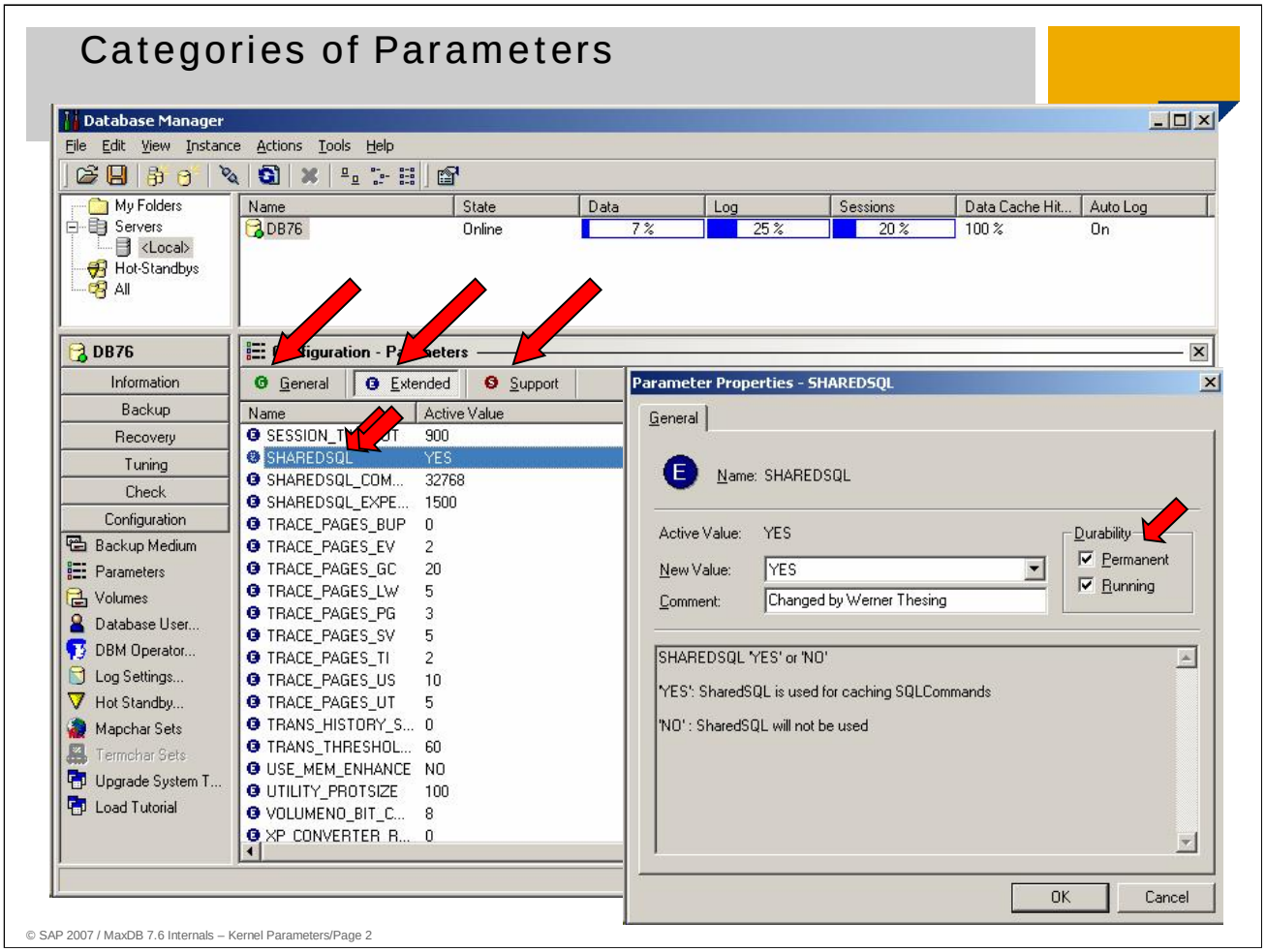
MaxDB

Kernel Parameters
Release 7.6

Heike Gursch
Werner Thesing



Kernel parameters are divided into three classes:

- **General**

  These parameters are set by database administrators.

- **Extended**

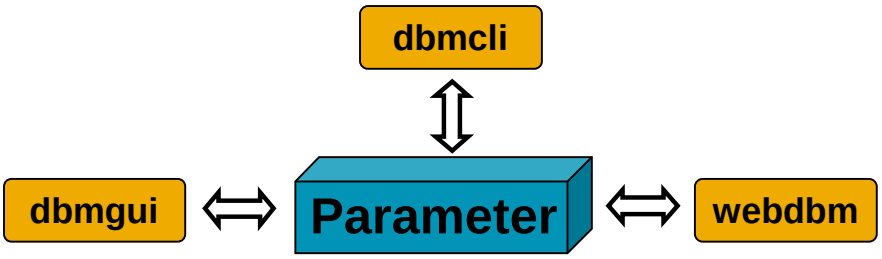
  These parameters are set in consultation with MaxDB Support or by implementing notes from the database administrator.

- **Support**

  These parameters are set by MaxDB Support or the developers.

Before a MaxDB version comes out, it is programmed to calculate the optimal values for the respective operating system platform.

How do I change parameters?



Example:

- Call: dbmcli –d MYDB –u control,pass
- Display all: param_directgetall
- Display: param_directget CACHE_SIZE
- Assign value: param_directput CACHE_SIZE 100000
- Calculate: param_checkall

The parameter file is stored in the file system in binary format.

The Database Manager's client tools enable you to read and change parameters.

With dbmcli , parameters can be changed directly or in a parameter session.

Above you see a few examples of commands for making changes directly.

If parameters are changed within a session, a Commit makes all the changes valid while an Abort makes them all invalid.

Use the commands:

- param_startsession — starts a parameter session
- param_commitsession — ends the parameter session and saves the values
- param_abortsession — ends the parameter session without saving it
- param_getvalue — displays a parameter value
- param_put — changes the value of a parameter
- param_restore — activates old parameter version

The dbmcli command “help param” displays more commands.

When do parameters become active?

Many parameters can be changed online:

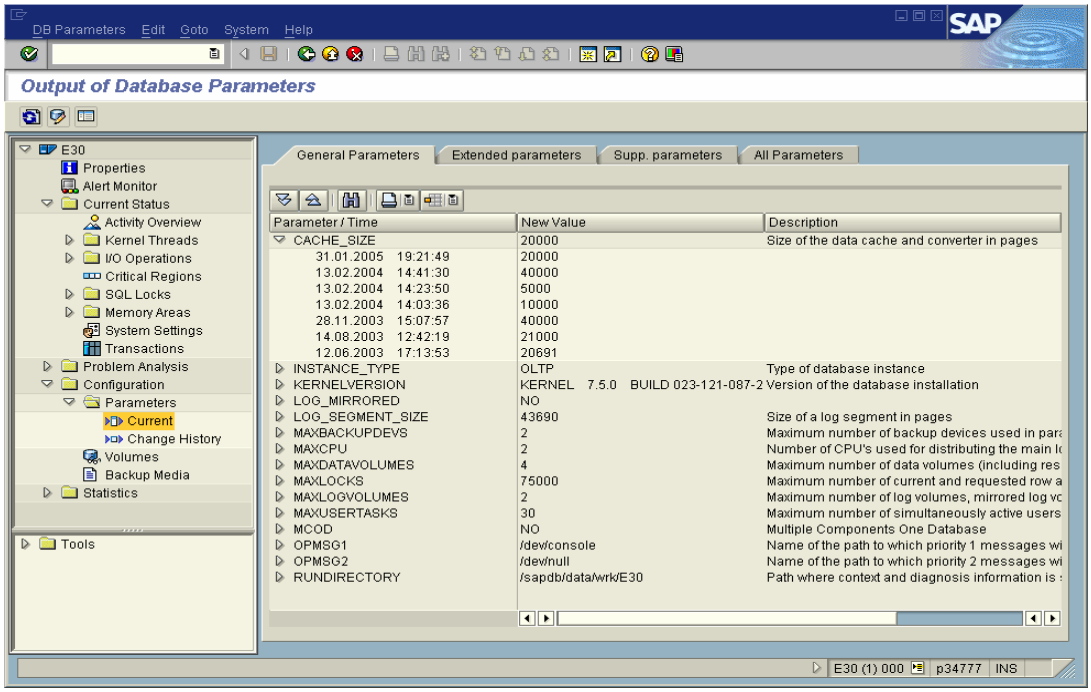
- dbmcli –d MYDB –u control,pass
  - param_getfull SHAREDSQL
    ...
    CHANGE RUNNING
    ...
  - param_put –running –permanent SHAREDSQL YES
  - Shared SQL is only active for new user logons

As of Version 7.6, MaxDB offers the possibility of changing parameter values in online operation.

In general, you can change status or counter values online. Some newly introduced parameters can be changed online although they influence the process or memory structure of the database. In the case of SHAREDSQL, changes can only be made with new database logons.

MaxDB development is pursuing the goal of making it possible to change all parameter values online.

Current Parameter Values



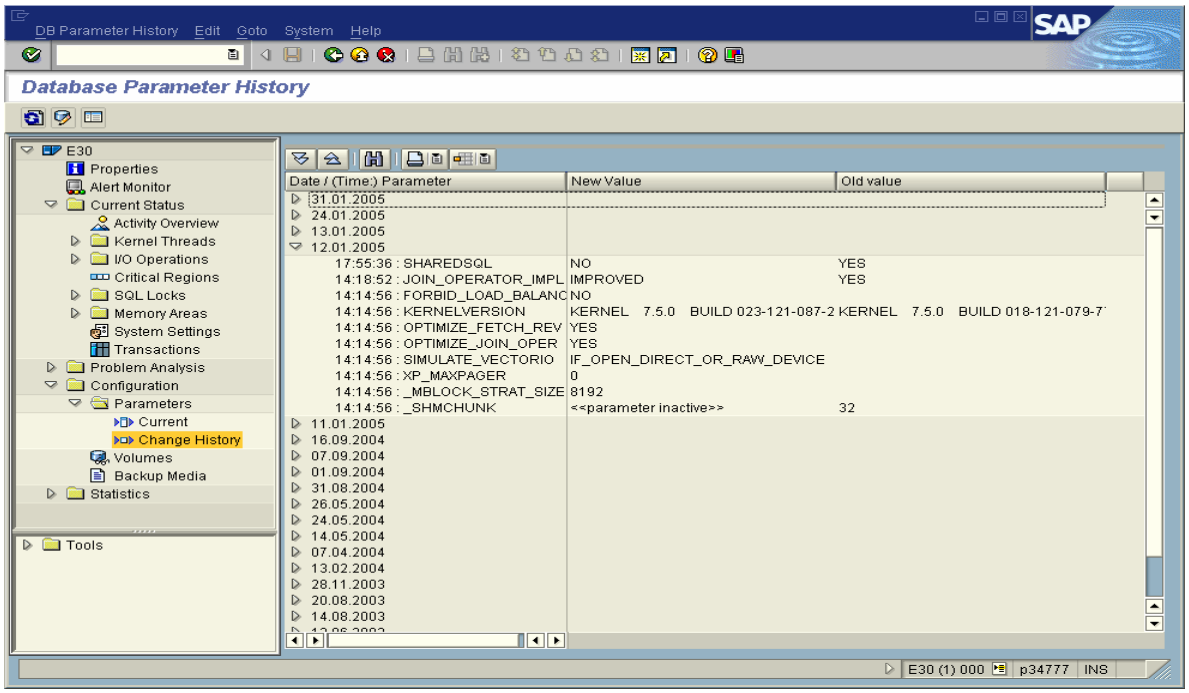
The parameter file is in the directory <IndepDataPath>/config.

The name is identical to the name of the instance. If changes occur, the old file is copied to <instance>.<no> .

You use the dbcmli command param_gethistory to display the parameter history. In the SAP system, you can display the parameters in transaction DB50 (or ST04) .

In data backups (complete or incremental), the contents of the parameter file are written to the backup medium.

DB50: Configuration and History of Parameters



In transaction DB50 you can view the list of all parameter changes, sorted according to the change date.

The system displays a list of the database parameters changed at this point in time, and their previous and new values.

Parameters that are no longer used by the database as of a particular date are assigned `<<parameter inactive>>` as a new value.

The parameter history data is logged once a day by a collector. If you have changed database parameters, these changes are only displayed in this output after the collector has run. A current display of the parameter history is offered by the DBMGUI.

Dependencies

File: <instroot>/env/cserv.pcf

```
...
ID MAXUSERTASKS
  TYPE int
  DEFAULT 50
  MANDATORY YES
  GROUP GENERAL
  CODE
    CONSTRAINT 1 MAXUSERTASKS <= \
            32000 MAXSERVERTASKS - MAXUSERTASKS >= \
            AND
  ENDCODE
EXPLAIN
 Maximum number of simultaneously active users (database sessions).
 Overconfiguration exceeding the actual requirement results in an
 excessive demand for address space (especially shared memory).

 (2 bytes integer)
ENDEXPLAIN
HELP
Maximum number of simultaneously active users (database sessions)
ENDHELP
...
```

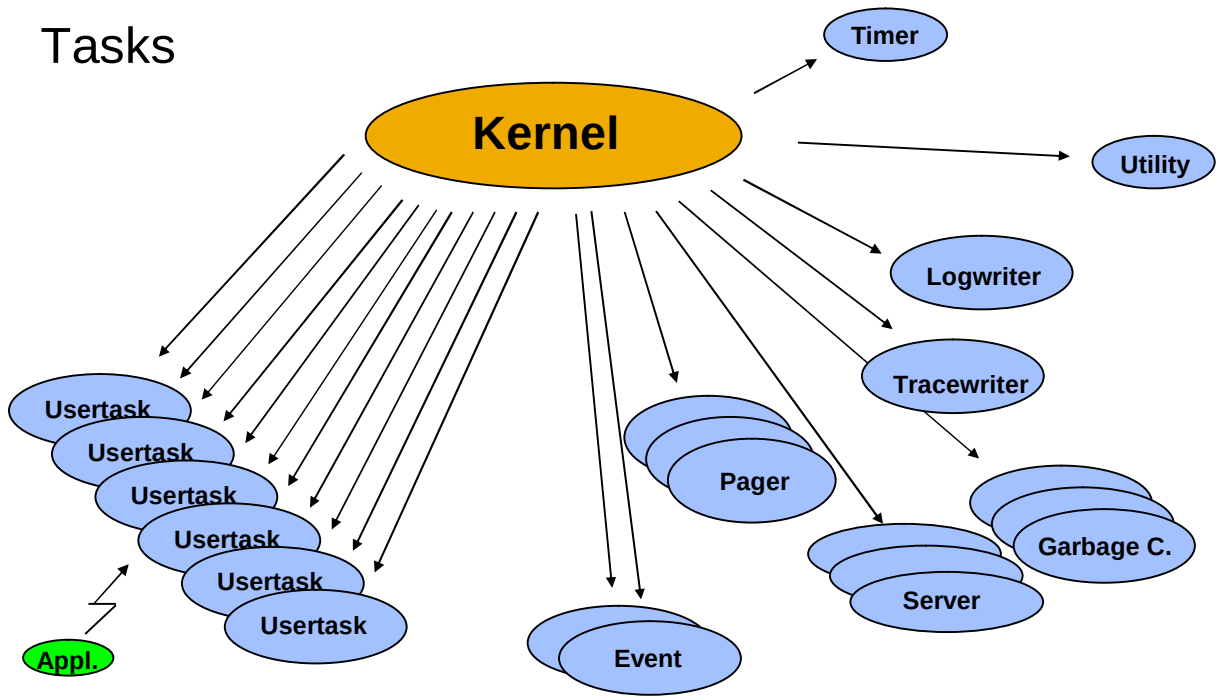

Calculation formulas, short texts and help texts for the parameters are found in the file <instroot>/env/cserv.pcf.

The following properties can be assigned to parameters:

Property	Description	Values
CHANGE	Parameter can be changed	YES | RUNNING | NO
INTERN	Value is in the parameter file	YES | NO
MANDATORY	Parameter must have a value	YES | NO
CLEAR	DB copy: Parameter is not copied	YES | NO
DYNAMIC	Automatic numbering (e.g. DATAVOL_?)	YES | NO
CASESENSITIVE	upper/lower case (contents)	YES | NO
OVERRIDE	Parameter value may be changed	YES | NO | HIGHER
DEVSPACE	Volume Parameter	YES | NO
MODIFY	Parameter may be changed after generation of instance	YES | NO GROUP
Classification	General, Extended, ...	
DISPLAYNAME	Parameter name displayed in DBMGUI	<value>
VALUESET	Permitted values	<values>
MAX	Maximum parameter value (numeric)	<value>
MIN	Minimum parameter value (numeric)	<value>

Do not change the file cserv.pcf under any circumstances unless instructed to do so by MaxDB Support or MaxDB Development.

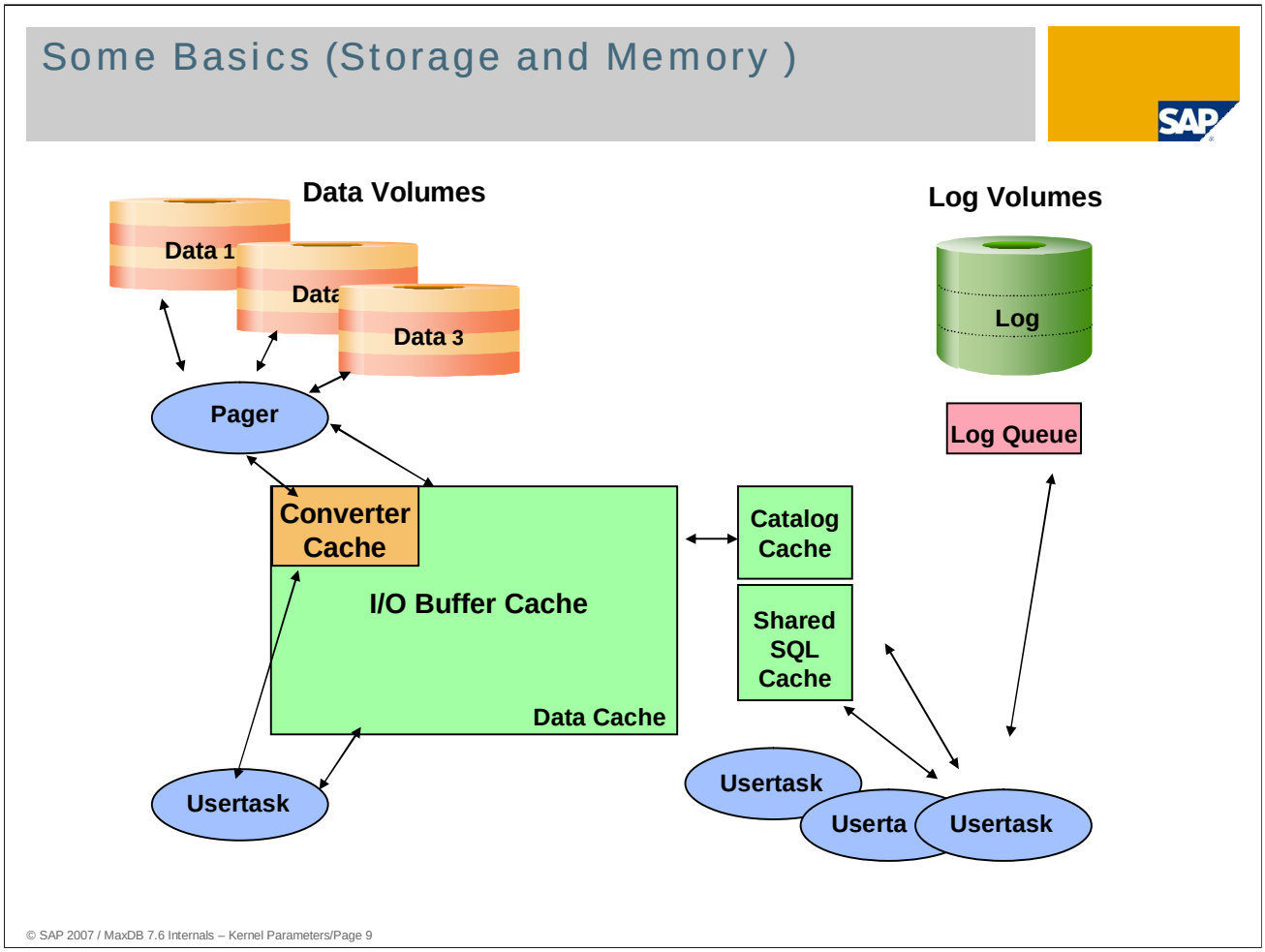
Some Basics (Tasks)



Tasks process self-contained areas of activity in the kernel.

One or more tasks are combined into a thread. This distribution is set in the internal parameter _TASKCLUSTER, which is calculated in relation to the open parameter MAXCPU. Threads that contain user tasks are called UKT(s).

- **User task:**
  A user task for processing SQL statements is created for each session of a database user.
- **Log writer:**
  Does the writing in the log area.
- **Trace writers:** Write the kernel trace (Vtrace)
- **Server:**
  For data backups, CREATE INDEX, ...
- **Pagers:**
  For I/O between DATA CACHE and data volumes.
- **Garbage collectors:**
  Delete the undo log entries of closed transactions
- **Timer, Utility, Event:**
  Not performance relevant.



Reading from the disks in the **data cache** is done by various tasks. Writing from the cache to the disks should generally be done by a data writer. A user task only writes to the disk if it finds no free data pages in the cache and has to displace pages. The standard settings for some parameters can prevent this.

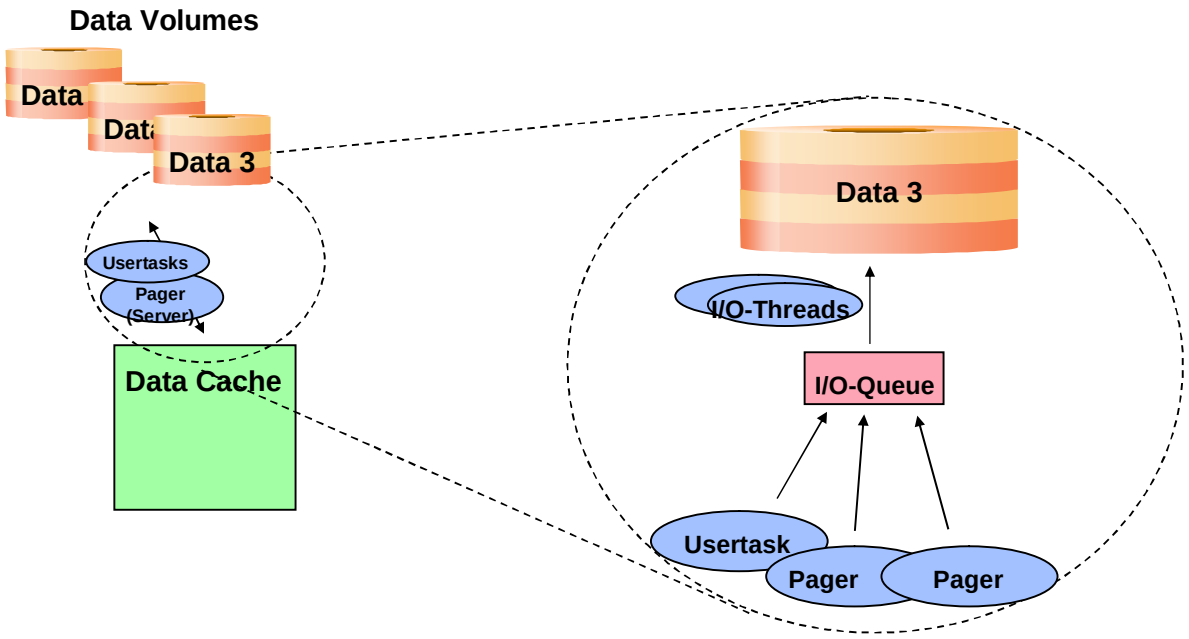
The converter is the conversion table between the logical page numbers and the physical addresses of the blocks on the disks (shadow memory administration). It is stored distributed across the data volumes. Because the converter has to be accessed for each read or write operation, the **converter cache** is of central importance.

The **catalog cache** contains catalog information as well as execution plans for SQL statements.

The **shared SQL Cache** contains known SQL statements and their execution plans with more information.

Log entries of user tasks are written to the log volumes via the **log queue**. With a Commit or when a page of the log queue is full, the page is written by the log writer. A Commit is confirmed when the log entry for the Commit is in the log volume.

Some Basics (Asynchronous I/O)

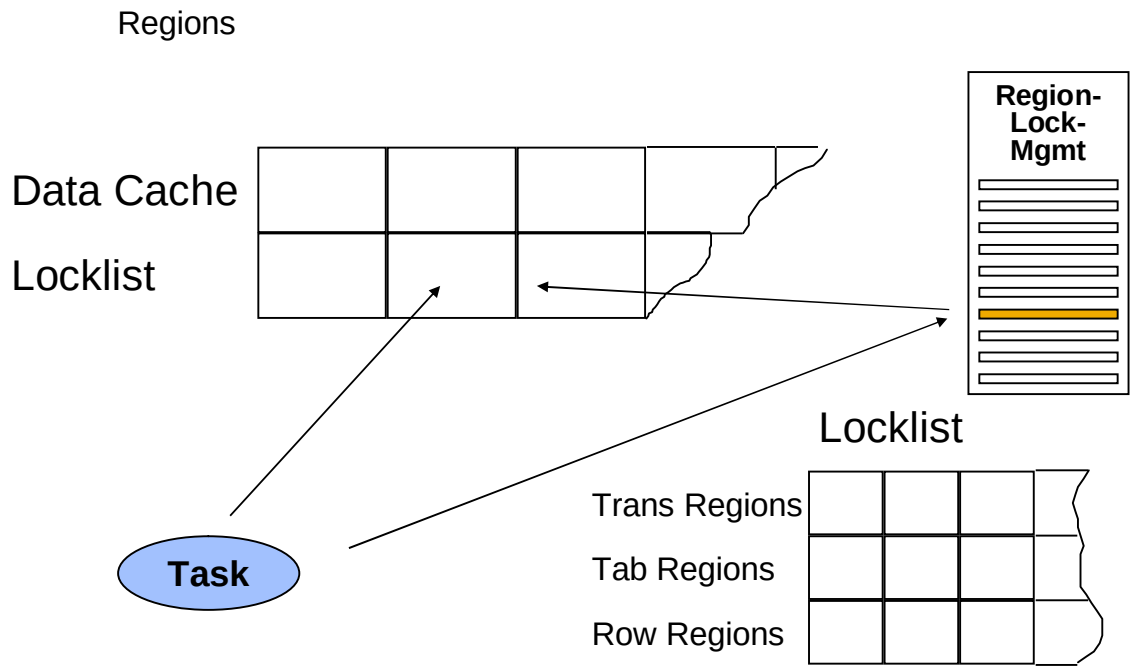


In general, tasks do not write directly from the data cache to the disks but rather into an I/O queue. That way the actual I/O can be processed by special threads asynchronously and in sufficiently large portions.

If the affected thread is currently unable to perform another task, a user task writes directly without going through the I/O queue in order to avoid a thread change.

On Windows, the asynchronous I/O calls of the system are used.

Some Basics (Synchronisation)



In multitasking mode, the different memory structures of the database kernel - data cache, lock list or transaction administration, etc. - must be protected against simultaneous access by different tasks. To enable a higher degree of parallelism, these are divided into independent, mutually-accessible units called **regions**.

To protect region accesses from other tasks, there is a special lock administration; this must not be confused with the administration of SQL locks on database objects such as rows or tables.

To determine appropriate tuning measures when performance problems occur, it can be important to know whether bottlenecks are the result of tasks waiting for SQL locks or region accesses; in the latter case, also to which regions.

Collisions of accesses to regions can be ascertained with the following command:

- x_cons <SERVERDB> show regions
- dbmcli –d <dbname> -u <dbmuser,passwd> -n <server> show regions

General Information on Parameters

Parameters are available for

- Constants of system configuration
- Caches and various memory structures
- Communication, I/O
- Process structure, CPU-Usage
- Optimizer
- ...

General Information on Parameters

Constants of system configuration

INSTANCE_TYPE	OLTP, BW, CS, LVC, ...
DATA_VOLUME_NAME_<nnn>	Names of data volumes
LOG_VOLUME_NAME_<nnn>	Names of log volumes
LOG_MIRRORED	Log volume mirroring
MAXDATAVOLUMES	Max. number of data volumes
MAXBACKUPDEVS	Max. number of parallel backup devices
MAXUSERTASKS	Max. number of parallel user sessions
RUNDIRECTORY	Default: /sapdb/data/wrk/<DBNAME>
KERNELDIAGSIZE	Size of file knldiag (in Rundirectory)
TRACE_PAGES_..	Size of trace buffer for dedicated tasks (default im Rundirectory)
...	

These parameters are, in part, performance relevant, but they are not tuning parameters, strictly speaking. They are not discussed further here.

Caches and various memory structures (1)

CACHE_SIZE	Size of I/O buffer cache in 8 KB pages
XP_DATA_CACHE_RGNS	Segmentation of data cache
_LRU_FOR_SCAN	Scan optimization
_IDXFILE_LIST_SIZE	Parallel create index
XP_CONVERTER_REGIONS	Segmentation of converter cache
CAT_CACHE_SUPPLY	Total size of catalog cache
SHAREDSQL	Shared-SQL cache
SHAREDSQL_COMMANDCACHESIZE	Size of Shared-SQL cache
SEQUENCE_CACHE	Size of caches used for sequences

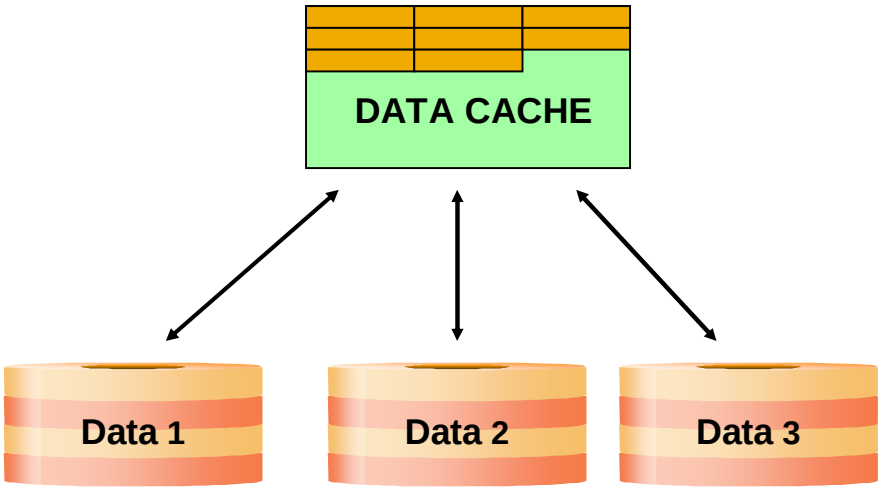
Caches and Other Memory Structures (2)

LOG_IO_QUEUE	Size of LOG_IO_QUEUE in 8 KB pages
LOG_QUEUE_COUNT	Number of log queues
LOG_IO_BLOCK_COUNT	Max. no. of log pages per I/O
LOCAL_REDO_LOG_BUFFER_SIZE	Transaction buffer for redo log entries
MAXLOCKS	Maximum number of parallel line locks
_LOCK_SUPPLY_BLOCK	Number of entries that are taken from free space management
DEADLOCK_DETECTION	Deadlock detection
USE_MEM_ENHANCE	AWE support under Windows
MEM_ENHANCE_LIMIT	Maximum size of used AWE memory

I/O Buffer Cache

CACHE_SIZE

- Size of I/O buffer cache in 8 KB pages.



The size of the I/O buffer cache depends on the size of the database, the type of use, the size of the main memory and the number of users working simultaneously.

The I/O buffer cache is divided into the converter cache and the data cache. The size of the converter cache grows and shrinks with the number of used data pages. The system view CONFIGURATION displays the size of the different areas:
SQL Statement: SELECT * FROM CONFIGURATION
WHERE DESCRIPTION IN ('Converter size', 'Datacache size')

The converter cache hit rate is always 100%. The data cache hit rate should be over 98%.

Values: Default: 5,000 pages
Min: 800, Max: 2,147,483,640
SAP system: > 30.000

The data cache hit rate can be determined as follows:

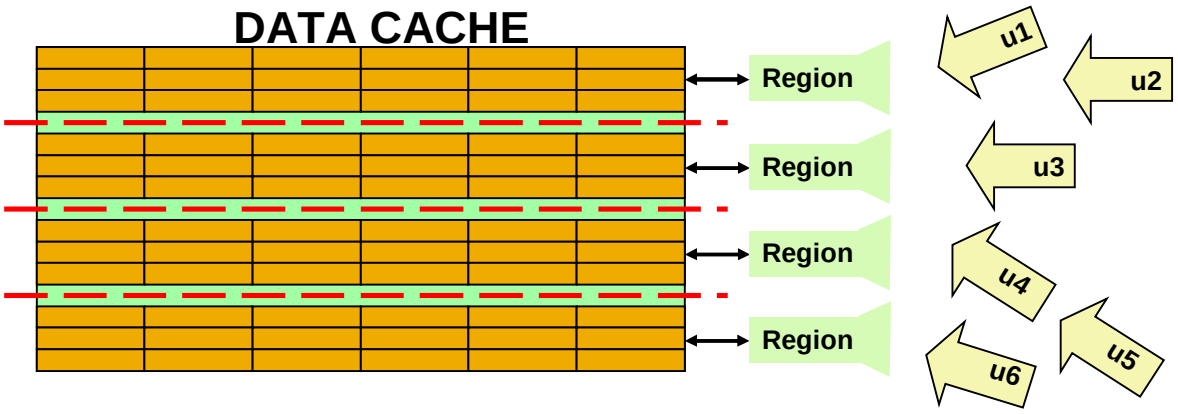
- SQL statements
  - SELECT * FROM MONITOR_CACHES
  - MONITOR INIT executes a reset
- DBMGUI (main screen)
- DBAnalyzer
  - Displays the hit rate for a defined period of time.

System swapping occurs if the data cache is too large. System swapping is generally more costly than removing data pages from the cache.

Data Cache Regions

XP_DATA_CACHE_RGNS

- Number of autonomous entities, the data cache will be divided in



XP_DATA_CACHE_RGNS increases the amount of work done in parallel on the data cache. The data from the data cache are striped across the data cache regions. Each region has its own semaphore. Changing the parameter XP_DATA_CACHE_RGNS causes the internal parameter _DATA_CACHE_RGNS to be calculated and set.

Each data page in the data cache is assigned to a region. Multiple users can change pages in different regions at the same time.

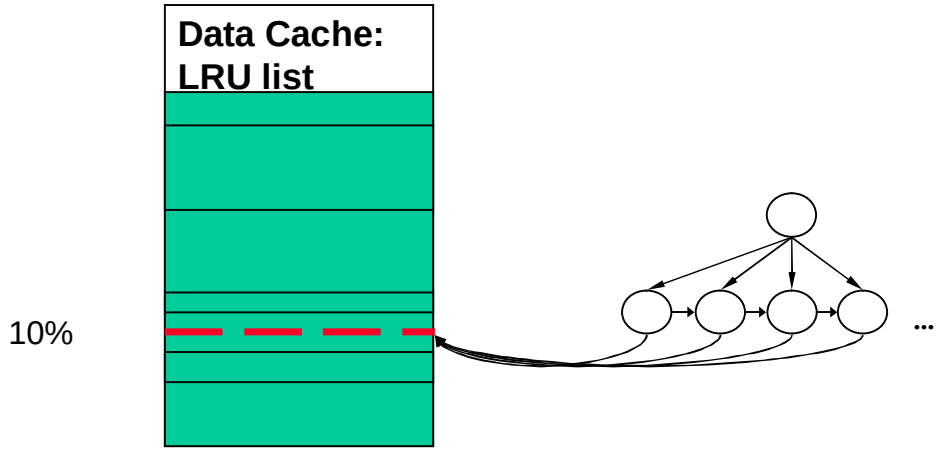
Values: Default: 8
Min: 1, Max: 64
The number of regions is calculated depending on the size of the parameter CACHE_SIZE.

If collisions occur during access to the data pages of a region, this bottleneck can be remedied by increasing the number of regions.

Scan Optimization

LRU_FOR_SCAN

- This parameter specifies, if scans should be handled privileged in data cache.
- Within OLTP it makes less sense, to use the complete data cache for table scans. In most cases the data will not be reused shortly after the scan.



It is presumed that the lion's share of the data read to the data cache in a table scan cannot be used again. The initial value disburdens the cache of such data and thus favors other commands. But this can lead to long delays for the user who is performing the table scan as he may frequently have to displace pages from the cache.

The 'YES' setting can be advantageous if, for example, multiple smaller tables in their entirety are to be held in the data cache.

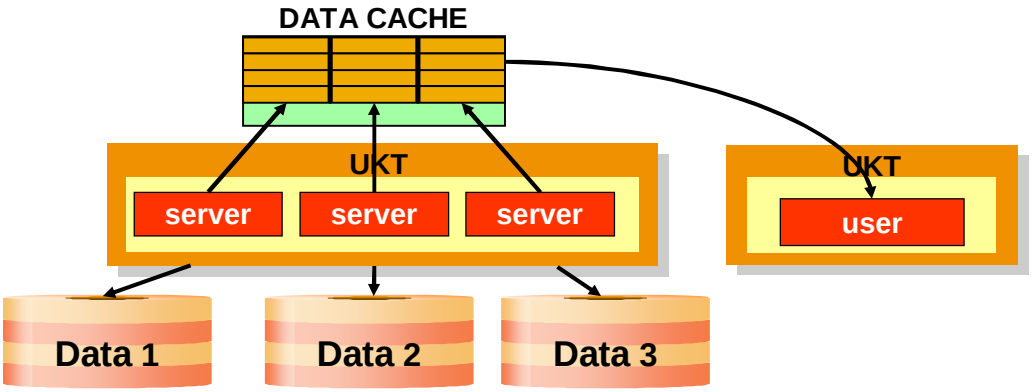
Values: Default: NO
YES: The whole data cache is used for scans.
NO: Only the last part of the data cache is used for scans.

The LRU list (Least Recently Used) is a concatenation of data pages. The pages used most recently are generally at the front.

Parallel Create Index

_IDXFILE_LIST_SIZE

- Maximum number of auxiliary B*-trees to support parallel Create Index
- A high number of B*-trees minimizes the number of merging steps.
- The B* index pages of the auxiliary B*trees have to fit into the data cache – otherwise performance will decrease.



With CREATE INDEX, server tasks read the data pages of the table from the volumes in parallel. The data required for the index are collected in temporary B* trees. The B* trees are then combined into a sorted index in a series of merge steps.

The speed of CREATE INDEX increases when as many temporary B* trees as possible are being used. This is only the case, however, if the B* trees can be held in the data cache.

If the index pages of the B* trees cannot be held in the cache, the performance of CREATE INDEX can worsen significantly.

The value for _IDXFILE_LIST_SIZE should not exceed CACHE_SIZE / 3.

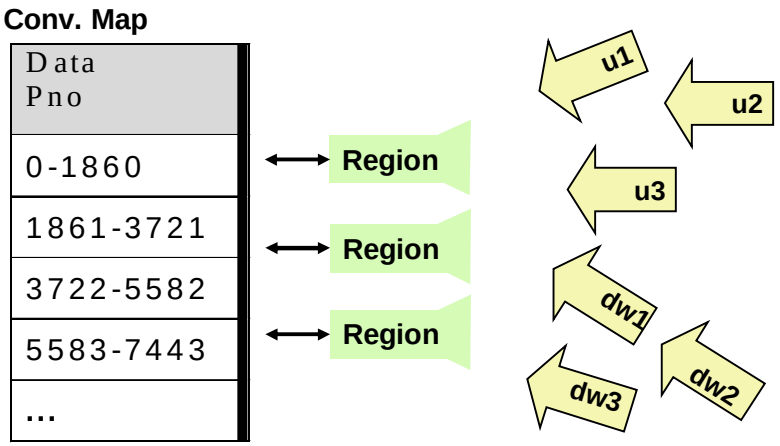
Values: Default: 2048

- Allowed: 0, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072

Converter Cache Regions

XP_CONVERTER_REGIONS

- Number of autonomous entities, the converter cache will be divided in



The parameter XP_CONVERTER_REGIONS is used for setting the number of regions through which accesses to converter pages are synchronized. Collisions can occur in online operation if the parameter is too small and many users request new page numbers at the same time.

The converter map is used to access a converter page. It contains an entry for each converter page. Each entry is assigned to a region. Multiple users can change pages in different regions at the same time.

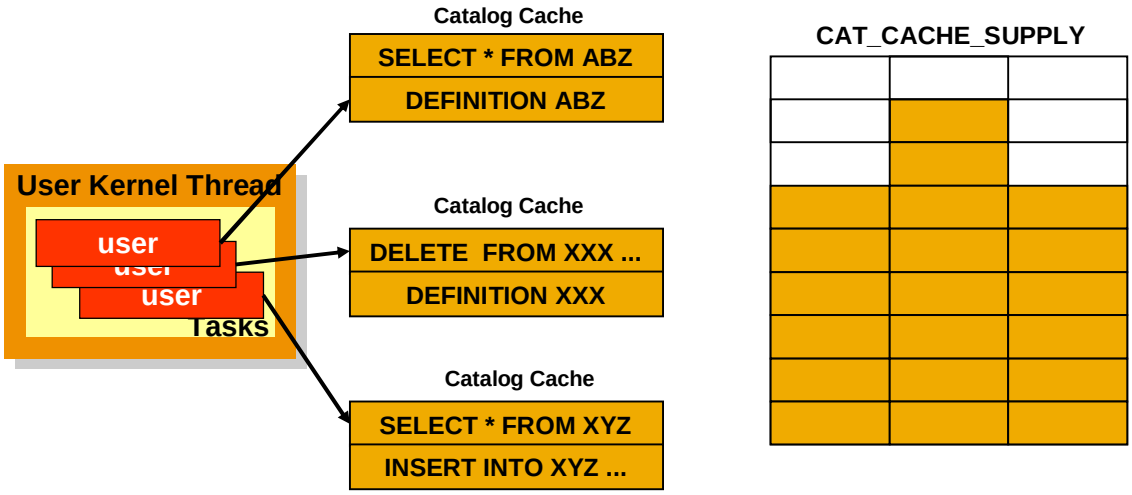
Values: Default: 8
Min: 1, Max: 64
No online change

If collisions occur during access to the data pages of a region, this bottleneck can be remedied by increasing the number of regions.

Catalog Cache

CAT_CACHE_SUPPLY

- Size of memory supply for catalog cache allocation in 8 KB pages. Each user task allocates its catalog cache from this pool.



The catalog cache holds user-specific data from the database catalog. At the beginning of the session, each user task receives a catalog cache; the size of this cache can be set with the parameter CAT_CACHE_MINISIZE (no separate slide). If necessary, the size of the catalog cache is increased during the session until the value CAT_CACHE_SUPPLY is reached for the system as a whole.

Memory for the catalog cache of a user task is only requested from the operating system when required and is released when the session ends.

The memory used by a particular user can be queried:

- SELECT * FROM CONNECTEDUSERS

The lower limit for the hit rate depends largely on the application. The parameter value is too small if the total allocated memory for the individual users reaches the parameter value.

The data cache hit rate should be over 85%.

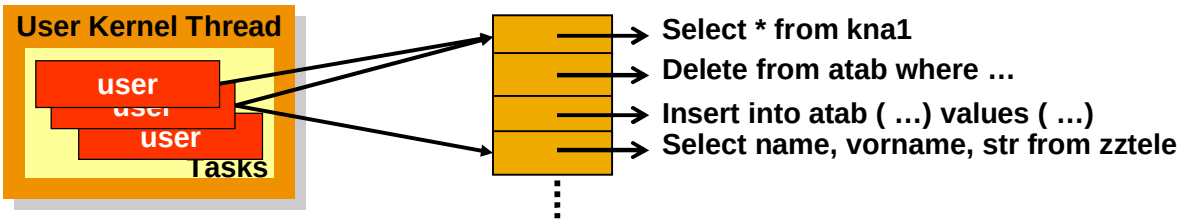
Values: Default: 32

- If (Maxusertasks + 1) / _CAT_CACHE_MINSIZE * 8192 > CAT_CACHE_SUPPLY
  CAT_CACHE_SUPPLY = (Maxusertasks + 1) / _CAT_CACHE_MINSIZE * 8192
- Restriction:
  _CAT_CACHE_MINSIZE * 8192 / MAXUSERTASKS + 1 <= CAT_CACHE_SUPPLY
- No online change

Shared-SQL Cache

SHAREDSQL

- The Shared-SQL cache stores prepared DML statements globally. This information can be reused by different users for a faster execution.
- Improved Monitoring.
- Values:
  - YES: Shared SQL is used
  - NO: Shared SQL is not used (default)



The catalog cache stores SQL statements on a by-user basis. That makes synchronization more efficient because parallel accesses within session contexts are not possible. If the same statements are executed in different user sessions, however, they are stored multiple times in the main memory. For that reason, it is only the prepared information and not the texts of the statements that is stored. This compromises monitoring.

In the shared SQL cache, the statements and their texts are stored globally for all user sessions. Additionally, various counters - for example the number of executions and the number of current executions - are stored.

You can get the statements and the corresponding counters:

- SELECT * FROM CONNECTEDUSERS

With the following statement, MaxDB displays the current SQL statements:

- SELECT * FROM COMMANDSTATISTICS WHERE CURRENTEXECUTECOUNT <> 0

In online operation, shared SQL can be switched on and off dynamically. Changes are active for all new user sessions.

- Switching on and off: DIAGNOSE SHARE PARSE ON/OFF
- Maintain monitor counter: DIAGNOSE ANALYZE COUNT ON/OFF

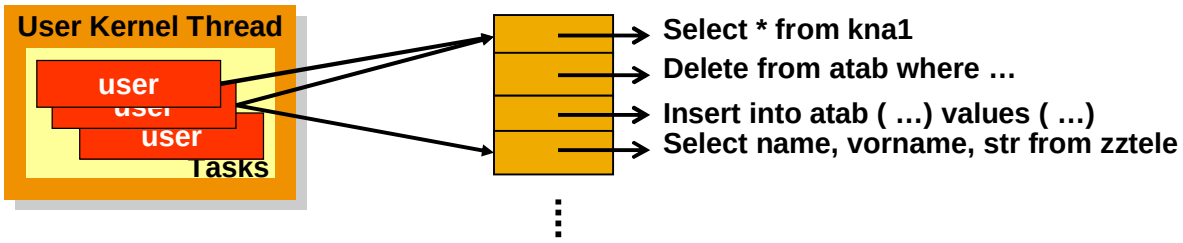
Some counters are not maintained in the standard setting.

Online change: Yes

Size of Shared-SQL Cache

SHAREDSQL_COMMANDCACHESIZE

- The parameter specifies the maximum size of the Shared-SQL Cache. Initially the cache is created with 16MB.
- If the maximum size is reached, older statements with no recent use are removed using an LRU mechanism. If there is still not enough space for a new statement, the cache management enlarges the cache stepwise by 2MB.
- Values (in KB):
  - Default: 262144 (256 MB)
  - Min: 16384 (16 MB), Max: 8388608 (8 GB)



The view COMMANDCACHESTATISTICS displays monitor data of the cache management for shared SQL:

- SELECT * FROM CONNECTEDUSERS

The view CACHESTATISTICS displays the hit rate in the shared SQL cache.

- SELECT * FROM CACHESTATISTICS WHERE NAME LIKE 'COMMAND%',

Values (in KB):

- Default: 32768 (32 MB)
- Min: 2048 (2 MB)
- Max: 8388608 (8 GB)
- Online change: No

When the cache no longer has enough space for a new statement, an LRU mechanism looks for the 20% of statements which have not been used for the longest time.
The mechanism then deletes the statements in this selection for which no parse ID still exists. Generally these are statements for which no cursor exists in the application any longer. For statements for which applications still have a parse ID, the mechanism only deletes the parse information. It stores the statement in a temporary file within the database. If the parse ID is used again by the application, the database kernel copies the statement back to the shared SQL cache and recreates the parse information. If the LRU mechanism cannot provide any free space in the cache, then the cache management enlarges the memory area by 2 MB.

When the maximum value for the size of the Shared SQL cache has been reached, statements can still be formatted in the catalog cache. This does lead to higher execution costs, however.

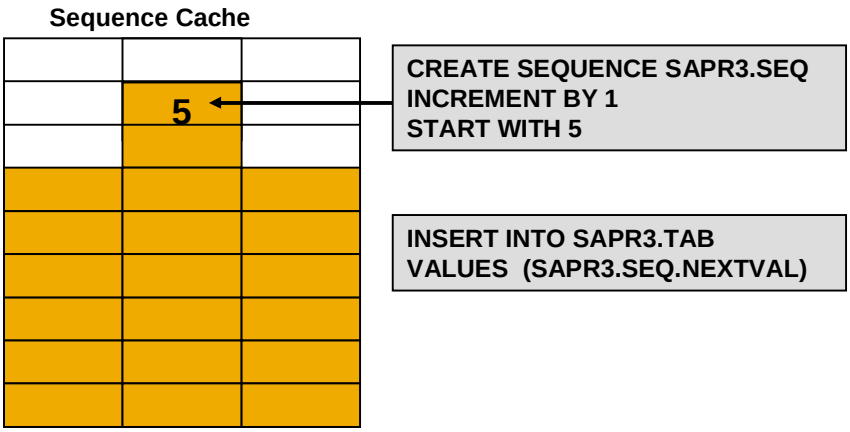
The additional parameter SHAREDSQL_EXPECTEDSTATEMENTCOUNT specifies the expected number of statements in the shared SQL cache. It is used to set the number of hash lists used to search for SQL statements.

For large values for AVGHASHTABLECHAINENTRYCOUNT or MAXHASHTABLECHAINENTRYCOUNT in the view COMMANDCACHESTATISTICS, it may be advisable to increase SHAREDSQL_EXPECTEDSTATEMENTCOUNT.

Sequence Cache

SEQUENCE_CACHE

- Size of memory area which is used for objects of type sequence



Sequences are database objects whose value automatically increases or decreases with each access (NEXTVAL).

Value and management information about the sequences is kept in the sequence cache for quick access.

Values: Default: 1
Min: 1, Max: 2147483640
SAP-System: 1

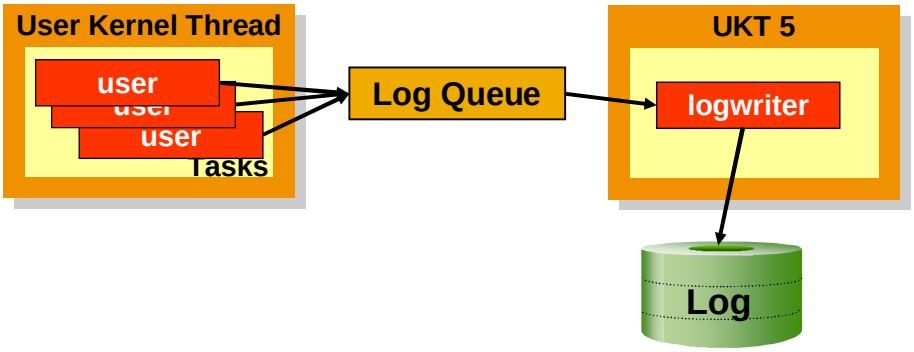
SQL statements

- SELECT * FROM MONITOR_CACHES
- MONITOR INIT executes a reset

Log Queue

LOG_IO_QUEUE

- Size of the LOG_IO_QUEUE (in 8 KByte pages).



The log queue serves to prevent load spikes and to enable larger block sizes when writing logs.

With applications with a high logging rate, the log queue can overflow and thus cause wait times without COMMIT statements. If this occurs frequently, the log queue can be enlarged to prevent unnecessary wait times for transactions that have not yet been completed.

The first thing to check is the I/O times for writing the log pages. Poor I/O times increase the chances of log queue overflows. They cause the database to have poor response times for write transactions.

Applications that almost exclusively read require only a small value.

Values: Default: 50 pages

- Min: 8, Max: Size of the log area
- SAP systems: > (2 * MAXUSERTASKS)
- Online change: No

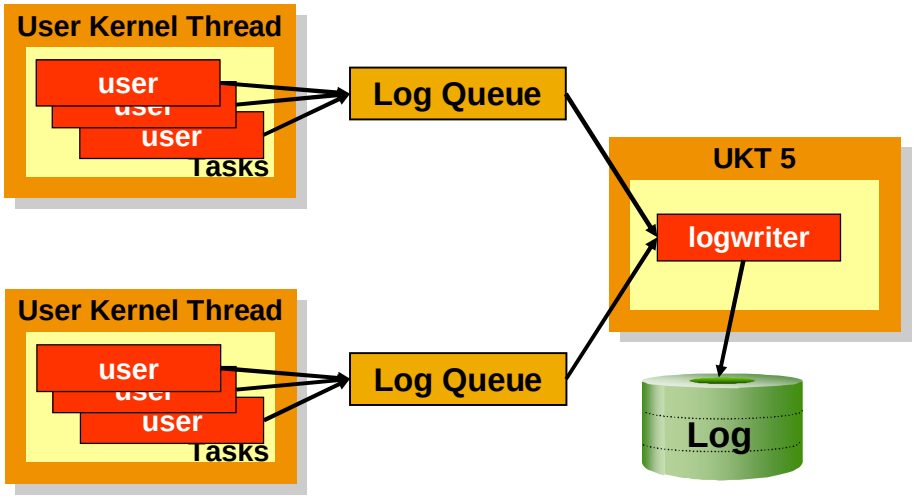
The system view LOGSTATISTICS displays the number of log queue overflows

- SELECT "Queue Overflows" FROM INFO_LOG_STATE

Log Queue Count

LOG_QUEUE_COUNT

- Number of log queues in the system.



As of version 7.6, MaxDB supports multiple log queues. This prevents collisions during access to the memory areas.

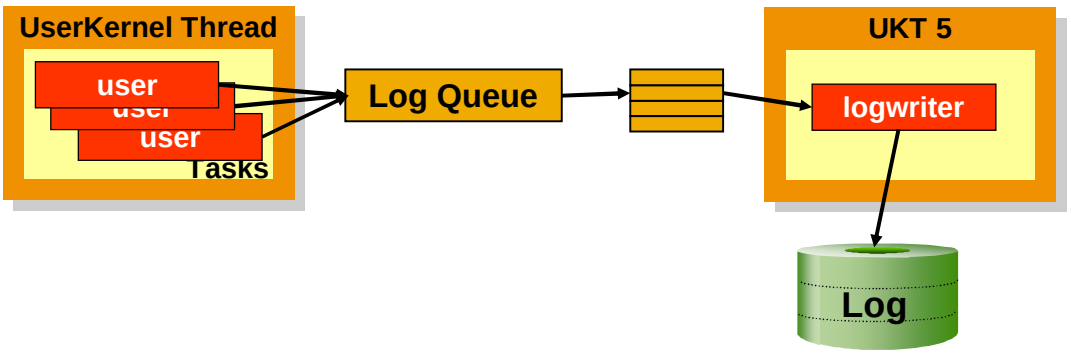
In the standard, the number of log queues is the same as the value for MAXCPU. Thus every UKT with user tasks writes to its own log queue. So users can no longer collide with each other, but only with the log writer.

Values: Default: MAXCPU
Min: 1, Max: MAXCPU
Online change: Yes

Maximum Number of Log Pages per I/O

LOG_IO_BLOCK_COUNT

- The log writer can combine multiple pages in a log queue into a block for an I/O request.
- The parameter defines the maximum block size for an I/O request for the log area.
- This block size is also used for redo during imports from the log area.

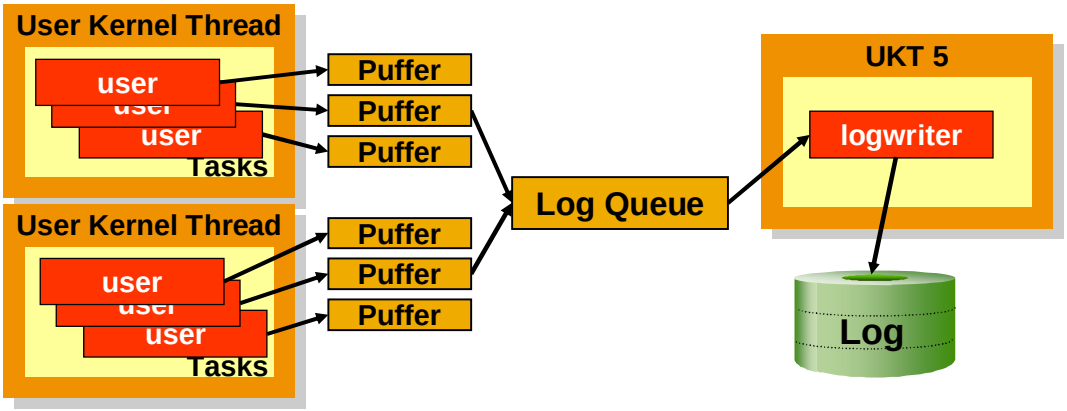


Values: Default: 4 (this amounts to 32 KB with 8 KB Pages)
Min: 4, Max: 32
Online change: No

Transaction Buffer for Redo Log Entries

LOCAL_REDO_LOG_BUFFER_SIZE

- To minimize the number of accesses to the log queue the database may create a buffer per transaction collecting the redo-log entries until the buffer is filled or up to a commit.
- This minimizes the number of collisions for log queue accesses.



When operating in an SMP environment with multiple CPUs (MAXCPU > 1), collisions can occur due to parallel accesses to the log queue.

Such collisions can be minimized - especially for larger transactions - by creating a transaction-dependent buffer for redo log entries.

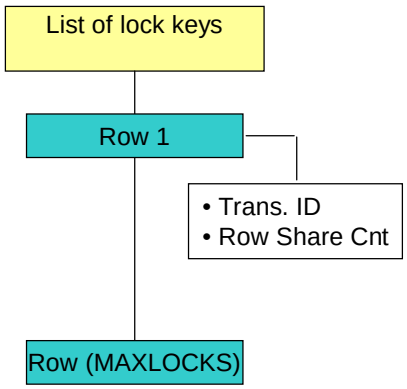
Values:	Default: 0	Transaction-dependent buffers for redo log entries are not used
	>0	Size per transaction-dependent buffer in bytes

Online change: No

Lock Lists

MAXLOCKS

- Maximum number of row locks that can be held or requested simultaneously.



MAXLOCKS specifies the maximum size of the lock lists for row and table locks.

MAXLOCKS is dependent on the isolation level, the number of simultaneously active users and the applications.

If MAXLOCKS is exhausted, statements that request locks are rejected (-1000 "TOO MANY LOCK REQUESTS").

MAXLOCKS should be increased if the following regularly occurs:

- LOCK LIST ESCALATIONS occur or
  LOCK LIST MAX USED ENTRIES is equal to MAXLOCKS or
  LOCK LIST AVG USED ENTRIES is nearly equal to MAXLOCKS

Alternatively, the number of write transactions can be reduced.

Values: Initial: 2.500

- Minimum: 500
- SAP systems: >= 150.000
- Excessive values for MAXLOCKS lead to longer search runs in the lock lists.

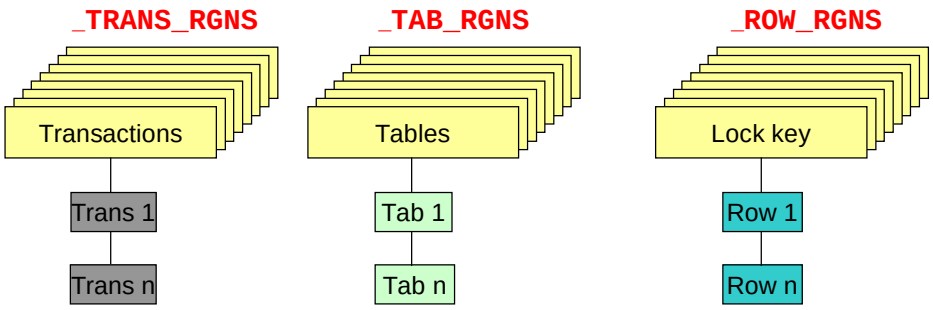
The criteria for increasing MAXLOCKS mentioned above can be queried:

- SQL statements
  - SELECT * FROM SYSDBA.LOCKLISTSTATISTICS
- DBAnalyzer
- SAP system: transaction DB50, (ST04).

Sections of Lock List

_TRANS_RGNS
_ROW_RGNS
_TAB_RGNS

- Number of autonomous units in which the different lock lists are devided.



SQL locks are entered in several lists:

- Transaction view
- Table view
- Row view

To prevent collisions of parallel accesses, several lists are used for each view.

Values: Initial: 8

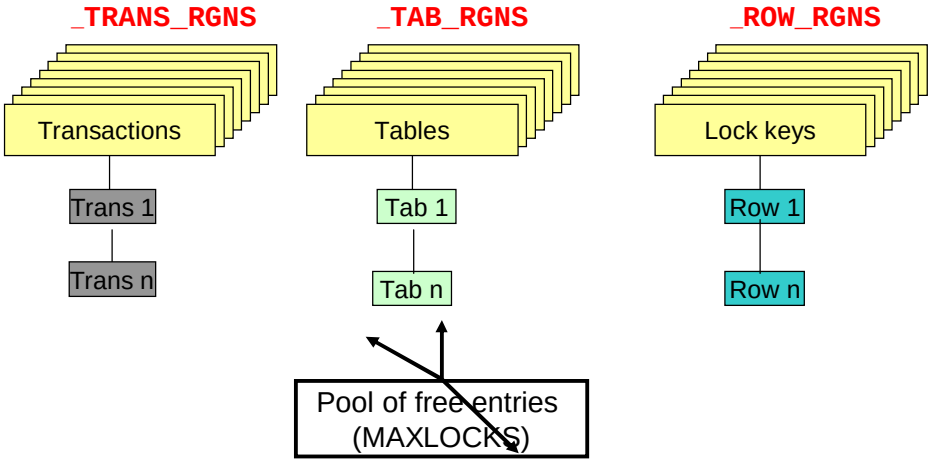
- Default: 8
- Min: 1, Max: 64

Online change: No

Enlargement of Lock Lists in Blocks

_LOCK_SUPPLY_BLOCK

- Number of locks provided for every individual lock list region (taken from the administration list of free entries), if all lock entries in the TRANS_REGION, TAB_REGION, ROW_REGION are in use.



If all the lock entries of the lock list are in use, their number can be increased by getting memory from the lock pool. This is done with the portion size parameter _LOCK_SUPPLY_BLOCK.

If the value is too large, eventually too much unneeded memory from the lock pool that could be used elsewhere will be tied to the list (upper limit via MAXLOCKS).

If the value is too small, memory may have to be retrieved several times in quick succession, which in turn can lead to collisions in the LOCKPOOL region.

Collisions can be queried with:

- x_cons <SERVERDB> show regions
- dbmcli –d <dbname> -u <dbmuser,passwd> -n <server> show regions
- SELECT * FROM SYSMON_REGION WHERE REGIONNAME = 'LOCKPOOL,
- DBAnalyzer

Values: Default: 100

- Min: 10, Max: 100000
- Online Change: No

Deadlock Detection

DEADLOCK_DETECTION

- Maximum search level for deadlock detection.

- Values:
  - 0: Deadlock detection is disabled, i.e. deadlocks are only released by the REQUEST_TIMEOUT.
  - n > 0: Deadlocks are detected up to the defined search level and immediately released.
  - Initial: 4
  - 0 - 10.000

A higher value than the initial value leads to significant costs. It is only advisable if an application causes serious problems with deadlocks and this cannot be remedied on the application side.

Values: See above

Online change: No

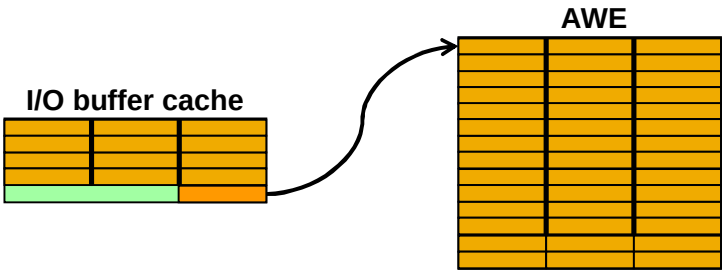
AWE Support for Windows

USE_MEM_ENHANCE

- AWE support for the I/O buffer cache (as from Windows/2000)

Note: AWE is no longer supported in versions >= 7.6!

- Values:
  - YES: AWE memory is used.
  - NO: AWE memory is not used (default).



AWE stands for Address Windowing Extensions.

The AWE memory is used exclusively for the data cache.

The database automatically generates a mapping table in the data cache. The parameter CACHE_SIZE must be set to a high enough value that there is enough space for the converter and the mapping table.

Limit the size of the used AWE memory with the parameter MEM_ENHANCE_LIMIT.

AWE is only used with Windows.

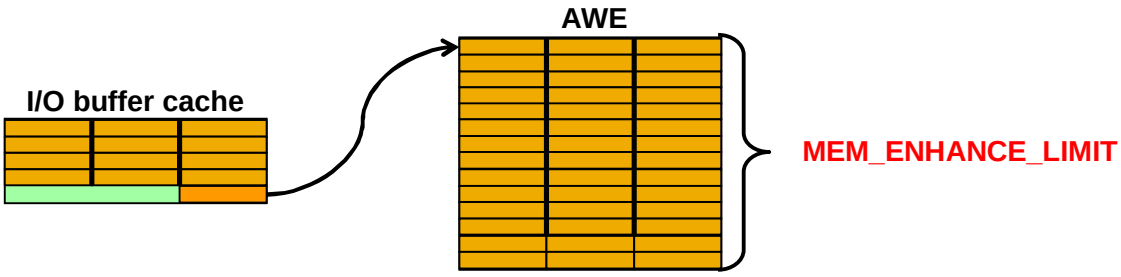
Values: See above
Online change: No

Maximum Size of Used AWE Memory

MEM_ENHANCE_LIMIT

- Limitation of the maximum size of AWE memory that can be used for the database instance.
- Values:
  - 0: The database at start time uses as much AWE memory as it can get.
  - >0: Limitation of AWE memory (in MB).

Note: AWE is no longer supported in versions >= 7.6!



If the parameter USE_MEM_ENHANCE is set to YES, the parameter MEM_ENHANCE_LIMIT should be set to a value > 0 to prevent memory bottlenecks with other applications or databases.

Values: Default: 0
Min: 0, Max: 2147483640
Online change: No

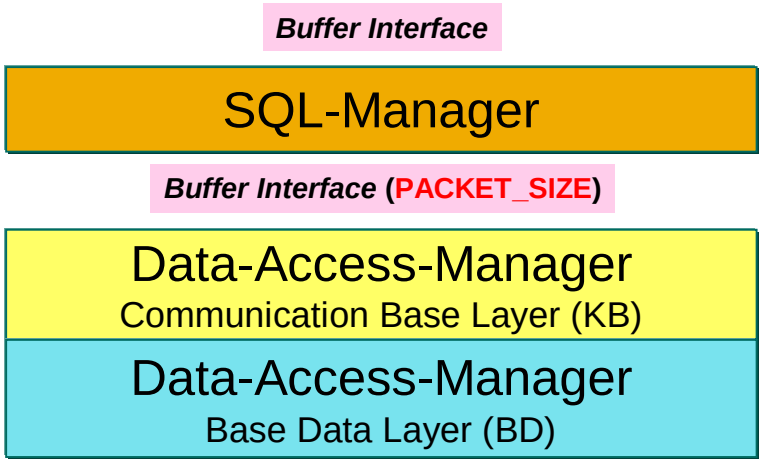
Communication, I/O

_PACKET_SIZE	Maximum length of communication packets
_USE_IOPROCS_ONLY	I/O only allowed through special threads
XP_MAXPAGER	Number of tasks for writing data pages
_RESTART_TIME	Time passed between two savepoints
_DW_IO_AREA_SIZE	Data cache area relevant for savepoints
_DW_IO_AREA_FLUSH	Point of time when writing starts
_DW_LRU_TAIL_FLUSH	Point of time when writing starts
DATA_IO_BLOCK_CNT	Max. number of blocks per I/O operation
BACKUP_BLOCK_CNT	Number of blocks per I/O for backup/restore
_IOPROCS_PER_DEV	Number of I/O threads per volume
_IOPROCS_SWITCH	Threshold value for the use of further I/O threads
_IOPROCS_FOR_READER	Additional I/O threads only for read requests
_READAHEAD_BLOBS	Asynchronous read of pages of a LONG value
AUTO_RECREATE_BAD_INDEXES	Automatic creation of indexes during restart
VOLUMENO_BIT_COUNT	Maximum size or number of data volumes
SET_VOLUME_LOCK	Enforce lock of volumes while attached
USE_OPEN_DIRECT	By-passing the Linux file system cache

Communication Packet

_PACKET_SIZE

- _PACKET_SIZE limits the maximum length of the communication packet and thus the maximum length of an individual SQL statement.



Enlarging the communication packet accelerates data transfer for mass commands and enables longer SQL statements, but it also requires more memory.

Values: Default: 32.768

- Min: 16,384, Max: 131.072
- SAP systems (ASCII): >= 36864 (see OSS note 140739)
  SAP systems (UNICODE): >= 65536
  SAP BW: >= 66560 (see OSS note 545385)
- SAP systems (from version 6.40) = 131072
- Online change: No

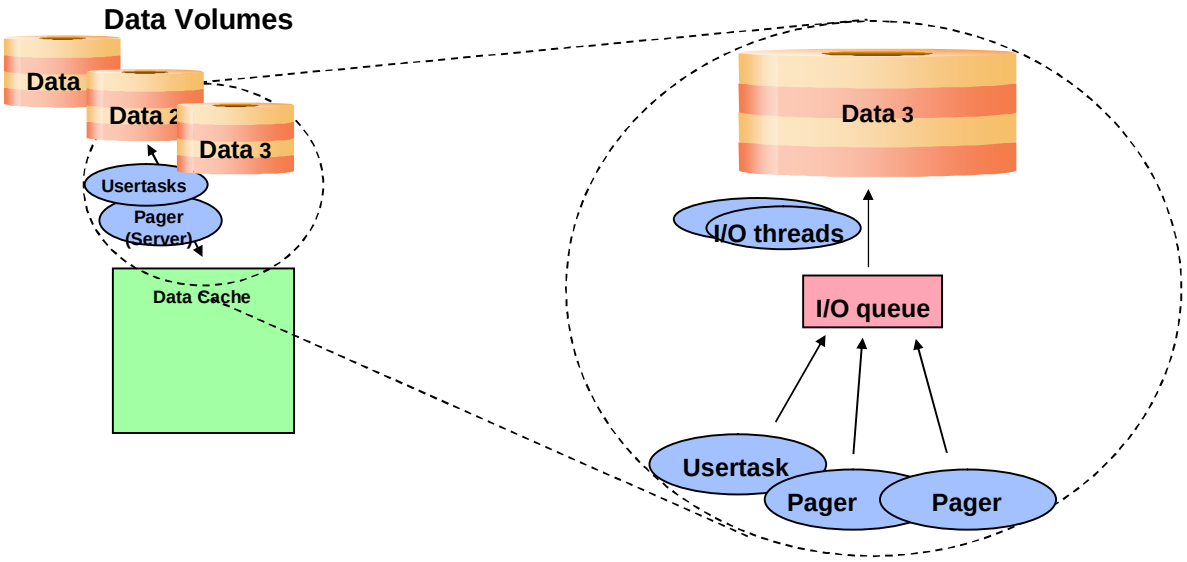
Typical errors if PACKET_SIZE is too small:

- -743 Input string too long
- -1104 Too complicated SQL statement
- -1114 Communication packet too small

Asynchronous I/O

_USE_IOPROCS_ONLY

- The parameter specifies, if I/O may exclusively be done by special I/O threads or may also be done by the UKT itself.



UKTs can themselves call I/O operations if

- the parameter _USE_IOPROCS_ONLY is set to "NO" and
- only one user task in the UKT is not in "Connect Wait" status or only one task is running in a UKT (e.g. log writer)

The I/O request is then not put in a queue and processed by the I/O thread.

The individual I/O operation can be executed more quickly if the UKT does not need to request an I/O thread.

If a user task executes an I/O request itself, other tasks cannot work until it is finished. The UKT is blocked and waits for the reply to the I/O request. This option can compromise performance in parallel operation.

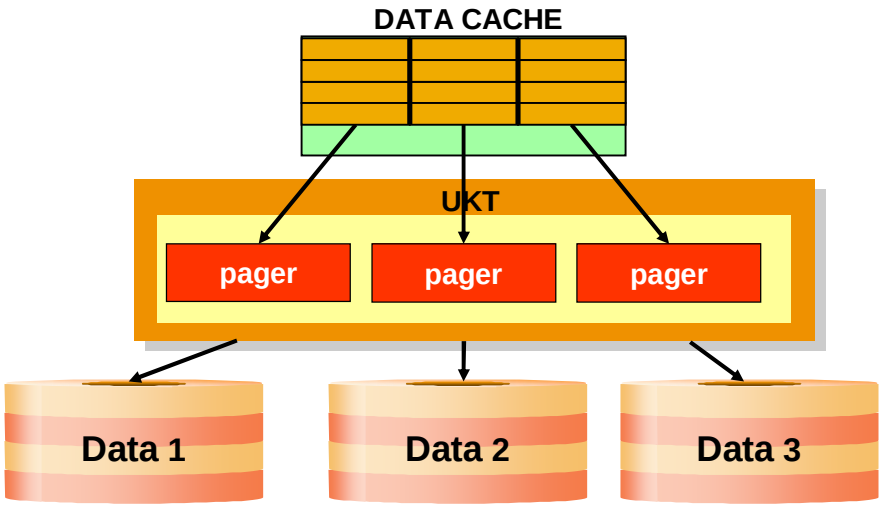
Values: Default: NO

- Online change: Yes

Pagers

XP_MAXPAGER

- Number of tasks of the database kernel, which write asynchronously changed pages of the data cache to the disks at savepoint time.
- They additionally become active between savepoints to reduce the duration of a savepoint and to prevent paging.



The pager tasks read changed pages from the data cache and put them in the queues of the I/O threads. In general, they do not execute the I/O request themselves (see parameter _USE_IOPROCS_ONLY).

As of version 7.4.03, with an online restart, the pagers read the converter in parallel.

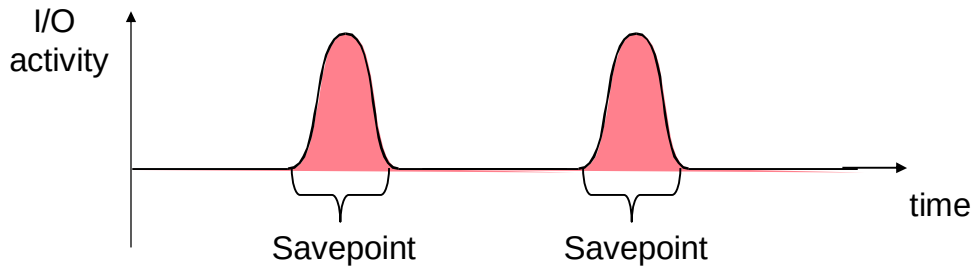
Values: Min: 1, Max: 64

- Default: Maximum of _DATA_CACHE_RGNS, CONVERTER_REGIONS and MAXDATAVOLUMES
- Online change: No

Time between two Savepoints

_RESTART_TIME

- Time period in seconds between start of two savepoints.



If the _RESTART_TIME value is increased, the number of savepoints within a time unit - and thus the workload associated with it - decreases. This can, however, prolong the time required for a restart after a system crash.

Values: Default: 600

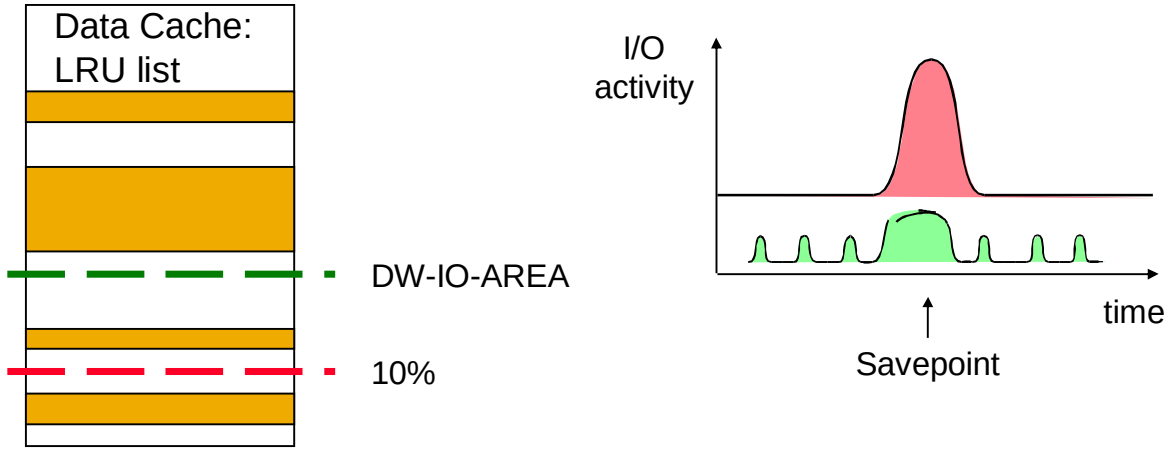
- Min: 0, Max: 100.000

SAP does not recommend changing the value of this parameter. If desired, you can influence the behavior of savepoints with the parameters _DW_IO_AREA_SIZE,_DW_IO_AREA_FLUSH and _DW_LRU_TAIL_FLUSH.

I/O Workload of Savepoints

When do pagers become active?

- Savepoint time
- Too many pages are changed within the whole cache.
- Too many pages are changed at the end (10%) of the LRU list.



Occasionally savepoints can lead to undesired I/O load spikes as all the changed pages in the data cache are written to the data volumes.

These I/O spikes can be mitigated if changed pages are written in parallel prior to the savepoint. This activity is done by the pagers, whose behavior can be controlled using the parameters described in the following.

Nevertheless, the overall I/O load rises in this case as pages that were changed several times before the savepoint also have to be written several times.

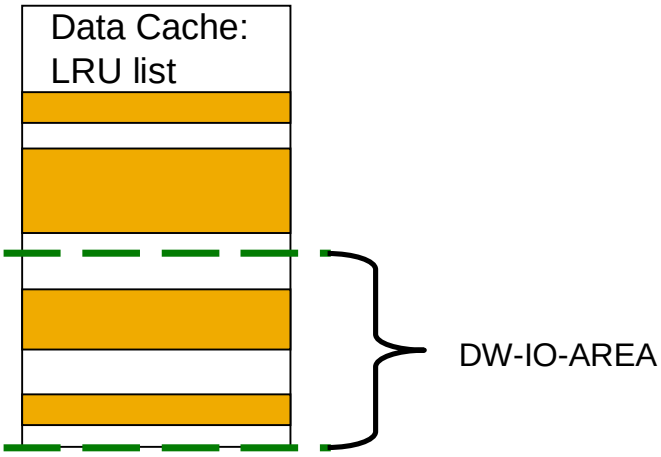
Only those pages from a part of the cache, the DW-IO-AREA, are written. The size of this area is controlled by a parameter.

The LRU list (Least Recently Used) is a concatenation of data pages. The pages used most recently are generally at the front.

Area for Pagers

_DW_IO_AREA_SIZE

- Size of the data cache area, where pagers become active between savepoints (in percent).



This parameter defines the area of the data cache in which the pagers work between savepoints.

Pagers write only those changed pages out of the cache that are at the back end of the LRU list, that is, the area defined by _DW_IO_AREA_SIZE.

A large value reduces the savepoint I/O load more, but increases the current I/O load.

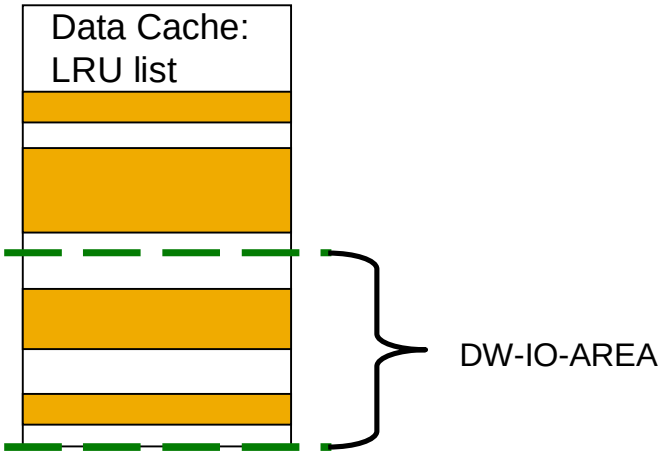
Values: Default: 50

- Min: 10, Max: 90
- Online change: No

Threshold Value for Activation of Pagers (I)

_DW_IO_AREA_FLUSH

- Number of changed pages within the whole data cache (in percent), when pagers become active.



This parameter specifies an event at which the pagers become active.
A large value reduces the savepoint I/O load more, but increases the current I/O load.

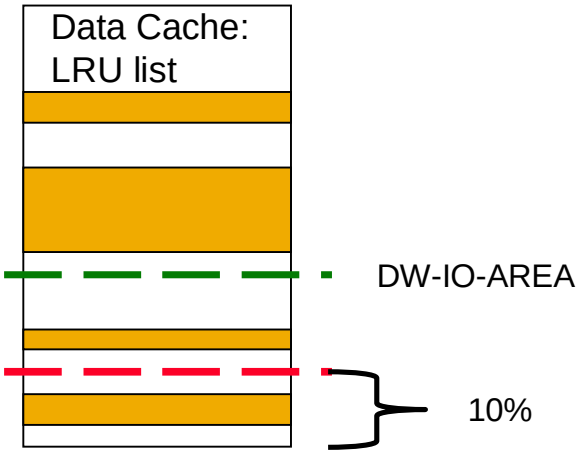
Values: Default: 50

- Min: 30, Max: 80
- Online change: No

Threshold Value for Activation of Pagers (II)

_DW_LRU_TAIL_FLUSH

- Number of changed pages within the last 10% of the data cache (according to the LRU list) in percent, when pagers become active.



This parameter specifies an event at which the pagers become active.

The purpose of this mechanism is to ensure that there are always enough free pages at the end of the LRU that user tasks are not forced to displace pages from the data cache.

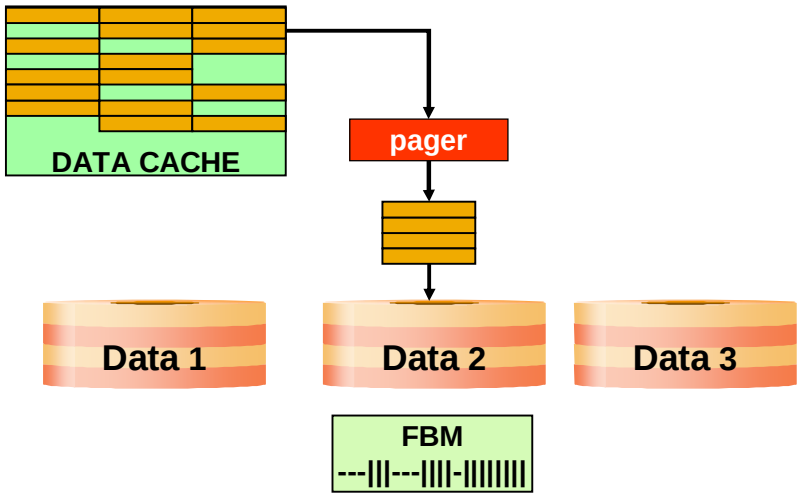
Values: Default: 25

- Min: 10, Max: 80
- Online change: No

Blocked I/O

_MULT_IO_BLOCK_CNT

- Size (in 8 KByte pages) of an I/O operation by the pagers.



Pagers can combine data pages and write them with an !/O operation (vector I/O).

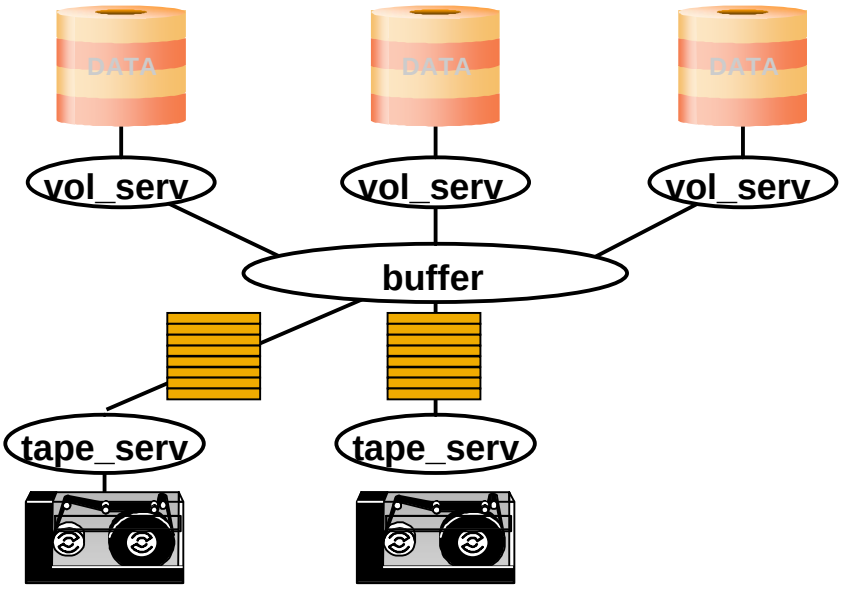
Values: Default: 64

- Min: 4, Max: 128
- Online change: No

Blocked I/O for Backup and Restore

BACKUP_BLOCK_CNT

- Size (in 8 KByte pages) of an I/O operation to backup media



During a backup, server tasks write data pages in blocks to the backup medium.

Values: Default: DATA_IO_BLOCK_COUNT

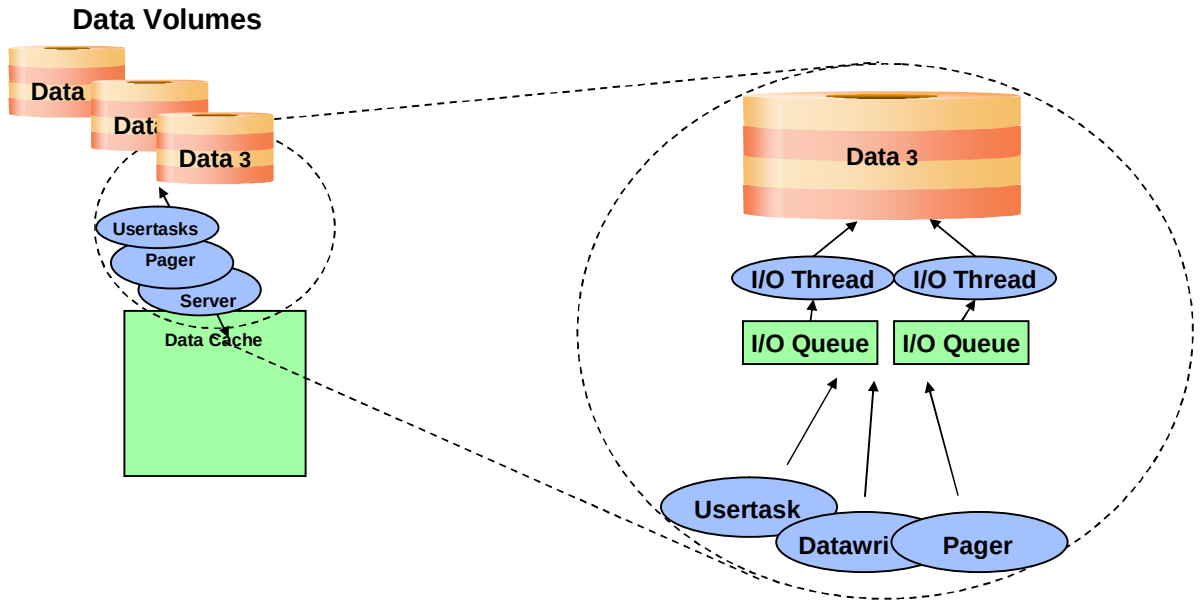
Min: 4, Max: 128
Online change: No

You can also specify the blocking factor when defining the backup medium with the DBMGUI or with the dbmcli comand medium_put .

I/O Threads per Volume

_IOPROCS_PER_DEV

- Number of I/O threads per data und log volume



Subsequent I/O operations in a volume are distributed among several threads.

Values: Default: 2 (recommended), WINDOWS: 1

Default: 1 when using asynchronous I/O of the operating system
2 for exclusive use of own asynchronous I/O

Min: 1, Max: 10

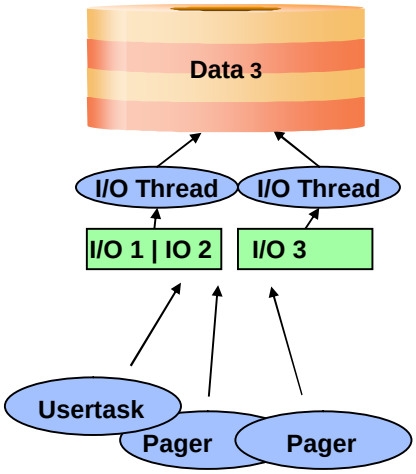
Online change: No

The number of I/O threads per volume should be raised if the database was configured with only a small number of data volumes but the I/O system allows more parallel I/Os. It is advantageous to use an appropriately high number of data volumes because it reduces the frequency with which pagers and user tasks have to check whether an I/O queue is already occupied.

Threshold Value for Use of Further I/O Threads

_IOPROCS_SWITCH

- Number of requests accepted by an I/O queue before another I/O thread gets the request.
- _IOPROCS_SWITCH requires_IOPROCS_PER_DEV to be higher than 1.



As soon as more requests than _IOPROCS_SWITCH are in the queue of an I/O thread, each additional I/O request is assigned to another I/O thread.

With up to two I/O requests, it is generally good to have only one thread, saving the time required for a thread change.

Values: Default: 2 (recommended)

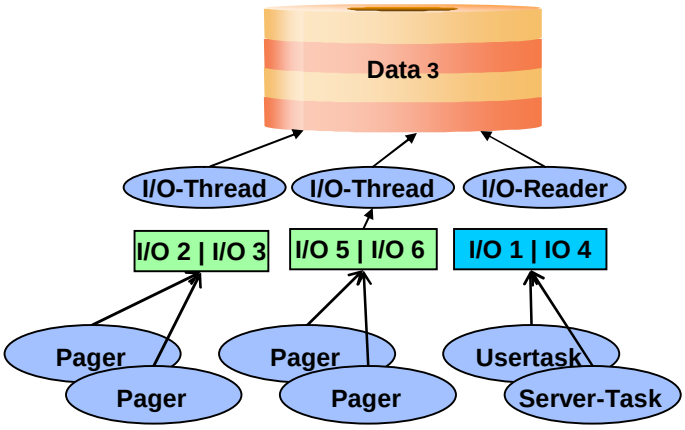
- Min: 0
- Online change: Yes

If the parameter _IOPROCS_PER_DEV has been increased because the I/O system can handle more parallel I/Os than there are configured data volumes, the parameter _IOPROC_SWITCH should be set to 1.

Additional I/O Threads for Read Requests

_IOPROCS_FOR_READER

- The number of additional I/O threads that only process read requests. This prevents "starvation" of read requests during savepoints.



During a savepoint, the pagers can put a large number of I/O requests in the queues of the I/O threads. In particular, this happens when the number of data cache segments is significantly higher than the number of data volumes.

If _IOPROCS_FOR_READER is set to > 0 , for read requests the I/O first checks whether the number of queue entries for read-threads exceeds _IOPROCS_SWITCH. If this is not the case, it puts the request in the checked queue.

The I/O threads for read and write requests are only used for read requests if there are at least _IOPROCS_SWITCH requests in all queues for read threads.

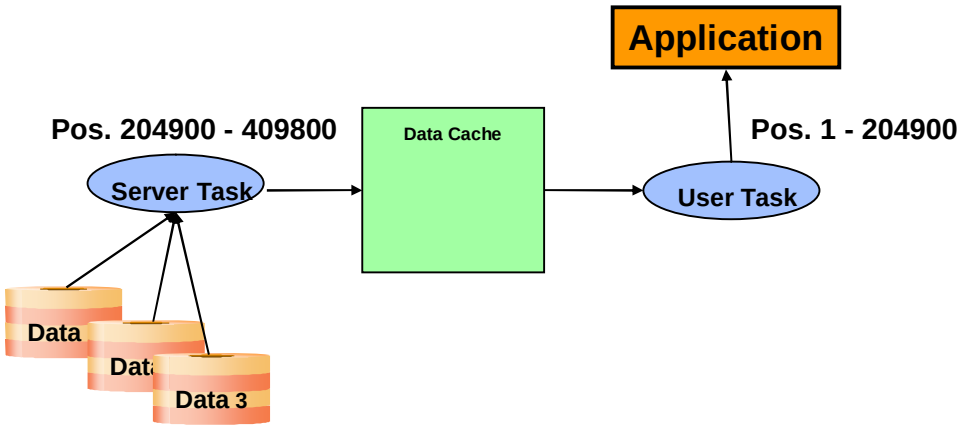
Values: Default: 0
Min: 0, Max: 10
Online change: No

This parameter was introduced with version 7.5.00 Build 35 and 7.6.00 Build 27.

Asynchronous Read of BLOB Pages

_READAHEAD_BLOBS

- Specifies if read operations of BLOBs should be optimized.
- A server task reads data pages from the data volumes during read of BLOB pages. In the meantime the user tasks may deliver previously read pages to the application.



When reading larger LONG values, it is a good idea to continue reading asynchronously from the volumes while sending a read package to the application. This is done by a server task if the length of the LONG value to be read exceeds the value of the parameter _READAHEAD_BLOBS.

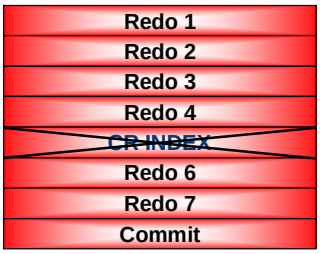
Values: Default: 25

- Min: 2 * _PACKET_SIZE / 8196 Max: 262144
- Online change: Yes

Automatic Creation of Indexes During Restart

AUTO_RECREATE_BAD_INDEXES

- With default setting a CREATE INDEX statement in the log is not executed during restart/redo. Thus restart and recovery times are minimized. The index is marked as „bad index“ within the catalog.
- The database recreates the indexes during Restart/Redo, if the parameter is set to YES.



You can get "bad indexes" with the following statement:

Select * from INFO_BAD_INDEXES

Use the following call to recreate the indexes:

dbmcli –d <database_name> -u <dbm_user>,<password> –uSQL <userid>,<password>
sql_recreateindex

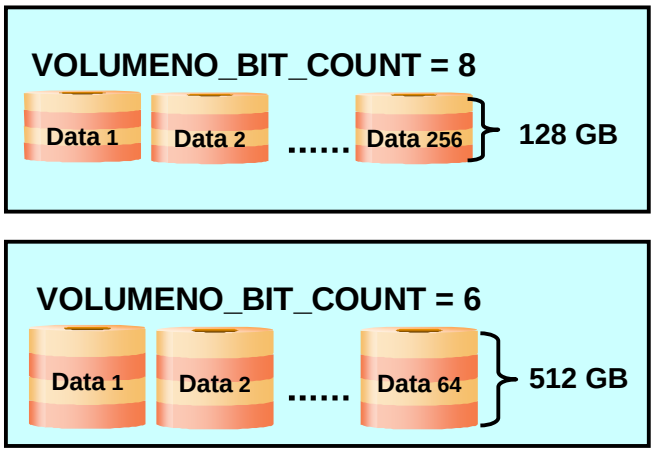
Values: Default: NO

- Online change: No

Maximum Size or Number of Data Volumes

VOLUMENO_BIT_COUNT

- Number of bits in a four byte converter block address reserved for the logical device number which identifies the data volumes.The rest of the block address is used for the device offset
- The size of volumes may vary to the benefit or disadvantage of the number of data volumes.



In its current implementation, the MaxDB converter provides 4 bytes (= 32 bits) for addressing data blocks. In the standard setting, one byte (= 8 bits) is used for the volume number and 3 bytes (= 24 bits) for the block position in the volumes. Thus with 8 KB - pages, the maximum size for MaxDB instances is 32 TB (2^32 * 8KB).

You can configure the number of bits that the converter management uses for addressing the volumes.

That enables you to configure the database so that it supports larger volumes, for example. If the database is configured to support larger volumes, the maximum number of volumes sinks.

Values (calculation for 8 KB pages):

Default = 8 ➔ max. 256 volumes, max. size 128 GB

Min: 6, Max: 12

Default = 6 ➔ max. 64 volumes, max. size 512 GB

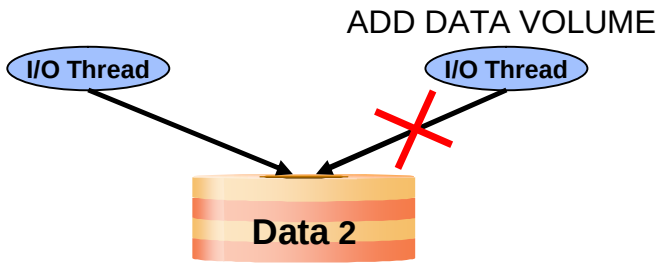
Online change: No

Change the value for VOLUMENO_BIT_COUNT only after consultation with MaxDB Support.

Lock on Attached Volumes

SET_VOLUME_LOCK

- An enforced lock prevents an ADD DATA VOLUME with a volume that is already in use by this or another instance.
- In case of NFS mounted volumes (e.g. using Network Attached Storage NAS) it may be reasonable not to set the lock.



From version 7.5, in the standard setting MaxDB sets a lock on open volumes that are in the file system. You can change this behavior using the parameter SET_VOLUME_LOCK.

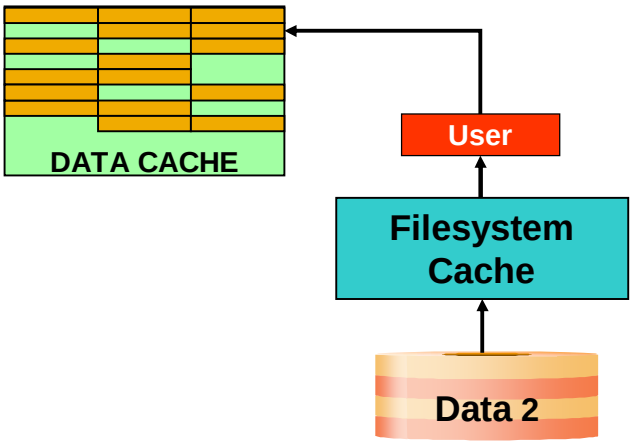
Values: Default: YES The database requests a lock when a volume is opened.

Online change: No

Bypassing the Linux File System Cache

USE_OPEN_DIRECT

- Use of I/O flag O_DIRECT to by-pass the Linux filesystem cache



Depending on the I/O system being used, Linux supports direct I/O, bypassing the file system cache.

This option is supported as of Linux kernel version 2.4.18. In older versions of the Linux kernel, this option is ignored.

If the parameter USE_OPEN_DIRECT is set to YES although the option O_DIRECT is not supported for the I/O system, the database kernel cannot open the files in the volumes upon starting.

Values: Default: NO (I/O call without flag O_DIRECT)
YES (I/O call with flag O_DIRECT)

Online change: No

Process Structure, CPU Usage

_TASKCLUSTER_01 ... _03	Distribution of tasks to threads
MAXCPU	Number of intensely used processors
LOAD_BALANCING_CHK	Load balancing between UKTs
LOAD_BALANCING_DIF	Threshold value for load balancing
LOAD_BALANCING_EQ	Precision of measurement for load balancing
ALLOW_MULTIPLE_SERVERTASK_UKTS	Distribution of server tasks to UKTs
MAXRGN_REQUEST	Stop running tasks
XP_MP_RGN_LOOP	Busy Waiting (BW) during collisions
_MP_RGN_QUEUE	BW without boundaries/ no queuing in case of collisions
_MP_RGN_DIRTY_READ	Dirty Read in region management/ usage of test and set operations
_MP_RGN_BUSY_WAIT	Busy Waiting for released regions/ after test and set op failed
_MP_DISP_LOOPS	Dispatcher loops
_PRIO_BASE_...	Base priority per task state/ assignment of priorities to tasks
_PRIO_FACTOR	Additional priorities for lockholders
_IOPROCS_FOR_PRIO	Number of I/O threads for prioritized tasks
_MP_DISP_PRIO	Interruption of tasks
_MP_RGN_PRIO	Favor prioritized tasks

Distribution of Tasks to User Kernel Threads

_TASKCLUSTER_01 to _03

- Specifies how the tasks of the database kernel are assigned to operating system threads.
- This parameter depends on the value of MAXCPU and should not be changed.

- Example:
  - MAXUSERTASKS = 40, MAXCPU = 2
  - TASKCLUSTER_01 = 'tw;al;ut;2000*sv;10*ev,10*gc'
    TASKCLUSTER_02 = 'ti,100*dw;20*us;'
    TASKCLUSTER_03 = 'equalize'

Tasks between two semi-colons are combined to make a thread.

Meaning of the example:

Trace writer, log writer und utility task each run individually in their own thread.

Up to 2000 (practically all) server tasks are combined in a single thread.

Garbage collectors and event tasks run together in a thread.

Timer and up to 100 pagers run in one thread.

The number of threads containing user tasks is limited to MAXUSERTASKS/2.

"equalize":
User tasks are distributed as evenly as possible across different threads.

"compress":
The maximum possible number of user tasks (in this case 20) is processed in a thread before a new thread is started.

"allinone":
All tasks run in one thread.

Online change: No

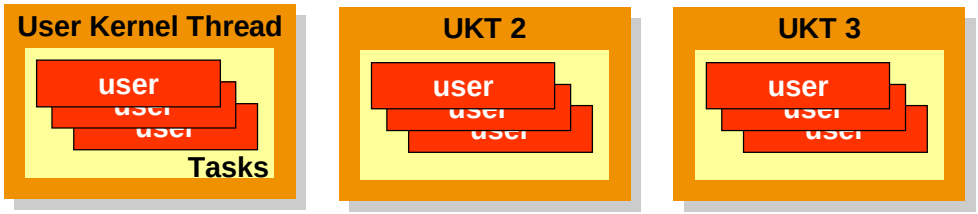
Warning:

Do not change these parameters without the explicit recommendation of MaxDB Support. Any changes would be reset the next time the parameters are calculated.

Number of Intensively Used CPUs

MAXCPU

- Maximum number of CPUs used by the database instance for concurrent processing of UKTs containing user tasks.



This parameter serves to inform the database kernel that multiple CPUs can be used.

At the same time, it allows the database system to restrict CPU usage. Such a restriction only applies to UKTs that contain user tasks. Other UKTs continue to access any number of CPUs even if the value for MAXCPU is reduced.

Generally speaking, MAXCPU indicates the number of CPUs simultaneously subject to intensive usage.

The value for MAXCPU strongly influences the distribution of database kernel tasks to the operating system threads (parameter _TASKCLUSTER).

If the computer is used exclusively as a database server, MAXCPU should correspond to the number of CPUs the computer has; otherwise the value should be reduced to free up some CPUs for other applications.

Values: Default: 1

SAP central system: 1/3 - 1/5 of available CPUs

Dedicated database serve with up to 7 CPUsr: 100% of available CPUs

Dedicated database server with more than 7 CPUs: 100% of available CPUs

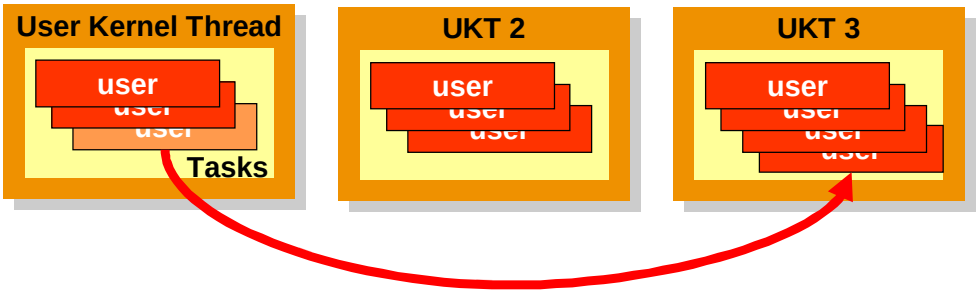
-1

Online change: No

Load Balancing Between UKTs

LOAD_BALANCING_CHK

- Time interval for checking the load on User Kernel Threads. If the value of the parameter is bigger than 0, tasks of an UKT with high workload can be moved to an UKT with lower load.



Load balancing enables optimal exploitation of all threads and thus of all the CPUs allocated to the database.

After the time interval of LOAD_BALANCING_CHK seconds, the database kernel searches for a task to move to another UKT. This is helpful when one UKT has a particularly heavy load and another UKT has a smaller load.

Between the checks after LOAD_BALANCING_CHK seconds, statistics are collected. The greater the time for gathering the data, the more meaningful the UKT load statistics that result. With smaller values, the shifting that occurs may not be optimal.

Load balancing is particularly useful for liveCache instances. These often run very CPU-intensive LCA routines over long periods. Multiple LCA routines should not work sequentially on one CPU if another CPU is free.

In OTLP operation, unbalanced load distribution among the UKTs is usually due to poorly-optimized statements with high CPU loads for a single job. For this reason, such statements should be identified and optimized before load balancing is employed.

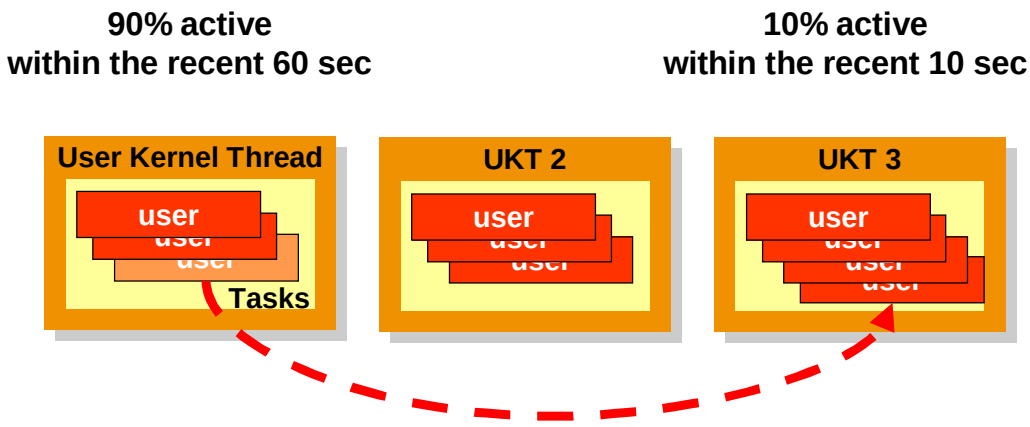
Values: Default: 0
Min: 4, Max: 600

Online change: No

Threshold Value for Load Balancing

LOAD_BALANCING_DIF

- Percentage of workload difference, when the distribution of tasks is started.



To assess the load on a UKT, the database kernel determines the time in which the thread is active.

The parameter LOAD_BALANCING_DIF indicates, in percent, the minimum difference between the CPU loads of two threads above which tasks are shifted.

From version 7.5, the database console provides information about tasks that have been moved:

x_cons <dbname> show moveinfo
x_cons <dbname> show moveinfo
dbmcli –d <dbname> -u <dbmuser,passwd> -n <server> show regions

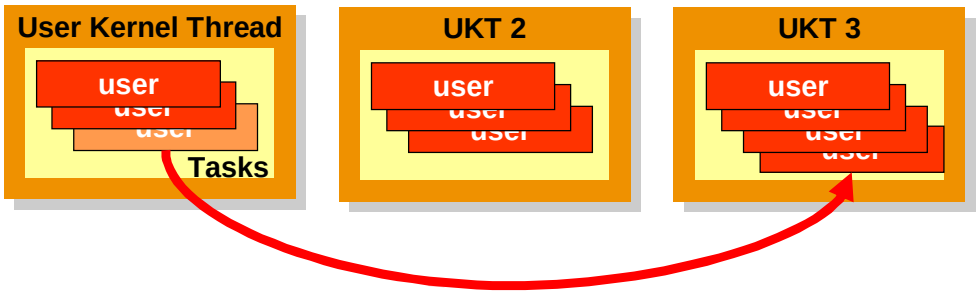
Values: Default: 10
Min: 0, Max: 99

Online change: No

Precision of Measurement for Load Balancing

LOAD_BALANCING_EQ

- Specifies the percentage the internal time values must differ before they are not treated as equal.



Moving tasks (task moving) is time-consuming. It should only be done when it is expedient.

Recorded time data for activities are considered equal if the difference between them does not exceed a certain percentage. The parameter LOAD_BALANCING_EQ defines this percentage.

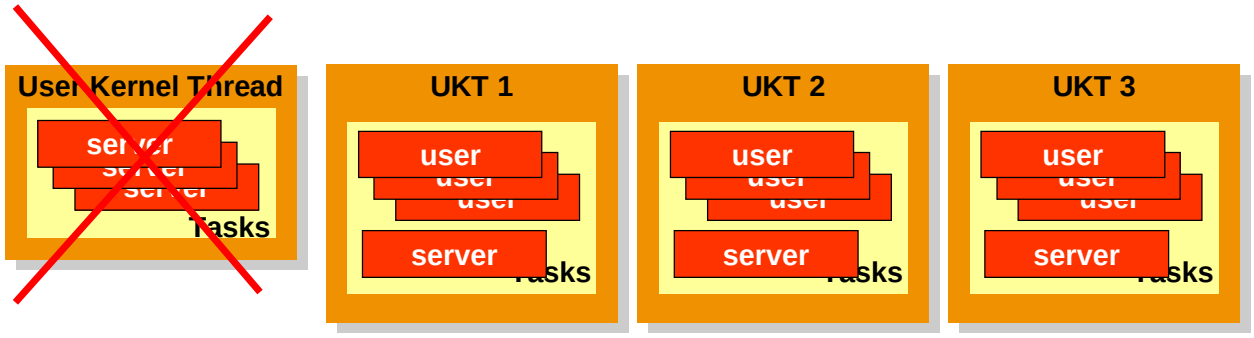
Values: Default: 5
Min: 0, Max: 50

Online change: No

Distribution of Server Tasks to UKTs

ALLOW_MULTIPLE_SERVERTASK_UKTS

- Specifies if server tasks can be distributed to UKTs containing user tasks.
- Values:
  - YES: Distribution of server tasks to UKTs with user tasks
  - NO: Server tasks are configured within their own thread
  - Default: NO



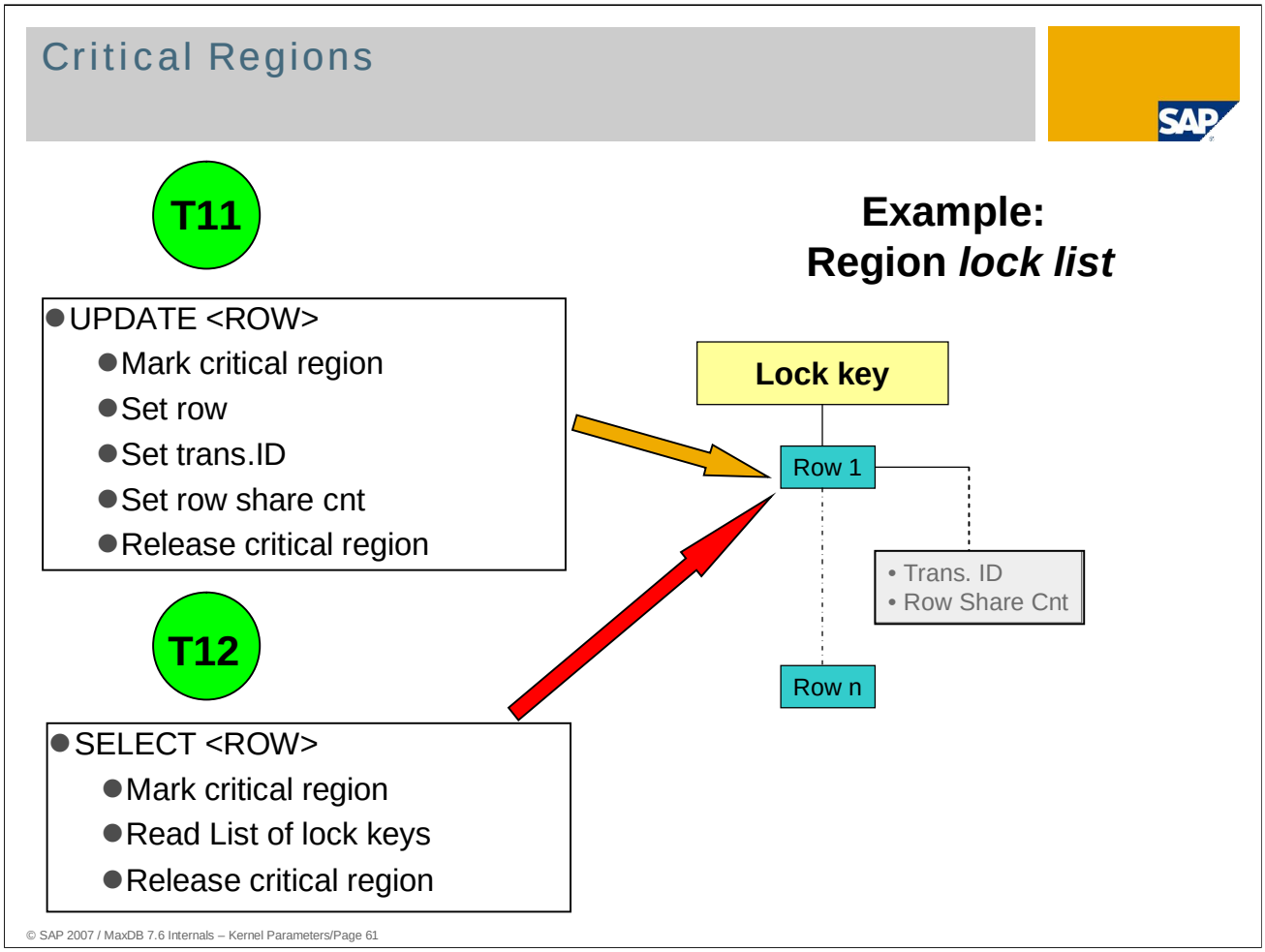
With the standard setting, server tasks are assigned to a UKT. In systems with mayn data volumes and fast I/O, the CPU load generated by the server tasks can lead to a bottleneck within the thread.

In liveCache instances in particular, such a bottleneck can lead to poor log recovery performance.

With the setting ALLOW_MULTIPLE_SERVERTASK_UKTS=YES, server tasks are distributed to the UKTs for user tasks. This can prevent a CPU bottleneck in the single UTK for all server tasks.

When the server tasks are distributed among the user task UKTs, users have to share the load per CPU with the server tasks. This can lead to compromised performance in online operation, for example while a backup is in process.

Online change: No



The present example should clarify the process of accessing a critical region.

Task 11 processes an update of a data record. Before it can be changed, a record has to be locked.
The lock is entered in the lock management, that is, in the lock key list, which is determined for this record by way of a hash algorithm.

During the change in the lock key list, values are changed and pointers set. While the change is taking place, the lock key list is not consistent.

Task 12, on an SMP machine, could want to read an entry from the lock key list at precisely the same time that task 11 is carrying out the change. The database does not allow this process because task 12 would be reading from an inconsistent list. So task 12 has to wait until task 11 has completed the change.

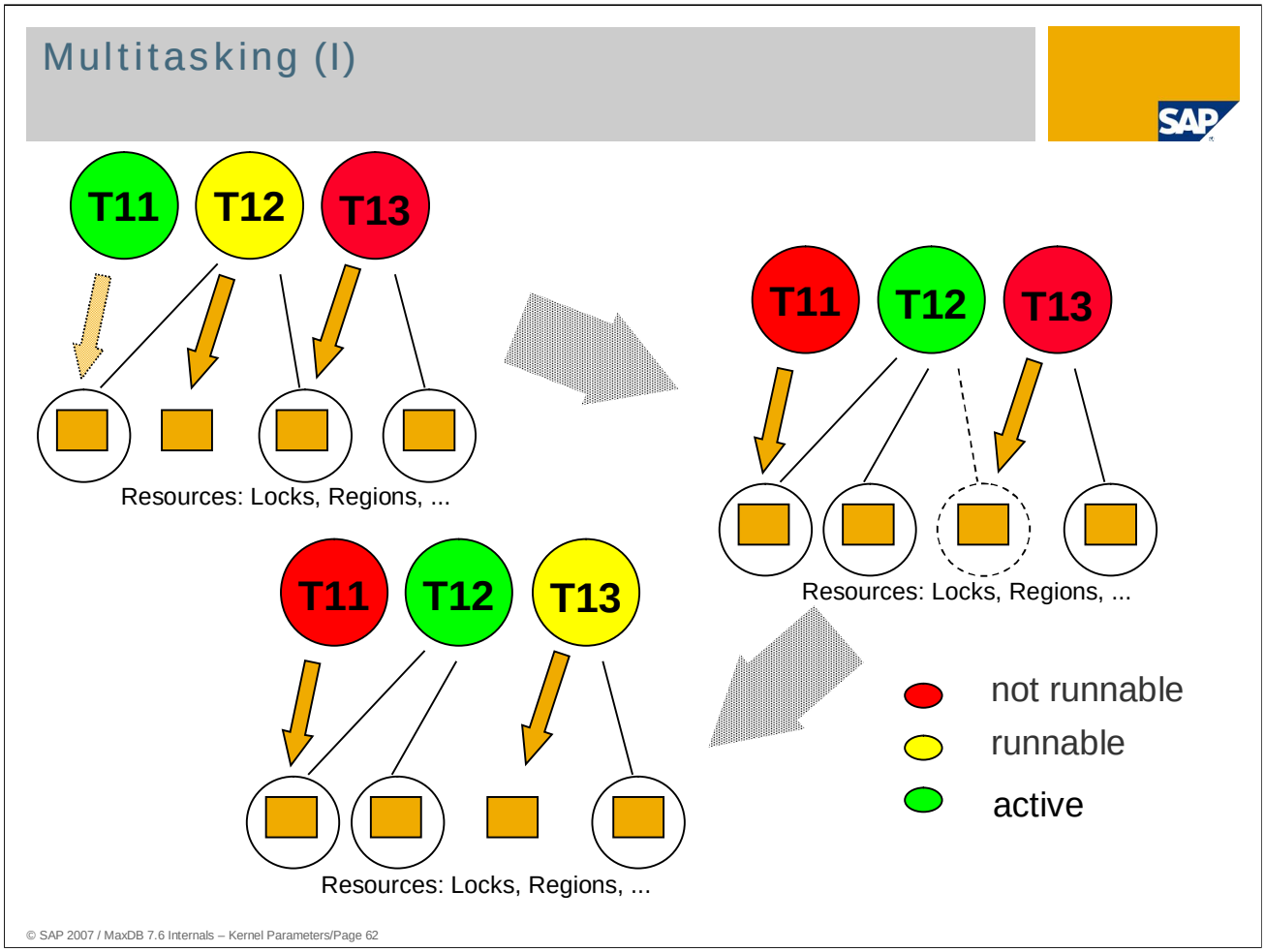
The source code, which can only be run by one task at a time, is designated a critical region. Defining regions enables synchronized access to resources that can be read or changed by multiple tasks in parallel.

Regions are only kept for a short time. The more tasks that want to access a region in parallel, the higher the risk of poor scaling. Scaling can be improved by shorter critical regions. Scaling can also be improved by shortening wait times and by defining multiple regions that allow parallel access (for example multiple lock key lists).

The command

- x_cons <SERVERDB> show regions
- dbmcli –d <dbname> -u <dbmuser,passwd> -n <server> show regions

shows which regions have been accessed how many times.



Cooperative multitasking

Multitasking in MaxDB means "cooperative multitasking." There is no central entity responsible for dispatching tasks. Tasks and threads independently decide on activation and prioritization using a number of simple rules.

A user task stops if it has nothing to do (e.g. it is waiting for an SQL statement from the application), has to wait for I/O or cannot obtain a lock for a database object or a region. If the required lock becomes free (or another of the reasons for the stoppage is removed), the task becomes executable again. It is then put in a queue and can be activated at the next opportunity.

The illustration:

1. figure:
Task T13 is not executable because it is waiting for a lock held by T12. Task T12 is executable. Task T11 is active and is currently requesting a lock held by T12.

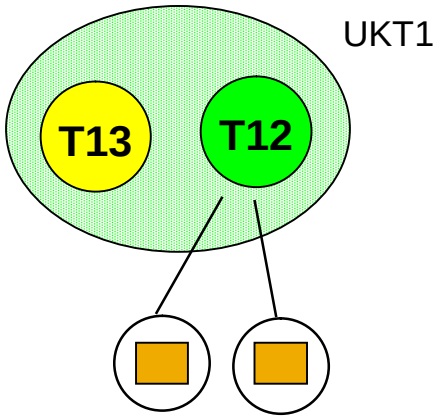
2. figure:
Task T11 had to stop and is not executable. T12 became active and now releases the lock for which T13 is waiting.

3. figure:
Task T13 is now executable and can become active if T12 stops and no other task is ahead of it.

Stop Running Tasks

MAXRGN_REQUEST

- Maximum number of times a task can try to obtain any region without being interrupted by another task in the same UKT.



If, by chance, a task gets all the regions it requests and otherwise encounters no obstacles (SQL locks or I/O) over a lengthy period of time, it could be blocking other tasks. The other tasks cannot directly stop the running task. So when the number of successful region requests reaches the value set for MAXRGN_REQUEST, the current task interrupts its work and triggers the search for another executable task within the UKT.

Small values result in more task changes. If not many users are working, this can result in unnecessary task changes. The overall cost of task switching rises.

Larger values mean longer task runtimes and fewer task changes. But blockage by "successful" tasks can occur.

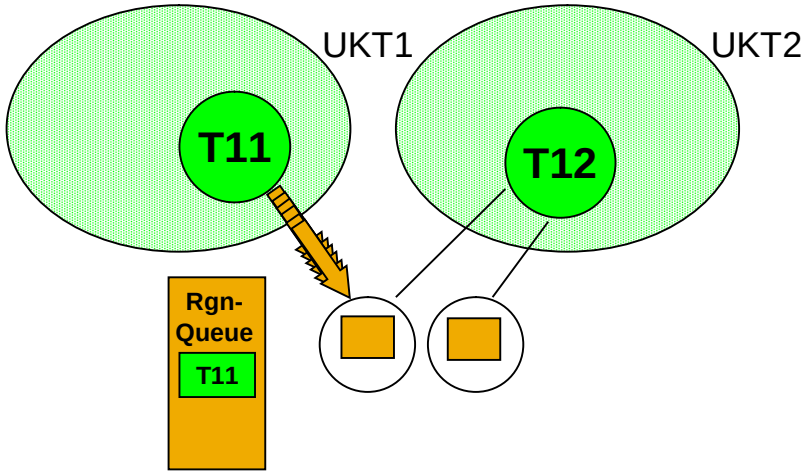
This parameter is especially important for single-processor systems.

Values
Default: MAXCPU = 1: 300
MAXCPU > 1: 3000
Min:100, Max: 100000
Online change: Yes

Busy Looping

XP_MP_RGN_LOOP

- Maximum number of attempts made to obtain a region used by another task (collision). If all attempts fail, the task is added to a specific queue.



Following a collision at a region, the task, being "executable", gets into the corresponding dispatcher queue in order to let another task become active. After it has been unsuccessful MP_RGN_LOOP times, the task enters into a special queue for that region in order to receive preferential access to it.

Values:

Default: -1

With the default value -1, at the start the database kernel calculates:

0 if MAXCPU = 1
100 if MAXCPU = 2-7
10000 if MAXCPU > 7 (starting in Version 7.6.02)

MP_RGN_LOOP should not be > 0 , if MAXCPU = 1 .

Online change: Yes

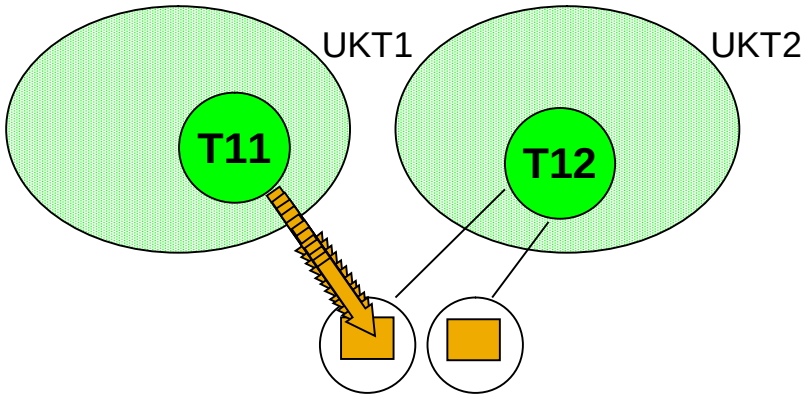
The optimal setting depends on the number of processors and their speed. The optimal value also depends on the speed of the operating system for IPC actions (semaphore, mutex) and thread changes.

In version 7.5, parameter XP_MP_RGN_LOOP serves to set the parameter MP_RGN_LOOP, which cannot be set manually.

Busy Looping Without Limitation

_MP_RGN_QUEUE

- In case of collisions busy waiting takes place trying to get a region without putting the task to a special queue.



Tasks are not put in a region queue when collisions occur if _MP_RGN_QUEUE is set to NO.

This can be a good idea if the database server has numerous fast CPUs but synchronization operations (semaphore, mutex) are very costly.

If a task fails to get the region over a lengthy period of time, it keeps a CPU busy with the thread and prevents that CPU and thread from being used for other operations.

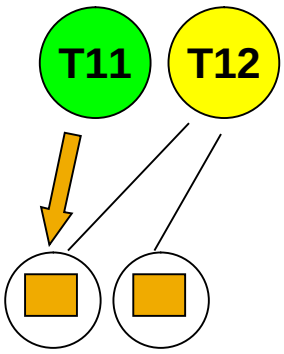
Values:

- Default: YES
- Online change: No

Dirty Read on Region Lock Management

_MP_RGN_DIRTY_READ

- Specifies, if during the attempt to get a region a loop with „dirty read“ on region lock management is passed through.
- Values:
  - 'YES': collision test with „dirty read“
  - 'NO' : "regular" collision test with lock in region lock management (Initial)
  - Default calculation:
    'YES' if MAXCPU > 1
    'NO' if MAXCPU = 1



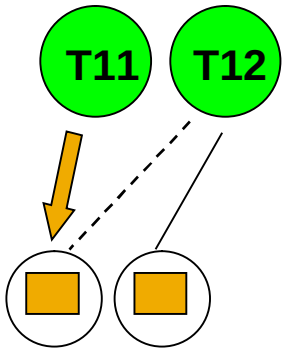
Helps to prevent collisions in region lock management.

Online change: Yes

Busy Waiting for Released Regions

_MP_RGN_BUSY_WAIT

- This parameter specifies whether busy waiting is to be performed after a failed test and set operation when exiting from a critical region.
- A task is not able to release a region, when another task is trying to obtain this region at the same time.
- Werte:
  - 'YES': Busy Waiting
  - 'NO' : Task becomes inactive (reschedule).
  - Default calculation:
    - 'YES' if MAXCPU > 1
    - 'NO' if MAXCPU = 1



This case only occurs if the task cannot access the region lock management. As this is unlikely to be the case for long, busy waiting obviates the need for a task change.

If a task cannot release a region because another task is trying to get it, it should not be put back in the run queue. It makes another attempt to release the region.

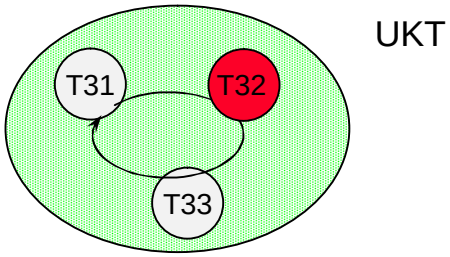
Online change: Yes

Dispatcher Loops

_MP_DISP_LOOPS

- Maximum number of dispatcher loops to search within a UKT for a runnable task.
- When this number is reached, the UKT voluntarily interrupts itself using the semop systemcall.

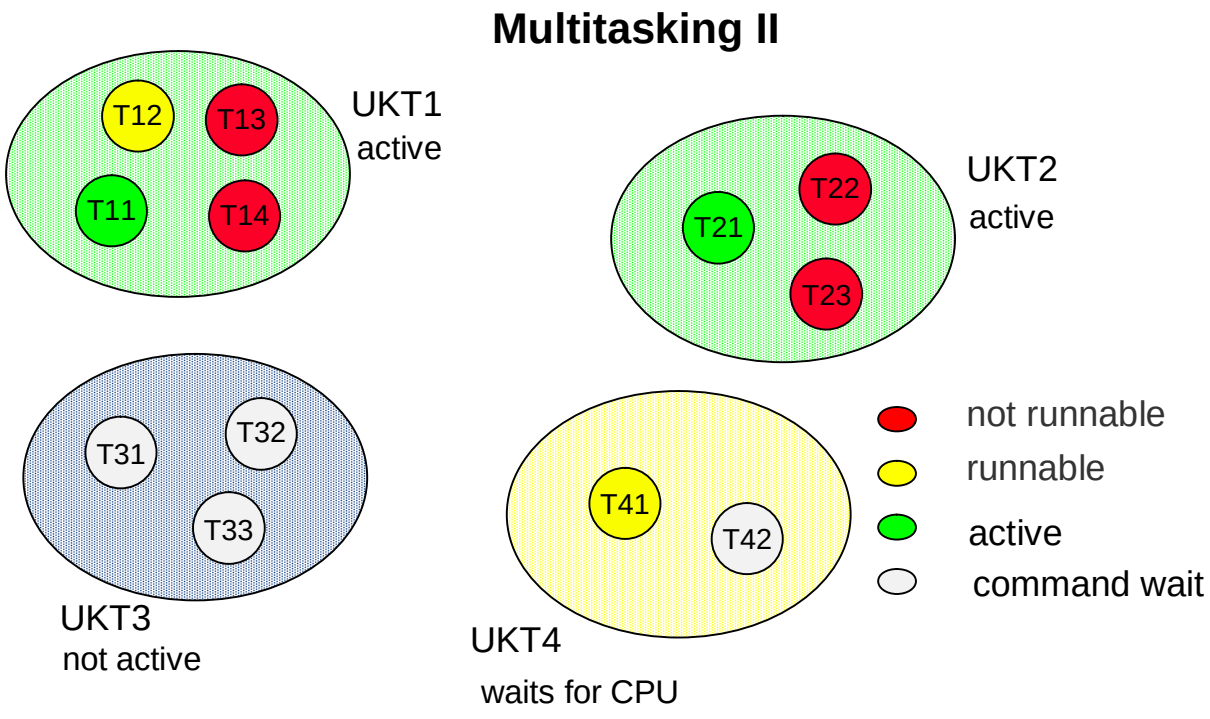
- Values:
  - Initial: 2
  - 2 – 10,000



The UKT only becomes active again when a task becomes executable.

Online change: Yes

Multitasking (II)



The threads UKT1 and UKT2 are running because they contain active tasks.

If task T11 stopped, T12 could become active, so UKT1 would remain active (as long as the operating system allows).

If task T21 stopped, UKT2 would no longer have an executable task and the thread would go to sleep.

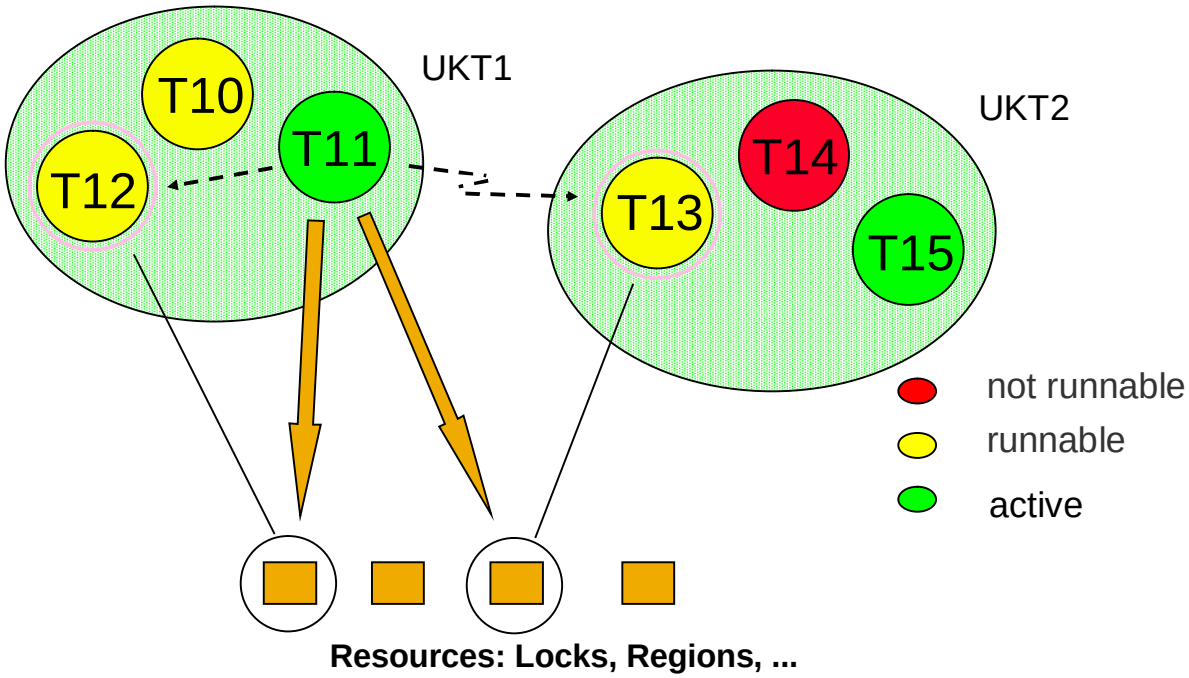
UKT3 is asleep. It is not awakened by the operating system because it is not executable as long as it has no executable task.

UKT4, if it were granted CPU time by the operating system, would take off and activate task T41.

Tasks T31, T32, T33 and T42 are in the "Command wait“ state.

Prioritization of Tasks

Which tasks are prioritized?



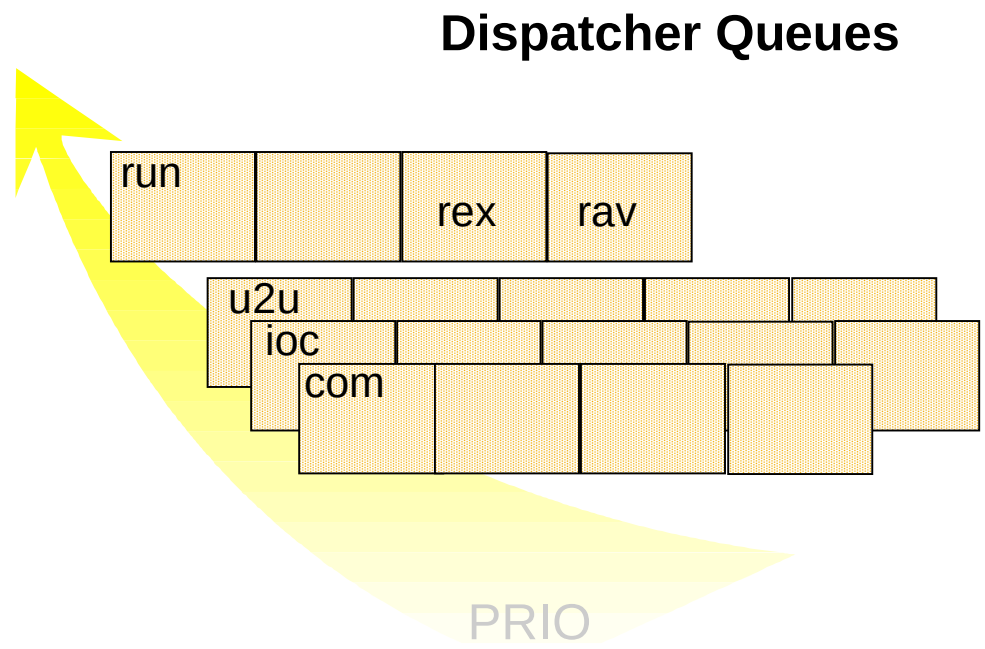
To ensure smooth operation of the database, you can prioritize tasks that contain resources needed by other tasks.

Exactly which tasks are prioritized and the manner of that prioritization can be set using parameters.

Measures to Favor Tasks

How are tasks favored?

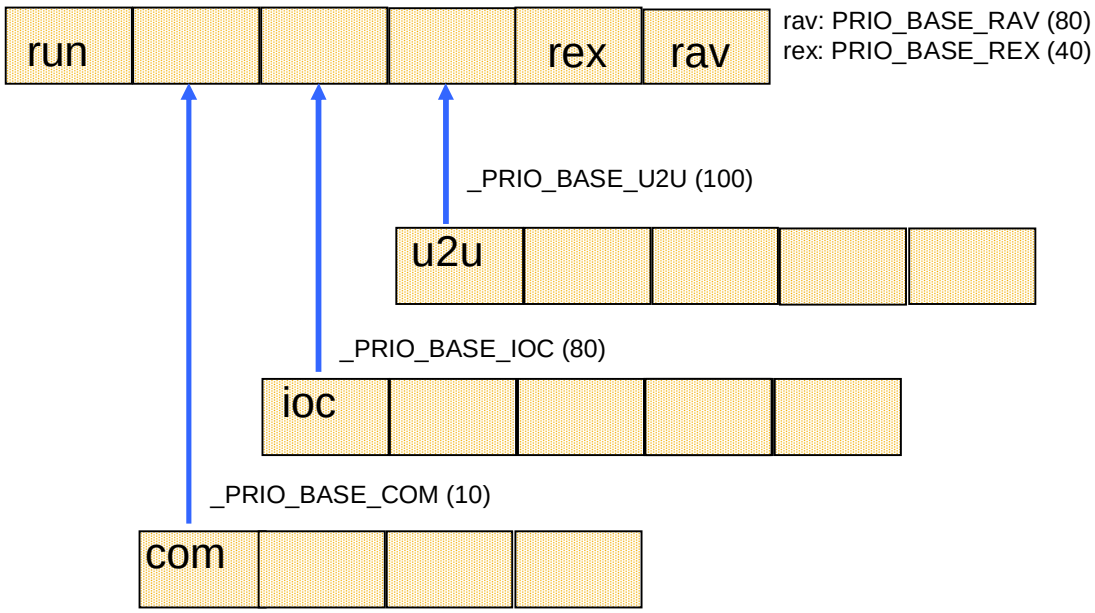
- special assignment of priorities
- interruption of running, not prioritized tasks
- provision of additional resources



Tasks are assigned to queues on the basis of their status. The priority of the task is set when it leaves the queue.

TASKS WARTEN AUF	BEZEICHNUNG IN X_CONS	QUEUE	PRIO
Kommunikation zwischen UKTs		u2u	u2u
I/O-Aufträge	I/O-Wait, Vvectorio, AsynWaitRead, AsynWaitWrite	ioc	ioc
SQL-Sperren oder kerninterne Sperren	vsuspend, vwait	run	rav
Kommunikation Anwendung-Kern	command wait, reply	com	com
"selbst-dispatched"	runnable, vsleep, Vbegexcl	run	rex

Base priority in the run queue



Executable tasks are assigned a base priority when they are added to the run queue or their status changes there.

This base priority depends on the previous state of the task.

The base priority can be set for the various prior states.

Base Priority Depending on the State

Base priority	Previous State	Default
_PRIO_BASE_COM	**Command Wait**	**10**
_PRIO_BASE_IOC	**I/O Wait** **80**	
_PRIO_BASE_U2U	**UKT to UKT communication**	**100**
_PRIO_BASE_REX	**self dispatched**	**40**
_PRIO_BASE_RAV	**waiting for lock (SQL, intern)**	**80**

Values: 0 – 32,000

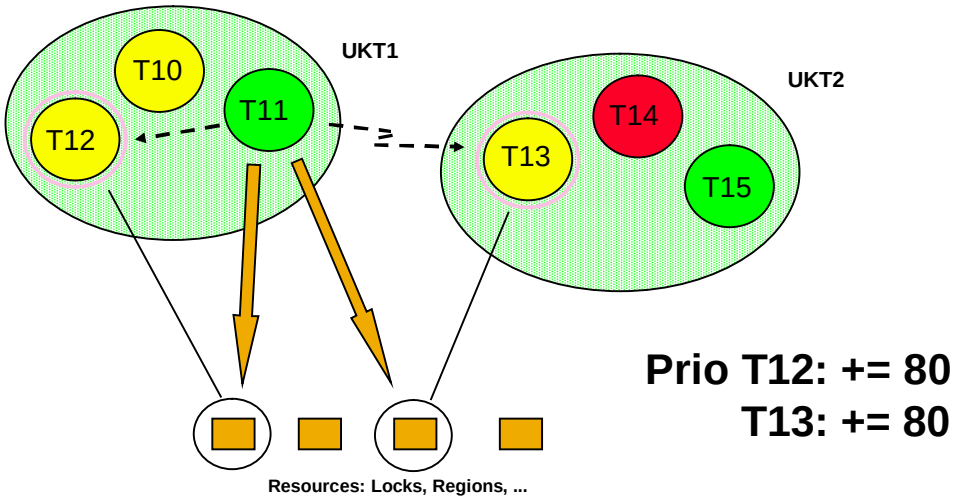
A task can withdraw itself if, for example, it exceeds _MP_RGN_LOOP.

Online change: Yes

Additional Priorities for Lockholder

_PRIO_FACTOR

- Increase the base priority of tasks holding a lock by the value of _PRIO_FACTOR, if other tasks are waiting for the lock.



A task that hold a lock can block several other tasks. These, in turn, can block yet other tasks. This can cause undesired waits.

To reduce wait times, tasks that hold locks for which other tasks are waiting are given higher priority.

If this parameter is set too high, a long-running job can run at too high a priority. Other tasks that are not working with the locked object would then receive insufficient CPU resources; i.e. the runtimes for short queries (such as single record access) would rise.

Values: Default: 80

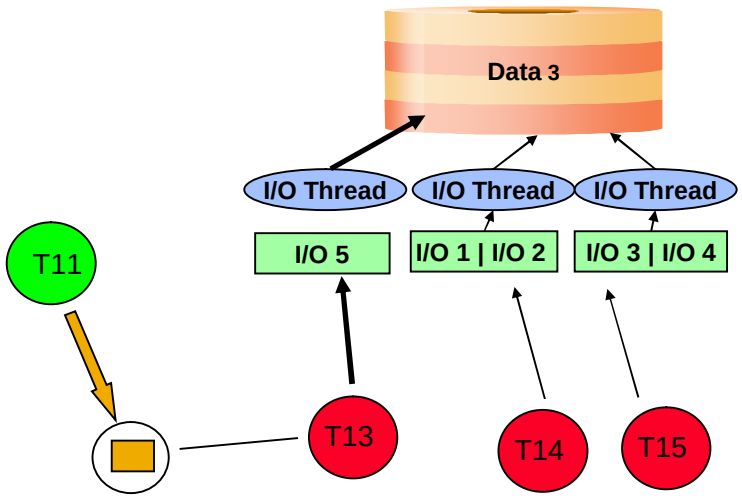
- 0 – 32,000

Online change: Yes

Additional I/O Threads for Prioritized Tasks

_IOPROCS_FOR_PRIO

- Number of addional I/O threads per system and data volume, which can only be used by prioritized tasks.



These threads are made available in addition to the fixed number of volume threads defined in the parameter _IOPROCS_PER_DEV.

The I/O requests of the prioritized tasks are not put into an I/O queue. This allows the I/O request to be processed more quickly.

The parameter can be set to > 0 if wait situations are often caused by tasks that hold locks and wait for I/O when the I/O system is under a heavy load. A better course of action would be to improve the performance of the I/O system.

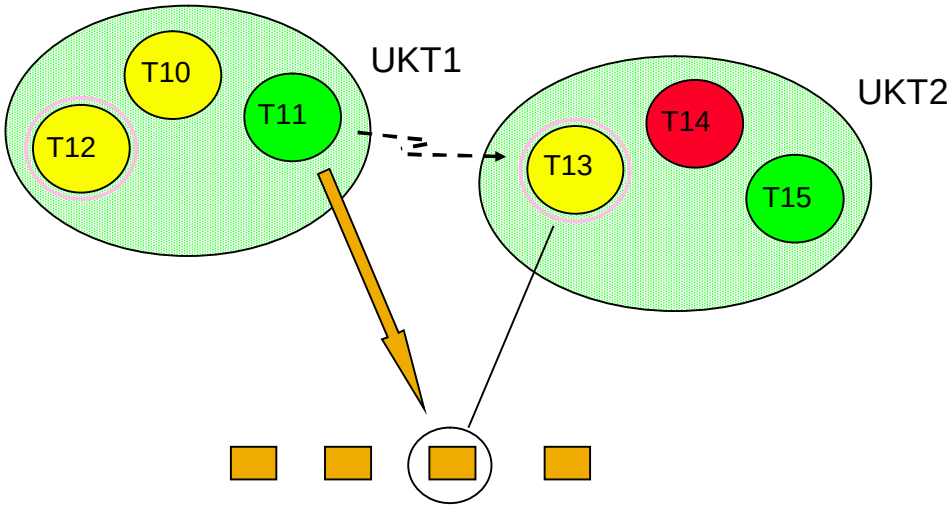
Values: Default: 0

- Min: 1, Max: 10
- Online change: No

Interruption of Tasks

_MP_DISP_PRIO

- Value ‘YES‘:
- If an inactive task gets a higher priority from a task of another UKT, the UKT of the inactive task stops its active task to activate the priority task.



In the illustration, task T11 runs into locks held by T13. That causes T13 to be given higher priority (if _PRIO_FACTOR is set).

UKT2 would then have to interrupt T15 and activate T13, if _MP_DISP_PRIO is set to YES. Otherwise T15 continues to run and T13 gets put into a prioritized queue.

Values: Default 'YES' if MAXCPU > 1
'NO' if MAXCPU = 1

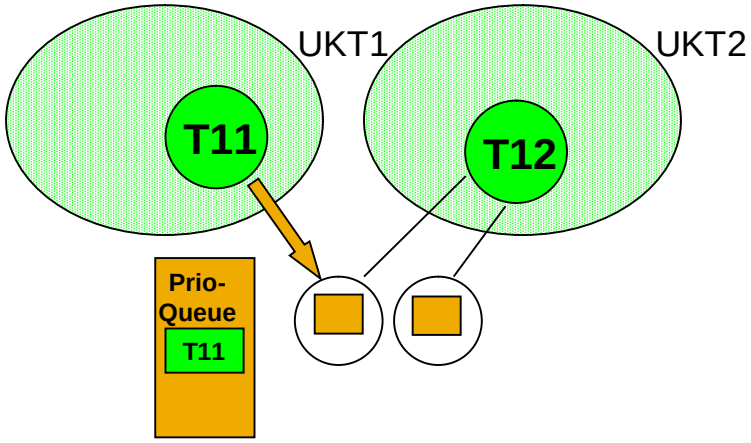
Online change: Yes

Favor Prioritized Tasks

_MP_RGN_PRIO

- Wert ‚YES‘:

A priority task is favored during the attempt to get a region.



If a prioritized task attempts to get a region that is being used by another task (collision), it immediately joins a special queue for the respective region rather than first having to make _MP_RGN_LOOP unsuccessful attempts (see also _XP_MP_RGN_LOOP).

Values: Default 'YES' if MAXCPU > 1
'NO' if MAXCPU = 1

- Online change: Yes

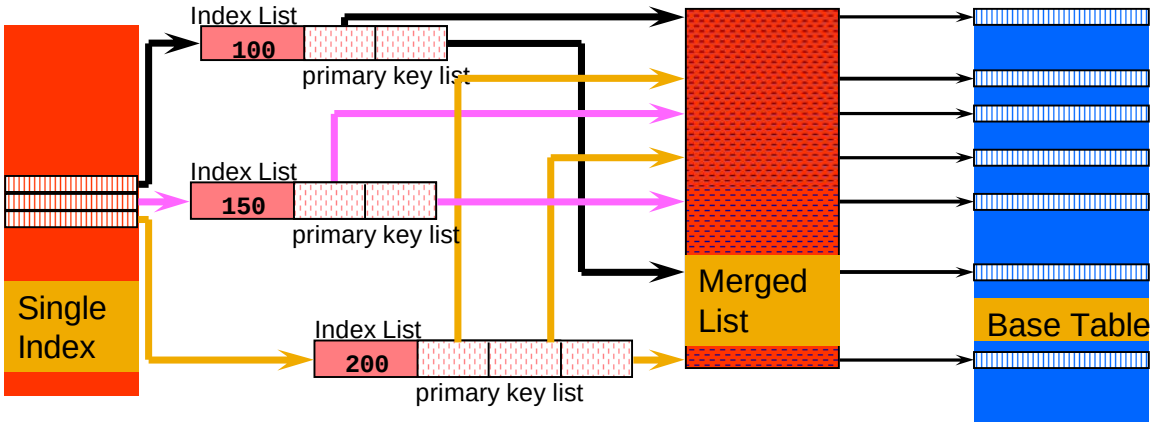
Optimizer

OPTIM_MAX_MERGE	Max. number of index pages for the creation of a merge list
OPTIM_INV_ONLY	Index Only Strategy
OPTIMIZE_MIN_MAX	Min and max optimization
OPTIMIZE_AGGREGATION	Aggregation through BD layer
OPTIMIZE_FETCH_REVERSE	Allow reverse index scan
OPTIM_CACHE	Determination of a strategy per statement
HASHED_RESULTSET	Create aggregates via hash tables
JOIN_SEARCH_LEVEL	Join sequence algorithms dependent of the
JOIN_MAXTAB_LEVEL4	Number of join tables
JOIN_MAXTAB_LEVEL9	
OPTIMIZE_OPERATOR_JOIN	Join processing without temporary result sets
OPTIMIZE_JOIN_PARALLEL_SERVERS	Parallel I/O for index transitions
OPTIMIZE_JOIN_HASHTABLE MAX_HASHTABLE_MEMORY	Use of hash joins
UPDATESTAT_PARALLEL_SERVERS	Number of server tasks for Update Statistics
UPDATESTAT_SAMPLE_ALGO	Sampling algorithm for determining statistics

Creation of Merge Lists

OPTIM_MAX_MERGE

- The parameter affects the decision, if an index strategy with a “merged list” is taken.
- If the number of pages of an index that need to be merged exceeds the value specified in OPTIM_MAX_MERGE, this index will not be used for an index merging strategy.



When employing an index strategy with a "merged list", the system does not immediately access the base table with each primary key value it finds in the index list; rather, it first generates a list of all the primary key values found and sorts them in the order of the primary keys.

Now the system can access the base table in the order of the primary keys. The time saved by this process has to be set against the cost of generating the "merged list."

Values:

Default: 500

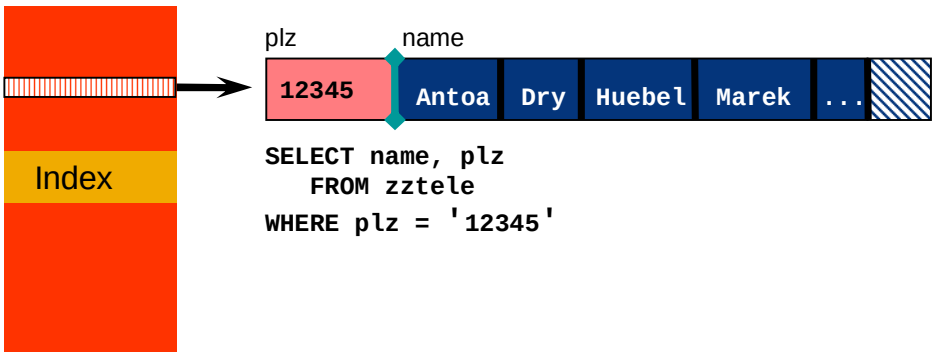
1 <= OPTIM_MAX_MERGE

Online change: Yes

Index Only Strategy

OPTIM_INV_ONLY

- With setting this parameter the database kernel allows to read all necessary data from an index, if possible. Access to the table tree is saved.



If the parameter OPTIM_INV_ONLY is set to YES, the INDEX ONLY is employed.

Values:

Default: YES

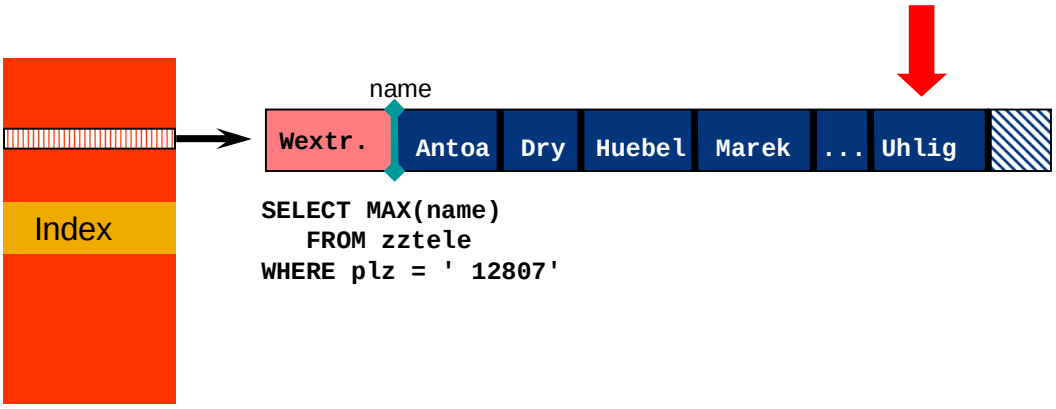
NO

Online change: Yes

Min and Max Optimization

OPTIMIZE_MIN_MAX

- If this parameter is set, the database determines a minimum and a maximum value for a column directly from an index or the primary key tree, if a suitable B* tree is available.
- The costs of access are unrelated to the number of records.



Values:

Default: YES

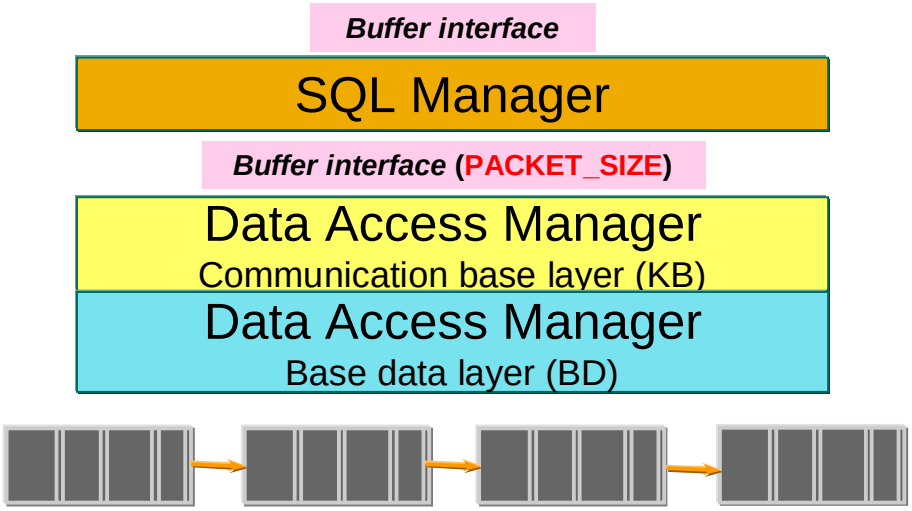
NO

Online change: Yes

Aggregate Optimization

OPTIMIZE_AGGREGATION

- With an Index Only Strategie, the BD layer caclulates the result of the aggregation. This makes communication between the kernel layers much more efficient.



Without aggregate optimization, the BD layer packs the determined individual individual values from the records into the request packet and thus transfers them to the SQL Manager. The SQL Manager then calculates the result.

With aggregate optimization, the BD layer directly calculates the result and returns it to the SQL Manager. This optimization significantly reduces the CPU load.

Values:

Default: YES

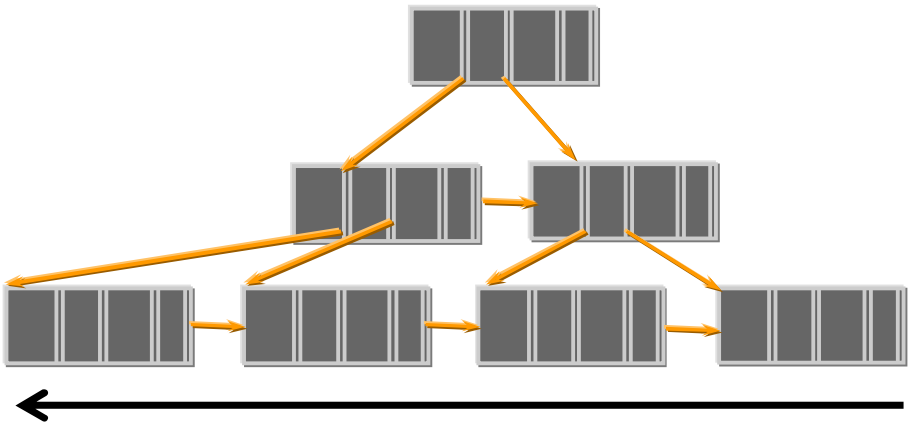
NO

Online change: Yes

Reverse Index Access

OPTIMIZE_FETCH_REVERSE

- If this parameter is set to YES, an index may be read backwards.
- Values:
  - YES (standard)
  - NO



Example:

```
explain
 select *
 from zztele
  where plz like '1%,
  order by plz desc
```

MaxDB allows you to backward scan an index. Backward scans place a greater load on the CPU than forward scans because the data pages do not have reverse chaining. To obtain access to the data in the index, the index pages of the tree must be read as well.

The database creates an internal sorted result table with OPTIMIZE_FETCH_REVERSE=NO. Creating an internal result table stresses the system more than performing a backward scan.

Values:

Default: YES

NO

Online change: Yes

Determination of a Strategy per Statement

OPTIM_CACHE

- As a rule an optimal search strategy is specified for each statement depending on data fulfilling the search condition. This behaviour can be switched off by setting OPTIM_CACHE.

SELECT * FROM <table>
WHERE status = ´not set´
➧ TABLE SCAN (affects a lot of rows)

SELECT * FROM <table>
WHERE status = ´set´ (affects just a few rows)
➧ RANGE CONDITION FOR INDEX COLUMN

If more than one search strategy is possible, it is a good idea to determine the best search strategy for each statement. This behavior can be switched off with the value YES for the parameter OPTIM_CACHE.

Values:

Default: NO: The optimal search strategy is determined for each statement.

YES: The optimal strategy is determined just once when the statement is parsed.

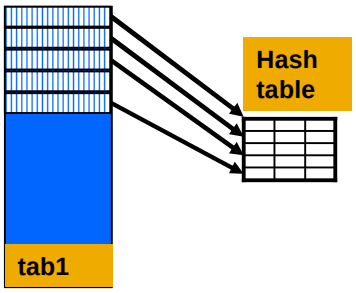
Online change: Yes

Use of Hash Joins

HASHED_RESULTSET

- Use of temporary hash tables for aggregates
- Values:
  - NO: Create aggregates via temporary B*Trees
  - YES: Create aggregates via temporary hash tables
  - Standard: NO

- SELECT min(available),
  max(available),
  avg(available)
  FROM stock
  GROUP BY year



Using hash tables to calculate aggregates is faster than creating a temporary B*Tree.

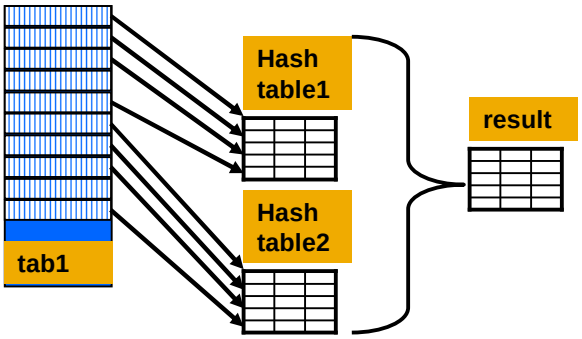
Online change: Yes

Maximum Cache Size per Hash Table

HASHED_RESULTSET_CACHESIZE

- Maximum of used cache per temporary hash table in KB.
- Values:
  - 32768 (32 MB) – 8388608 (8 GB)
  - Default: 262144 (256 MB)

- SELECT min(available),
  max(available),
  avg(available)
  FROM stock
  GROUP BY month



For performance reasons temporary hash tables need to remain in the main memory. The parameter HASHED_RESULTSET_CACHESIZE limits the size of those hash tables.

MaxDB stores the data of one hash table into a sorted page chain in the data cache if the hash table reaches the limit. It uses a new hash table in memory to continue with the select.

The last hash table and the data of the swapped out to page chains are joined into one result after all relevant records have been read.

Online change: Yes

Join Algorithms

JOIN_SEARCH_LEVEL

- Specifies the algorithm for the join sequence search to find the best order for table processing.
- Values:
  - 0: The parameters JOIN_MAXTAB_LEVEL4 & JOIN_MAXTAB_LEVEL9 determine the algorithm.
  - 2: Greedy Algorithm
  - 4: Internal Algorithm
  - 9: Perform Permutation
- Further algorithms are planned.
- Default: 0

Join algorithm

Algorithm	Performance	Precision
Permutation	Slow	High
Internal	Middle	Middle
Greedy	Fast	Low

The join optimizer should always determine the optimal sequence of tables. The best sequence depends on the size of the tables and the intermediate result set.

The best strategy is determined when all the possible sequences have been evaluated (permutation). This only makes sense with a small number of tables as permutation is costly. With five tables, 12 combinations have to be evaluated; with six tables, 720 combinations. The number of combinations is calculated according to n!, where n is the number of join tables.

An algorithm developed by the MaxDB team works considerably faster than permutation, but delivers somewhat less precise results.

If there are very many join tables, even the internal algorithm is too slow.

The parameter JOIN_SEARCH_LEVEL specifies which algorithm is always to be used independent of the number of join tables. If you want to use different algorithms depending on the number of join tables, set the parameter JOIN_SEARCH_LEVEL to 0.

Online change: Yes

Join Algorithm Depending on Number of Tables

JOIN_MAXTAB_LEVEL4, JOIN_MAXTAB_LEVEL9

- Specifies which algorithm for the join sequence search is used dependent of the number of join tables.

- If (# tables < JOIN_MAXTAB_LEVEL9)
  then perform permutation
- If (# tables > JOIN_MAXTAB_LEVEL4)
  then perform Greedy Algorithm
- If (JOIN_MAXTAB_LEVEL9 < # tables < JOIN_MAXTAB_LEVEL4)
  then perform Internal Algorithm
- JOIN_MAXTAB_LEVEL9: Default = 5, Values: 2-10
- JOIN_MAXTAB_LEVEL4: Default = 16, Values: 2-64

The standard settings are as follows:

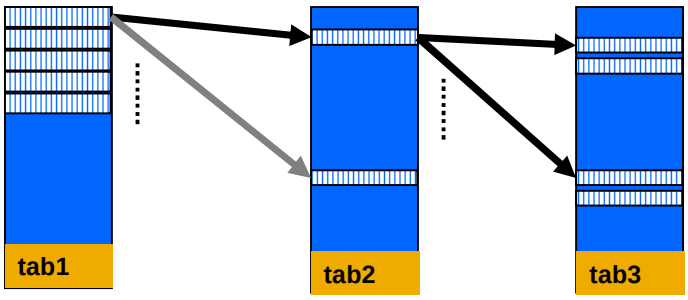
If the number of join tables

- is less than 5, a permutation is carried out,
- greater than 5 and less than 17, the internal algorithm is carried out,
- greater than 16, the greedy algorithm is carried out.
- Online change: Yes

Join Processing Without Temporary Result Sets

OPTIMIZE_OPERATOR_JOIN

- Allows join processing with „Nested Loop Join“ without creation of temporary result sets.
- Values:
  - YES: Nested Loop Join (Default)
  - NO: Sorted Merge Join



In contrast to sorted merge, with nested loop join a temporary result set is not generated for each join transition. For each record in the first table, the join transition to the next table is processed, and so on.

Particularly when aggregate functions are being used in the selection list and there are few records in the result set, nested loop join works with considerably smaller result sets and thus faster.

The parameter **OPTIMIZE_OPERATOR_JOIN_COSTFUNC** determines whether an additional CPU-conserving algorith (improved) should be used.

If the parameter **OPTIMIZE_JOIN_OPERATOR_SORT** is set and there is no primary key or index for the join transition, then the database first generates a sorted temporary result set of the relevant columns. The join is then executed with the sorted data in the temporary result set. This can lead to major savings of join steps.

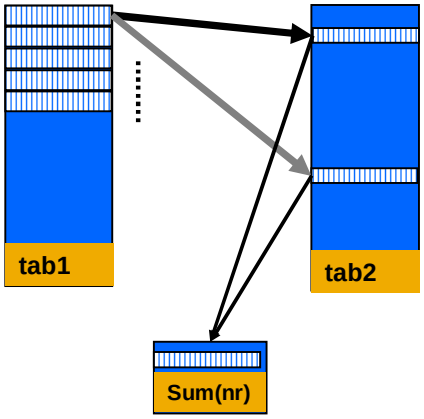
The additional parameter **JOIN_TABLEBUFFER** defines the size of the buffer per user for storage of the table data. 128 KB is the default setting.

Online change: Yes
A change to parameter JOIN_TABLEBUFFER only goes into effect after restarting the database.

Join Aggregate Without Temporary Result Sets

OPTIMIZE_JOIN_ONEPHASE

- Creation of the aggregate directly at the join. The complete join result is not generated in order to then compose the aggregate.




```
SELECT SUM(nr)
 FROM zztele t, zzstadtteil s
WHERE t.plz = s.plz
  AND s.stadtteil = 'dummy'
```


In the previous procedure, first the join result was generated. Then the aggregate was composed out of the result. Processing was multi-phase.

Now MaxDB can create a join during execution of the join. This leads to reduced memory requirements - and thus better performane - since the result set of the joins is not generated. Processing is done is one phase.

Values:

Default: YES: Single-phase processing

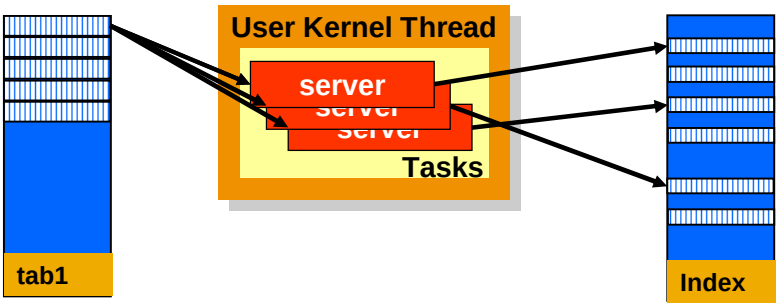
NO: Multi-phase processing

Online change: Yes

Parallel I/O for Index Transitions

OPTIMIZE_JOIN_PARALLEL_SERVERS

- Number of server tasks which are used for a join transition via index to read index blocks in parallel.
- Values:
  - 0: no parallel read of index blocks (Default)
  - >0 : Index blocks are read in parallel for join transitions



For join transitions that are to be processed via an index, importing index blocks in parallel can improve performance considerably.

Depending on the number of server tasks specified, the corresponding Selects can strain the I/O system, putting other users at a disadvantage. The value of OPTIMIZE_JOIN_PARALLEL_SERVERS should by lower than the number of configured data volumes.

The parameter **OPTIMIZE_JOIN_PARALLEL_MINSIZE** determines when the parallel read algorithm can become active based on the size of linked tables.

Online change: Yes

Determination of the Order of Table Processing

OPTIMIZE_JOIN_OUTER

- Specifies if the optimizer determines the order of tables for outer join processing.
- Values:
  - YES: The optimizer determines the order.
  - NO : The order is taken from the select.

```
Select  name, vorname, strace, plz, kfz
  from  zztele, zzstadtteil, zzkreis
 where  zztele.plz      = zzstadtteil.plz
   and  zztele.kfz      = zzkreis.kfz (+)
   and  zztele.name     = 'Schroeder'
```

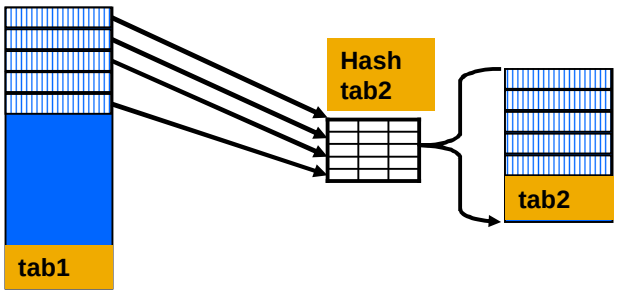

With outer joins, the result can vary depending on the processing sequence. For this reason, the parameter OPTIMIZE_JOIN_OUTER allows you to specify whether the database should choose the best sequence or if the sequence is to be taken from the Select statement.

Online change: Yes

Hash Joins

OPTIMIZE_JOIN_HASHTABLE

- Using hash joins
- Values:
  - YES: Hash joins used
  - NO: Hash joins not used
  - Standard: YES



Using the parameter OPTIMIZE_JOIN_HASHTABLE you can activate and deactivate the use of hash joins online.

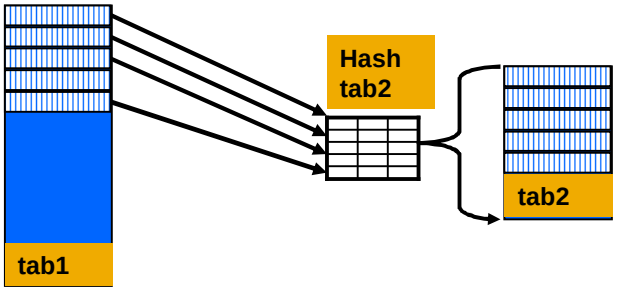
The parameter OPTIMIZE_JOIN_HASH_MINIMAL_RATIO defines the minimum ratio of output amount to the table for the next join step, after which hash joins are used. The deafult value is 1%.

Online change: Yes

Maximum Cache Size for Hash Joins

MAX_HASHTABLE_MEMORY

- Maximum size of memory for temporary hash tables to be used by all user sessions for join execution.
- Values:
  - 0: Hash joins are not used
  - >0: Size in KB
  - Standard: 512



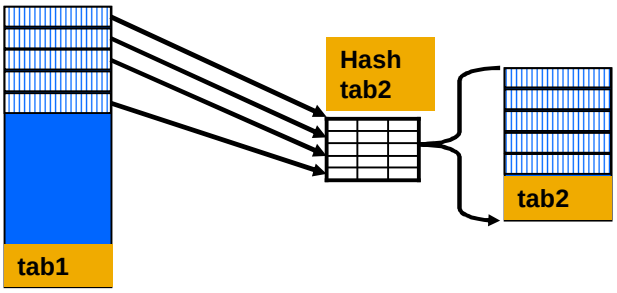
The parameter **MAX_SINGLE_HASHTABLE_SIZE** defines the maximum number of pages a table may have for a join transition. The standard setting sets a limit of 512 KB.

Online change: Yes

Maximum Table Size for Hash Joins

MAX_SINGLE_HASHTABLE_SIZE

- Maximum size of a join table which allows building of a temporary hash table.
- Values:
  - 0: Hash joins are not used
  - >0: Size in KB
  - Standard: 512



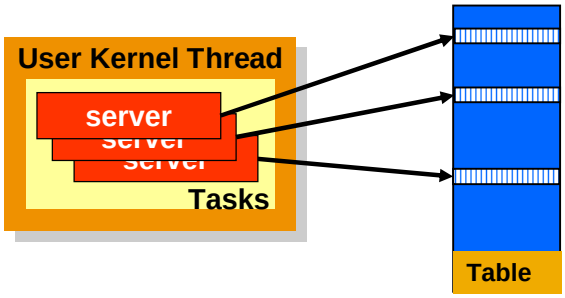
It is only sensible to generate temporary hash tables for join tables below a certain size. With large tables, you lose more time with the scan than you gain through the better search algorithm on the temporary hash table.

Online change: Yes

Number of Server Tasks for Update Statistics

UPDATESTAT_PARALLEL_SERVERS

- Number of server tasks executing Update Statistics simultaneously. Update Statistics is started automatically by the kernel or manually. dbmcli commands:
  - sql_updatestat
  - sql_updatestat_per_systemtable
  - auto_update_statistics
- Values:
  - 0: Number of server tasks is equivalent to number of data volumes
  - >0: Manually defined number of server tasks



As of version 7.6, parallelization for Update Statistics also occurs for single tables. This reduces the runtime for collecting data per table.

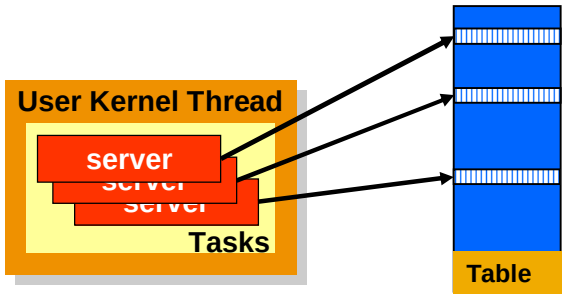
In the standard, the Update Statistics run uses as many server tasks as data volumes are defined. If the system load caused by the parallel I/O is too high, you can use the parameter UPDATESTAT_PARALLEL_SERVERS to reduce the number of parallel I/O for Update Statistics.

Online change: Yes

Algorithm for Update Statistics

UPDATESTAT_SAMPLE_ALGO

- Choice between different algorithms for the sampling procedure used in generating statistics.
- Values:
  - 0: Traditional algorithm
  - 1: Advanced algorithm (default)
  - 2: Advanced algorithm with tendency to overestimate the number of different values in a column
  - 3: Traditional algorithm with additional sampling improvements
  - 4: Combination of 1 and 2



Because of previously unstatisfactory results for determining statistics using the sampimg method, new algorithms were introduced in 7.5 and 7.6. As of version 7.6, the database default is the new algorithm 1.

Results to this point show that importing 5% of table data is sufficient with the new algorithm. With the traditional algorithm, at least 10% were necessary. Often it was necessary to read much more than 10% of the table data to get reliable statistics.

Online change: Yes

In Version 7.5, the new algorithms are available as of 7.5.00 Build 34.

Thank you!