Time	Hardware and software service time for DMS I/O operations/s per volume
SERVICETIME	Duration of IOs for device	Hardware service time for I/O operations per volume with detailed breakdown and the wait time before the volume

ℹ Storage systems of type ETERNUS DX/AF can be monitored with the openSM2 Manager.

Tuning measures for device overloads (volumes)

The first thing to do is always to check the hardware service time (report “Time” of the report group DISK with the SAMPLING-DEVICE monitoring program switched on), taking into account the number of transferred PAM blocks per I/O operation.

In very rare cases the analysis must be continued as follows:

1. If the hardware service time is only negligibly longer than the DEVICE CONNECT TIME (report “Duration of IOs for device” of the SERVICETIME monitoring program or report group), you are dealing with read hits or fast write hits.

   The overloading of the logical volume is caused by the frequency of the accesses and leads to wait times for the tasks in front of the volume. In this case, it makes sense to use the PAV function. If you do not, it is necessary to relocate particularly frequently used files to other, less busy volumes (if possible also on a different physical disk drive).
2. If the hardware service time contains a significant share of DEVICE DISCONNECT TIME (report “Duration of IOs for device” of the report group SERVICETIME), the cause may be one of the following:
   - Insufficient support for the read caching due to too small a cache.
   - If there is a lack of “breathers” during continuous write loads, cache contents cannot be saved to disk in the interim.
   - A competition situation between logical volumes on the same disk drive (check the volumes per disk drive with the follwoing command of the SHC-OSD software product:

     ```
     /SHOW-STORAGE-DEVICE-CONFIG UNIT=*BY-VOLUME (...), INF=*PHYSICAL .
     ```
3. A share of REMAINING SERVICE TIME (report “Duration of IOs for device” of the report group SERVICETIME) in the hardware service time indicates the following:
   - Dilation of the hardware service time by other guest systems under VM2000. In this case, you should check the setting for the CPU quotas.
   - Use of REC (see "Performance behavior in the case of synchronous remote replication" (Replication: volume-based mirroring for storage systems)).
     A REMAINING SERVICE TIME part, that is the same size as the DEVICE CONNECT TIME, indicates a number of “Remote Data Links” that is too low.

Paging

Main memory management is characterized by the paging mechanism, which allows the efficient use of a relatively small main memory in relation to the virtual address space requirements.

Because paging both causes time delays and adds to the workload on the processor and the disk peripherals, the paging rates specified in section "Standard values for BS2000 servers" should not be exceeded.

You should also make sure that a sufficient number of volumes is available for performing the paging I/O operations. Around 30 paging I/O operations per second can be implemented for each volume (given a workload of 10% and a read hit rate of 70%).

Clues as to the cause of the paging rate are provided by the ratio of the memory used by the system (SWS) to the number of main memory pages available for paging (NPP).

See also the notes on "Reports "Page frames" of the report group MEMORY and "Main memory utilization" of the report group WORKING-SET".

> The tables in the section "Standard values for BS2000 servers" contain representative main memory configurations for various servers. The actual main memory required depends on the workload. TP and interactive loads usually have a considerably higher main memory requirement than batch loads.

CPU utilization

As a rule, the resource CPU is used simultaneously by many tasks. Here, too, the size of the workload is a key criterion for the amount of time a task has to wait for the CPU.

The average hardware service time of the resource CPU per user request (e.g. between two DMS I/O operations) is relatively small as compared to the execution of I/O operations.

Consequently, the wait time in front of the resource CPU has considerably less effect on response or dwell time than the wait time in front of disk drives (volumes). Therefore, a heavier workload is feasible as compared to shareable volumes. However, for the **main application** a workload of **70%** should not be exceeded (with multiprocessor systems - depending on the number of CPUs - a maximum workload of 90% can be tolerated). When multiple guest systems are used, generally a 5 - 15% higher overall workload than in native operation is possible.

In addition, the task scheduler PRIOR allows tasks to be given precedence by means of priority assignment. This makes it possible to use part of the remaining **30%** (or 10%) by running **low-priority tasks** (with correspondingly long wait periods in front of the resource CPU). Utilization of 100% of the CPU without adversely affecting the main application will only be possible in rare instances.

ANALYZER reports for assessing the CPU workload

Report group	Report	Meaning
CATEGORY-CPU	CPU utilization (TU+TPR) for category	CPU workload in the TU and TPR processor states for each task category
CPU	Utilization real	CPU workload in the TU, TPR and SHI processor states (actual values). When VM2000 is being used, this report is important for determining the CPU share of the guest system vis-à-vis the overall system
	Active logical machines	Number of active logical machines (number of CPUs used by BS2000)
	Sum SVC calls	Sum of all SVC calls in the TU and TPR processor states

High CPU utilization

CPU workloads are considered heavy when they exceed 90%.

A heavy CPU workload does not necessarily mean that the resource CPU is the source of the bottleneck.

If you have determined that the rate of CPU utilization is high (the number of CPUs used by BS2000 must match the number of actually available CPUs), you must first check whether the CPU resources are being used efficiently. You can do this by studying the relationship between the workload in processor states TU and TPR and the workload in processor state SIH (see the notes on "Report group CHANNEL").

Try to achieve the following ratio:
TU + TPR > 3 * SIH (/390 server) or TU + TPR > 7 * SIH (x86 server)

If the ratio is poor, the investigation should be conducted with the objective of reducing the SIH time, which is essentially nonproductive for users:

- Checking the level of the DMS I/O rate (see section "I/O load").
- Checking the level of the paging rate (see section "Paging").
- All calls to BS2000 are handled via SVCs. The processing of SVC interrupts involves corresponding overhead in the system, which is charged to the account of the calling task.

> ¡ If the number of SVC calls per second shown in the tables in the section "Standard values for BS2000 servers" is reached, the overhead required for the interrupt analysis and termination handling of the system routine (i.e. not the resources used for the system routine itself) amount to approx. 5% of CPU time.

The total number of SVCs can be determined via report “Sum SVC calls”. The TASK monitoring program of SM2 can be used to count the number of SVCs for each task. Further classification in the form of SVC statistics is possible by means of user task measurement for selected tasks followed by SM2-PA analysis.

The following relationship helps to assess whether the resource CPU is forming a bottleneck in conjunction with heavy CPU workload **and** efficient utilization:

Wait time for using the CPU / hardware service time of the CPU

Standard value

The resource CPU can be regarded as overloaded when the main application yields the following ratio or the background application yields a correspondingly higher ratio.

Wait time for using the CPU / hardware service time of the CPU > 3

If long wait times for use of the CPU occur for the main application, there could be one of two reasons:

- the system control parameters (in particular the priority assignment) were not set in accordance with the actual requirements;
- the total CPU time requirement of all tasks is too high (critical case).

Further information is provided by the dwell time percentages in the system queues of the SM2 monitoring program SYSTEM (evaluation with the SM2R1 statement //PRINT-QUEUE-TRANSITION).

Example 1

CPU utilization	96 %
TU portion:	36 %
TPR portion:	45 %
SIH portion:	15 %

Main application: TP task category

Dwell time in 01 (%) / Dwell time CPU (%) = 5.4% / 1.2% = 4.5

background application: Batch task category

Dwell time in 01 (%) / Dwell time CPU (%) = 2.1% / 1.4% = 1.5

By choosing a poor setting for the system control parameters, the background application BATCH does not run in the background as intended. This forces the tasks in the main application TP to wait too long for use of the CPU.

Remedy: Assign a higher priority to main application TP or use PCS (see section "PCS concept").

> The CPU workload **cannot** be deduced from the dwell time percentages in the system queues. (The sum of the dwell times in all system queues is 100%.)

Example 2	96 %
CPU utilization	
TU portion:	36 %
TPR portion:	45 %
SIH portion:	15 %

Main application: TP task category

Dwell time in 01 (%) / Dwell time CPU (%) = 5.4% / 1.2% = 4.5

background application: Batch task category

Dwell time in 01 (%) / Dwell time CPU (%) = 16.8% / 1.4% = 12

The total CPU time required by all tasks is too high. The CPU resource is overloaded.

If the CPU workload is heavy and the performance demands of the individual users are not being met despite efficient utilization and short wait times, the resource requirements per user request (e.g. CPU time required per transaction) are too high (see section "Examining application performance with openSM2").

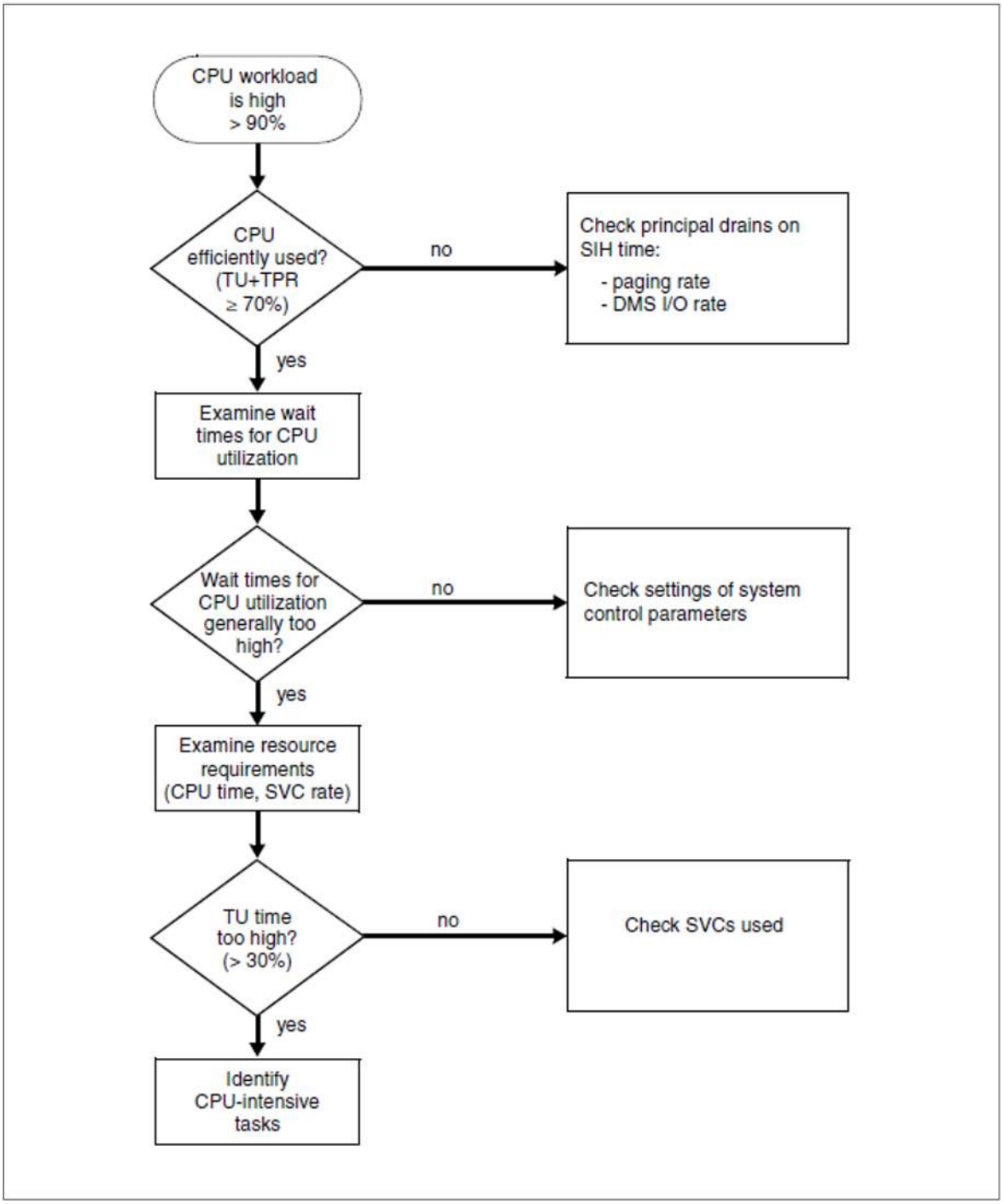


Figure 17: Tuning approaches in the case of high CPU utilization

Low CPU utilization

If performance problems arise without a correspondingly heavy CPU workload (<90%), one of the following causes may be involved:

- Conflict situation on the DMS I/O side
  (see section "I/O load")

  Overloading of volumes containing files which are required equally by a large number of users (e.g. particular user control files or the system file TSOSCAT or SYSEAM) has a particularly unpleasant effect.

  The SM2 monitoring program FILE can be used to calculate the number of I/O operations on these files.
- Paging rate too high (see section "Paging")

  In the case of severe main memory shortage (i.e. when the paging rate far exceeds the standard values given), bottlenecks result on the public volumes with regard to the number of paging I/O operations to be executed per second. This leads to a forced CPU wait state (IDLE).
- Delays caused by PASS/VPASS calls

  If the SVCs PASS/VPASS are called too often, tasks are placed in a wait state too frequently and cannot utilize the CPU.
- Bottlenecks caused by server tasks

  Some applications have large portions of their workload handled by specialized server tasks. If the progress of these tasks is impeded by paging, deadlocks, resource overloads, unfavorable priorities, etc., this can detrimentally affect the entire application. The server task itself acts as a resource.
- Insufficient parallel processing because number of tasks too small

  Not only do the service times of the relevant hardware resources affect performance, but also the amount of time during which a task is occupied with CPU and I/O activity in order to perform a desired user function.

Example

If the average processing time per transaction is 0.5 seconds (0.2 seconds CPU time + 0.3 seconds I/O time), it is possible to obtain the following maximum transaction rate (transactions/s) for a task (not including the wait time for the relevant hardware resources):

    1 / 0.5 sec. per transaction = 2 transactions per second

This yields a CPU workload of 0.2 * 2 = 0.4 or 40%.
A higher transaction rate or more favorable utilization of CPU capacity can only be obtained by raising the number of tasks (for details, see section "Task occupancy time").

Examining application performance with openSM2

It is considerably more difficult to locate the causes of user-oriented performance problems than to investigate system-oriented performance problems (using SM2).

If response or dwell times are unsatisfactory for certain users, it is necessary to interpret the task-specific measurement values as well as to assess the global workload values.

In addition, SM2 user task measurement should be performed:

/START-TASK-MEASUREMENT allows users or systems support to specify tasks which they want to have monitored.
Besides task-specific measured values, program counter statistics and SVC statistics can also be requested. The measured values are written to a user-specific file and analyzed by the software product SM2-PA (see the “SM2-PA” manual [30 (Related publications)]).

Task occupancy time

The task occupancy time consists of the CPU time and the I/O time of a desired user function plus the wait time for the corresponding hardware resources due to their workload.

The longer the task occupancy time, the smaller the number of user requests which this task can execute per unit of time.

Example

Task occupancy time for an average transaction in TP mode:

Ocupancy mode	**Time**
Hardware service time of CPU	200 ms
Wait time for CPU (with 60% CPU workload, assuming “M/M/1”)	300 ms
Hardware service time of all I/O operations	400 ms
Waiting for I/O (with 30% workload, assuming “M/M/1”)	170 ms
Sum	1070 ms

If this application requires a higher transaction rate than 1 / 1.07 sec. per transaction = 0.93 transactions per sec. at peak load time and if only **one** task is available, that task will create a bottleneck.

The resultant wait periods in front of the task (incoming messages accumulate in the BCAM buffer) can be many times longer than the task occupancy time (see also report “Inwait time distribution” of the "Reports of the report group BCAM-CONNECTION").

The SM2 monitoring program TASK provides information on the task occupancy time.

This task occupancy time can be estimated from the duration entries for the various wait states and the CPU time used. Difficulties arise if it is unsure whether certain wait states are voluntary or inadvertent, i.e. whether or not a task requested the service which is provided for it itself.

The following ratio indicates that the task occupancy time is too long (D = Duration):

(CPU time + CPU-WAIT(D) + DISK-IO-WAITS(D) + NON-DISK-IO-WAITS(D)) / dwell time > 0,7

The dwell time **does not** include the wait time for other tasks (it may consequently be necessary to add this).
The computed CPU time refers to the TU and TPR processor states; the times for I/O activity refer solely to software service time.

Task conflicts

The following conflict situations are possible:

- Conflict situations in connection with access to shared public/private volumes
- Conflict situations in connection with updating shared data
- Problems in connection with multi-level task concepts (server tasks or handler tasks)

Conflict situations in connection with access to shared public /private volumes

A SOFTWARE DURATION value (volume occupied) more than 10% higher than the corresponding value for HARDWARE DURATION is a clear indication that other tasks are impeding processing (exception: asynchronous input/output, see the explanation of the ""Time" report of the DISK report group").

In this case, a task must wait for the volume before initiating the I/O operation, because the it is busy. The SM2 monitoring program TASK (with INFO=HIGH) indicates which other tasks are likewise occupying this volume with their I/O operations.

The problem can be solved by using the PAV function or file relocation.

Conflict situations in connection with updating shared data

Conflict situations can arise during the updating of shared data (e.g. tables) whose contents must be kept consistent for purposes of job processing

Conflicts of this sort can be regarded as forced serialization. They can be identified in the task-specific characteristic data by a relatively high proportion of

- “active waits” when serializing via the bourse mechanism or issuing VPASS 0 wait calls
- “inactive waits” when the PASS wait call is used.

There are, however, many instances in which SM2 can make no distinction whatsoever between such a state and a normal condition. User systems (e.g. UTM, UDS, SESAM/SQL, ORACLE) now offer their own monitors for use in pinpointing such conflicts.

Problems in connection with multi-level task concepts (server tasks or handler tasks)

Functions shared by all tasks are very frequently isolated and concentrated into a **single** task (e.g. I/O operations for special data sets, database calls, etc.).

If the task occupancy time for this central task is too long, wait states will result for the tasks which draw upon it.

A bottleneck ahead of a UTM application as a result of too small a number of (UTM) server tasks can be measured directly (see the explanations in "Threshold value monitoring of the number of UTM tasks using openSM2" (Optimizing the various phases)).

Often wait states can be detected by the increasing number or duration of “active waits” computed by the SM2 monitoring program TASK for the calling tasks. In the case of TP tasks, the number of “active waits” also contains not only those “waiting for input/output” but also those “waiting for terminal input”. This portion is negligible when the central task has a long task occupancy time.

The server task must be programmed so as not to require any locks, which it might then have to wait out. If the server task is not given priority over other tasks, it likewise runs the risk of encountering sudden losses in throughput. Due to the way jobs are handled between the job source task and the server task, congestion may occur in the event of programming which is not error-free, causing the server task to come to a halt. All of these problems are indications that the especially high quality of the coding required for server tasks is not being attained.

Example (simplified)

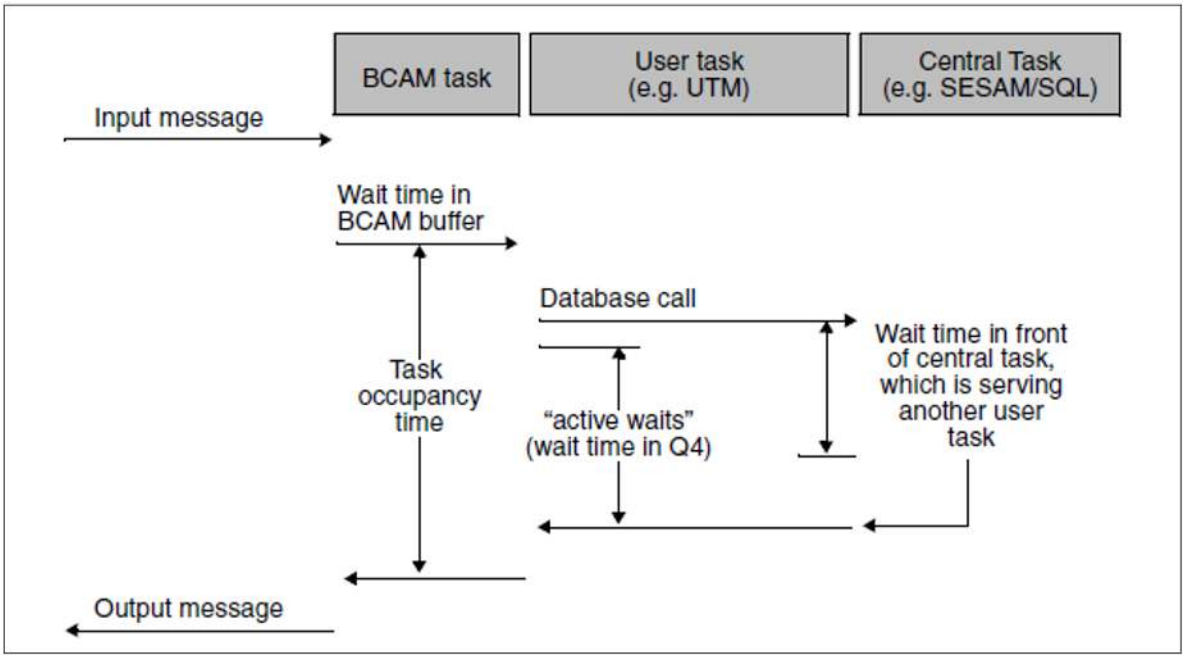


Figure 18: Time behavior of multi-level task concepts

Procedure in the case of high resource requirements

If the performance expectations of the user are not being satisfied despite the fact that resources are being utilized efficiently and have short wait times, this indicates that the resource requirements per user request are too high (e. g. CPU time requirement and/or number of I/O operations per transaction).

Indications regarding resource requirements can be found in the figures supplied by the SM2 routine TASK on

- CPU time consumption
- number of SVC calls in TU
- number of DMS I/O operations

For a more detailed investigation of application program behavior, SM2 user task measurement with an SM2-PA analysis is required:

- Program counter statistics enable those program areas to be pinpointed which are frequently executed.

The SVC statistics contain all the SVC numbers called and their call addresses.

Influence of the network

Previous recommendations have all concentrated on improving the efficiency of the CPU. As described in section "Formulating a performance expectation", good transaction times are essential to productivity. Measures for optimizing response times in transaction mode and with BCAM are described in section "Optimizing an OLTP application".
As already shown in figure 1 (Characteristic performance values of an online application), network runtimes are included as part of the transaction times. These runtimes may be greater than the actual response time in the host. For this reason it must be ascertained whether or not unsatisfactory transaction times are attributable to the network.
If the reports of the report group BCAM-CONNECTION indicate that response times are acceptable yet predominantly long transaction times are observed at the terminals, this means there is a bottleneck in the network.

A network is a complex entity with many interdependencies. In view of the great variety of possible network structures, only the most important characteristics and general conditions are indicated here. Describing how to deal with such problems comprehensively is beyond the scope of this manual.

Load requirements

Load type	Requirements
TP or interactive mode	Short response time
Data backup, file transfer	Throughput
Internet, email	Throughput, response time

Possible bottlenecks

- Performance of network connection (e.g. older-type HNC)
- Client system (e.g. hardware performance, structure of the application, networking parameters)
- Network configuration with conflicting goals with regard to response time / throughput (e.g. parallel operation of transaction-oriented and throughput-oriented applications)
- Network structure (e.g. configuration of network nodes, routers, switches, hubs)
- Network overload by third party (e.g. the Internet)

Related publications

You will find the manuals on the internet at .

[1] **ARCHIVE**

User Guide

[2] **BCAM**

User Guide

[3] BS2000 OS DX

Files and Volumes Larger than 32 GB

User Guide

[4] **Fujitsu Server SE Series**

Administration and Operation

User Guide

[5] **DAB** (BS2000)

Disk Access Buffer

User Guide

[6] BS2000 OS DX

Utility Routines

User Guide

[7] **DRV** (BS2000)

Dual Recording by Volume

User Guide

[8] BS2000 OS DX

DMS Macros

User Guide

[9] BS2000 OS DX

Introduction to DVS

User Guide

[10] BS2000 OS DX

System Administration

User Guide

[11] **FDDRL** (BS2000)

User Guide

[12] **HIPLEX MSCF** (BS2000)

BS2000-Processor Networks

User Guide

[13] **HNC**

High-Speed Net Connect

User Guide

[14] **HSMS** (BS2000)

Hierarchical Storage Management System

User Guide

[15] BS2000 OS DX

Commands

User Guide

[16] **LEASY** (BS2000)

Program Interface and Strategies

User Guide

[17] BS2000 OS DX

Manual System Exits

User Guide

[18] **openSM2** (BS2000)

Software Monitor

User Guide

[19] **openUTM**

Concepts and Functions

User Guide

[20] **openUTM**

Administering Applications

User Guide

[21] **openUTM**

Generating Applications

User Guide

[22] **openUTM** (BS2000)

Using openUTM Applications under BS2000

User Guide

[23] **PCS** (BS2000)

Performance Control Subsystem

User Guide

[24] **ROBAR** (BS2000)

Controlling MTC Archiving Systems

User Guide

[25] **SDF** (BS2000)

SDF Management

User Guide

[26] **SESAM/SQL Server** (BS2000)

Database Operation

User Guide

[27] **SESAM/SQL Server** (BS2000)

Performance

User Guide

[29] **SHC-OSD / SCCA-BS2**

Storage Management for BS2000

User Guide

[30] **SM2-PA** (BS2000)

SM2 Program Analyzer

User Guide

[31] **SPACEOPT** (BS2000)

Disk Optimization and Reorganization

User Guide

[32] BS2000 OS DX

System Installation

User Guide

[33] BS2000 OS DX

System-managed storage

User Guide

[34] **VM2000** (BS2000)

Virtual Machine System

User Guide