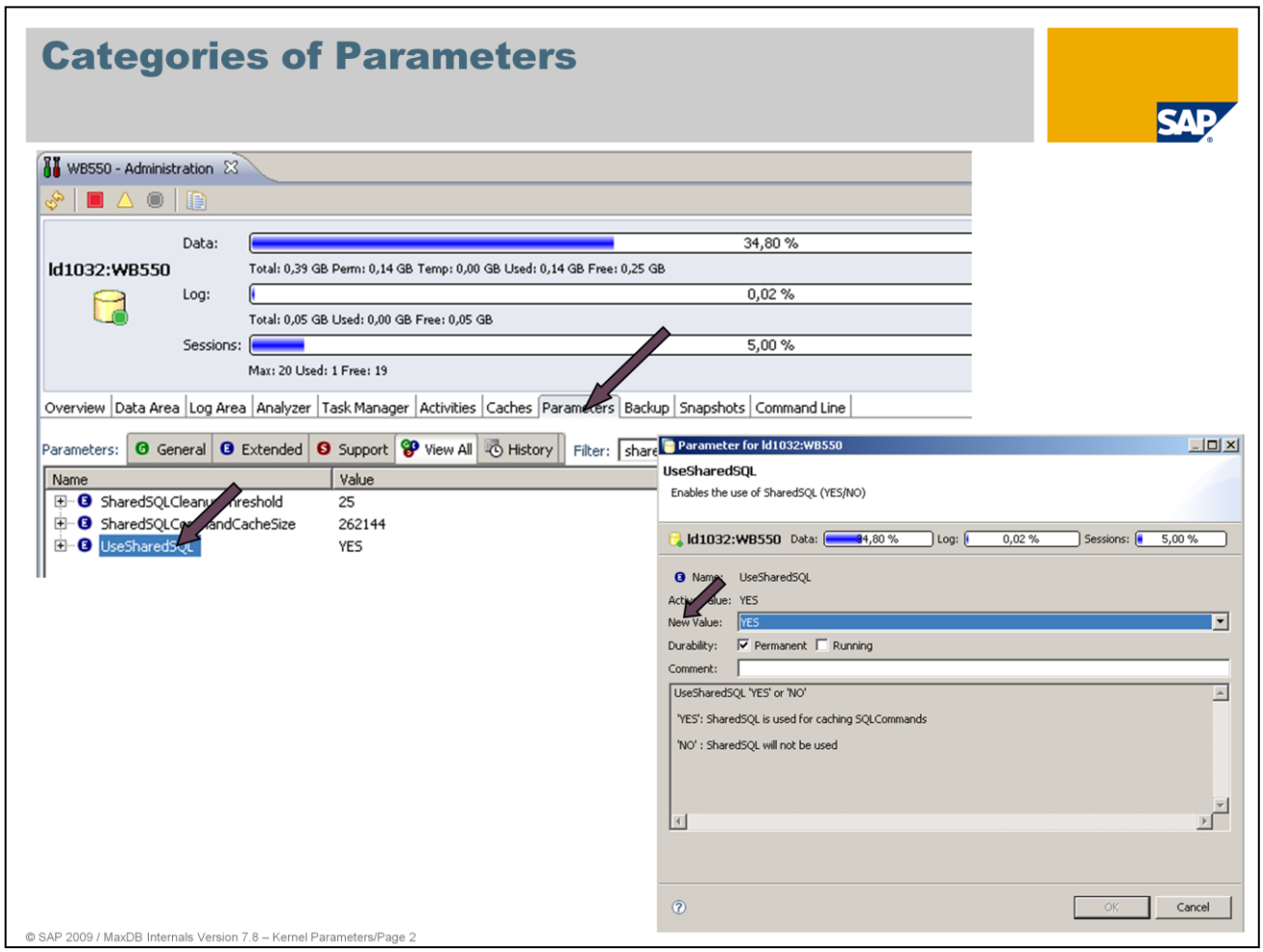
SAP® MaxDB™
Kernel Parameters
Version 7.8

Heike Gursch
Werner Thesing

THE BEST-RUN BUSINESSES RUN SAP™



Kernel parameters are divided into three classes:

- **General**

  These parameters are set by database administrators.

- **Extended**

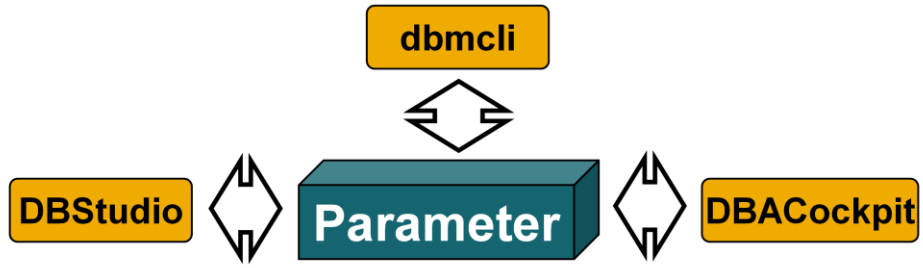
  These parameters are set in consultation with MaxDB Support or by implementing notes from the database administrator.

- **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_getvalue CacheMemorySize
- Assign value: param_put CacheMemorySize 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
- recover_config — restores parameter settings from a data backup

The dbmcli command “help param” displays more commands.

When do parameters become active?

Many parameters can be changed online:

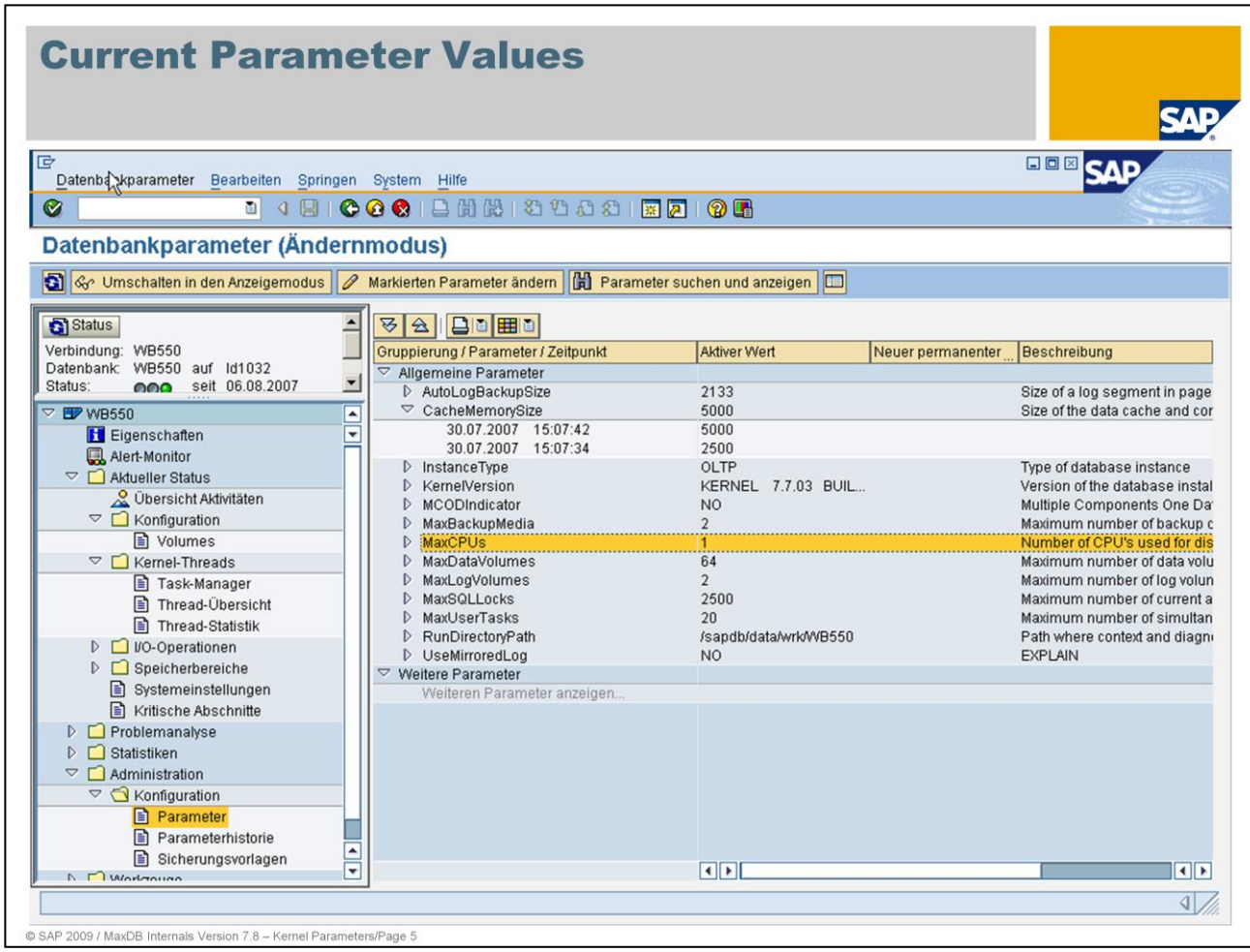
- dbmcli –d MYDB –u control,pass
  - param_getfull EnableQueryRewrite...
    CHANGE RUNNING
    ...
  - param_put –running –permanent EnableQueryRewrite YES
- The current setting of the parameters is shown by the view ACTIVECONFIGURATION.

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 example the execution of Query Rewrite is activated.

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

As of version 7.7.03 the parameter names were consolidated. Therewith most parameters got a new name without containing underlines. The legibility of parameter names is improved by the use of upper and lower case characters. You can read and set the parameters by using the old names. The command param_directgetall only shows the new parameter names. The view ACTIVECONFIGURATION shows old and new parameter names.

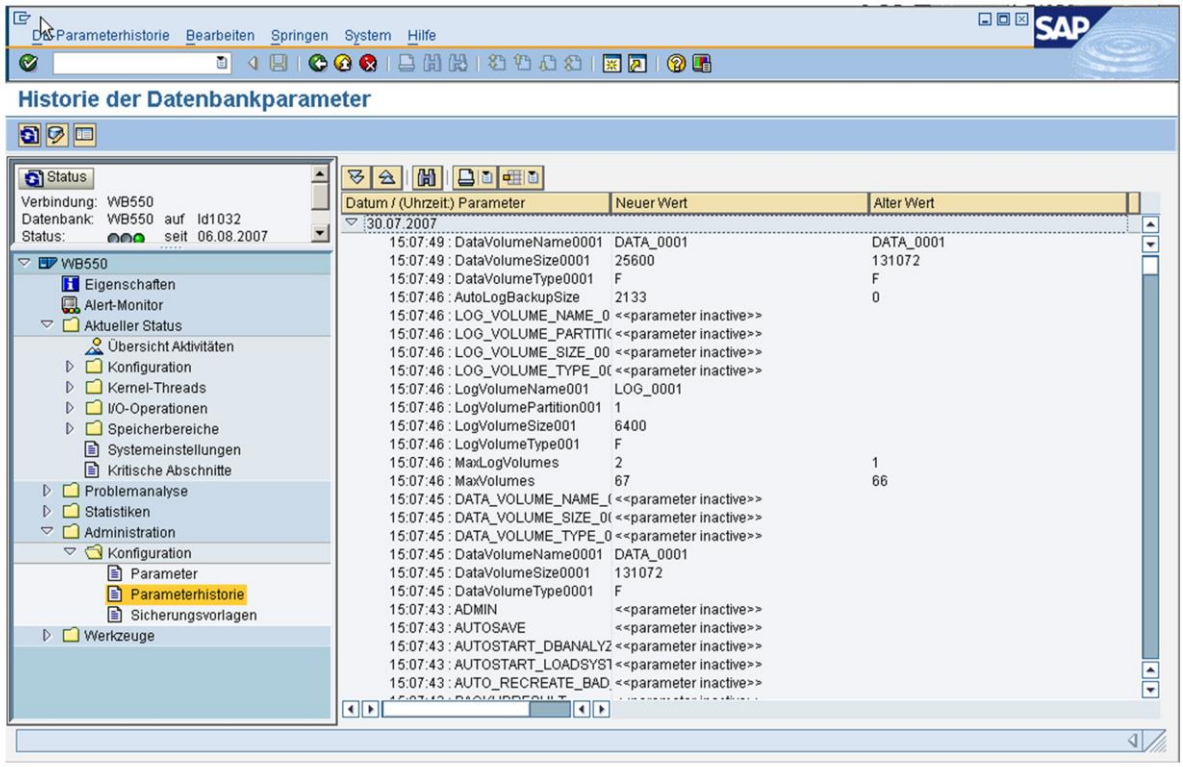


The parameter file is in the directory < PrivateDataPath>/config.

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 and DBACockpit .

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

DB50: Configuration and History of Parameters



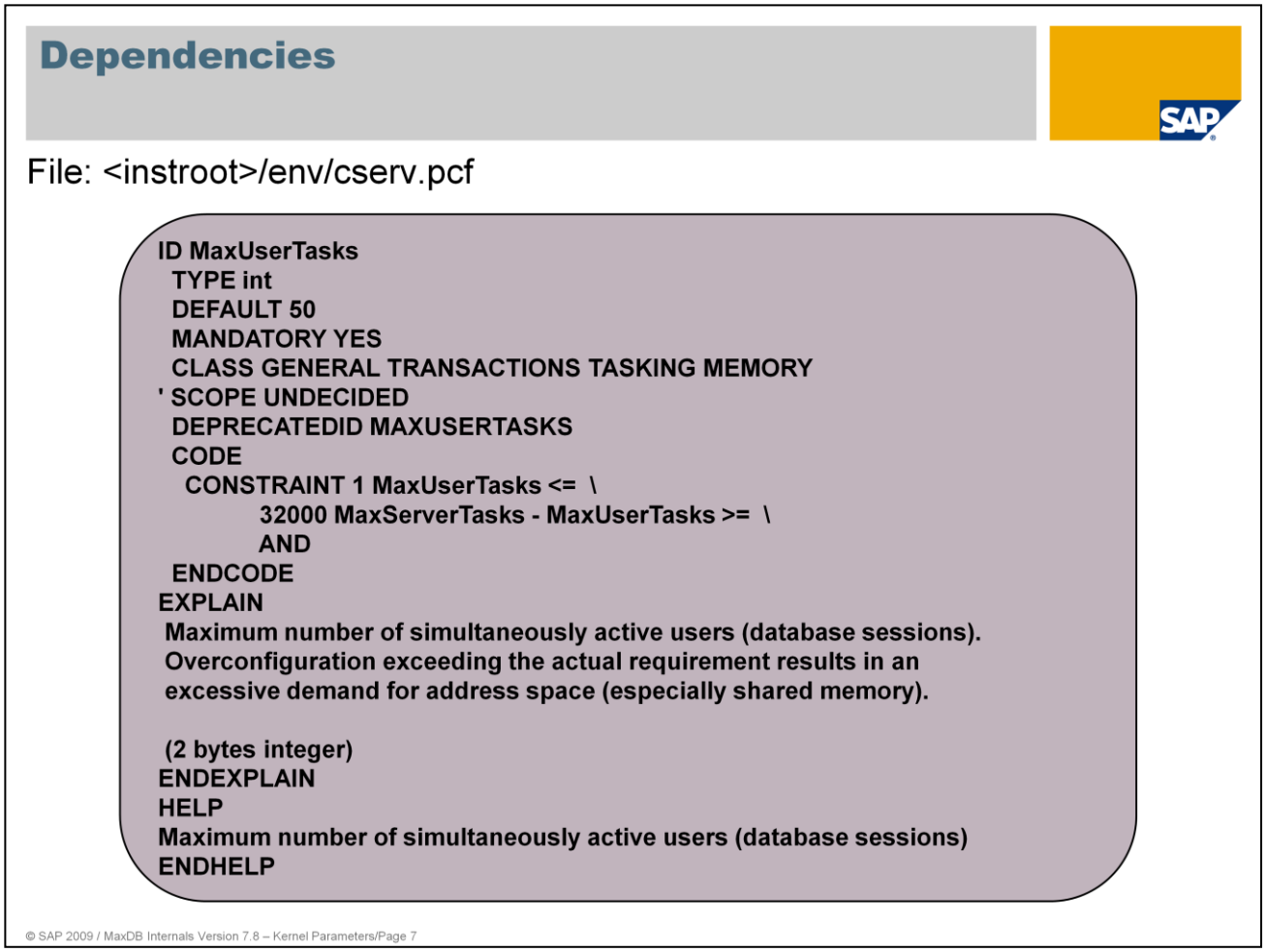
SAP note 1308217 provides recommendations for parameters in version 7.8

In transaction DB50 and DBACockpit 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 Database Studio.



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 not 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
CLASS	Classification	General, Extended, ...
DEPRECATED	Old parameter name	
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.

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

InstanceType	OLTP, LVC
DataVolumeName_<nnn>	Names of data volumes
LogVolumeName_<nnn>	Names of log volumes
UseMirroredLog	Log volume mirroring
MaxDataVolumes	Max. number of data volumes
MaxBackupMedia	Max. number of parallel backup devices
MaxUserTasks	Max. number of parallel user sessions
RunDirectoryPath	Default: /sapdb/data/wrk/<DBNAME>
KernelMessageFileSize	Size of file KnlMsg (in Rundirectory)
TraceBufferSize ..	Size of trace buffer for dedicated tasks
...	

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)

CacheMemorySize	Size of I/O buffer cache in 8 KB pages
DataCacheStripes	Segmentation of data cache
DataCacheSyncGranularity	Synchronization for page accesses
DataCachePinAreaThreshold	Cache size for cache pinning
TempResultsetPinAreaThreshold	Cache size for temporary page cache pinning
UseDataCacheScanOptimization	Scan optimization
MaxTempFilesPerIndexCreation	Parallel create index
ConverterStripes	Segmentation of converter cache
CAT_CACHE_SUPPLY	Total size of catalog cache
UseSharedSQL	Shared-SQL cache
SharedSQLCommandCacheSize	Size of Shared-SQL cache
SequenceCacheSize	Size of caches used for sequences

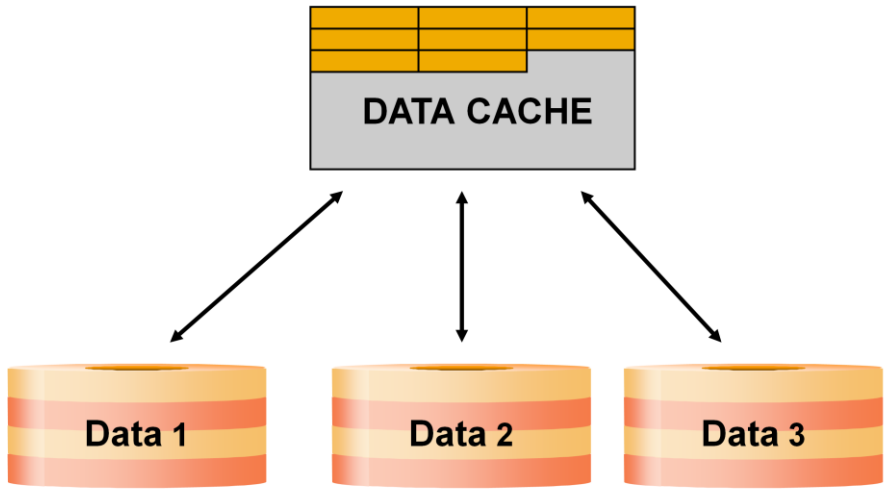
Caches and Other Memory Structures (2)

LogQueueSize	Size of LOG_IO_QUEUE in 8 KB pages
LogQueues	Number of log queues
MaxLogWriterTasks	Maximum number of log writer tasks
LogIOClusterSize	Max. no. of log pages per I/O
LocalRedoLogBufferSize	Transaction buffer for redo log entries
MaxSQLLocks	Maximum number of parallel line locks
InternalLockStructureEntries	Number of entries that are taken from free space management
DeadlockDetectionLevel	Deadlock detection

I/O Buffer Cache

CacheMemorySize (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: 10,000 pages
Min: 800, Max: 2,147,483,640
SAP system: > 30.000
Online change NO

The data cache hit rate can be determined as follows:

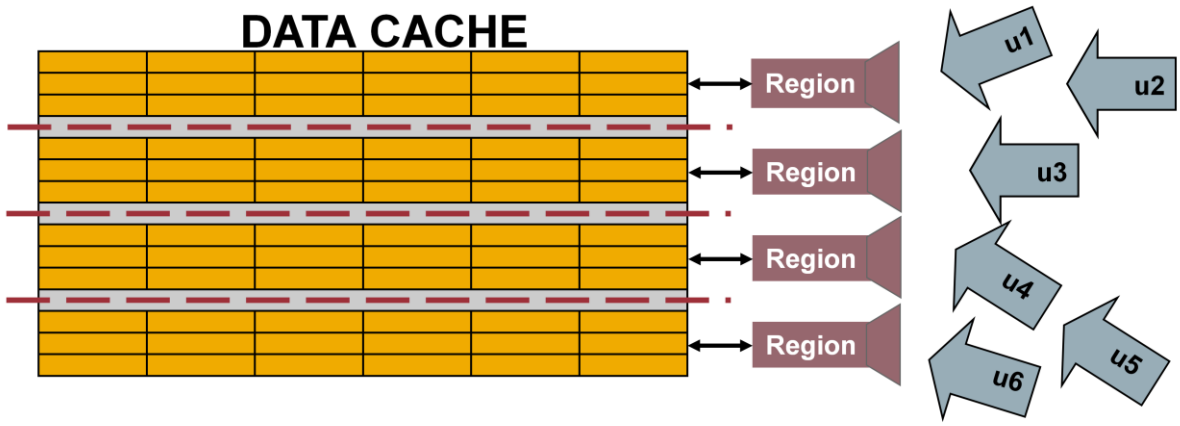
- SQL statements
  - SELECT * FROM MONITOR_CACHES
  - MONITOR INIT executes a reset
- Database Studio tabulator Caches in the administration window)
- DBAnalyzer
  - Displays the hit rate for a defined period of time.

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

DataCacheStripes (XP_DATA_CACHE_RGNS)

- Number of autonomous entities, the data cache will be divided in
- Each region has it's own LRU (least recently used) list
- The regions allow parallel accesses to the LRU lists



DataCacheStripes 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.

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: 1024
The number of regions is calculated depending on the size of the parameter CacheMemorySize.
Online Change: NO

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

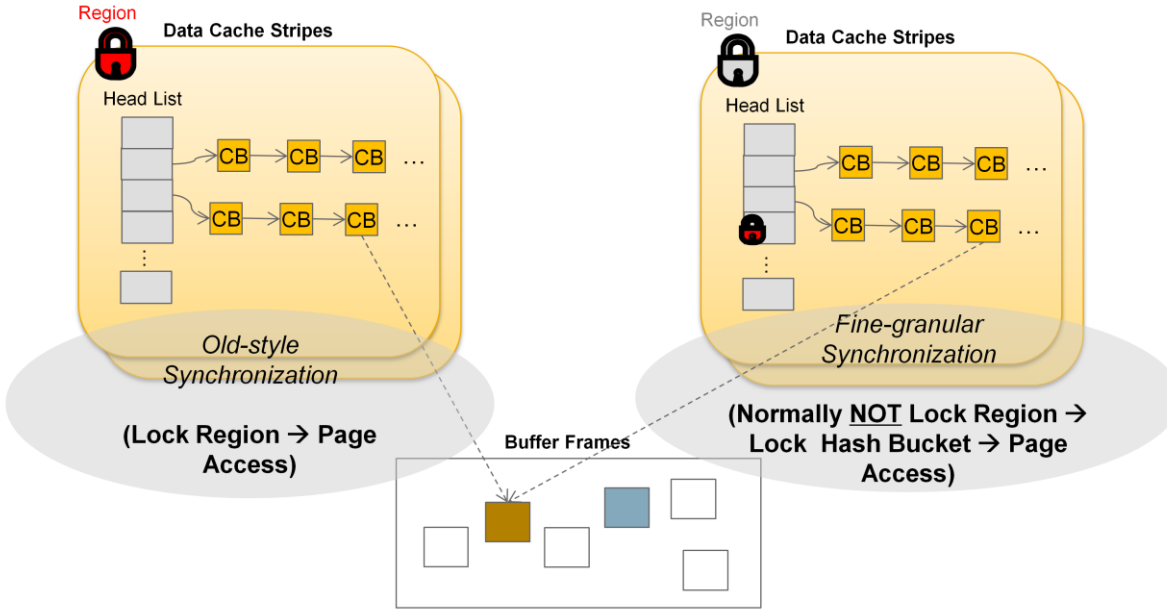
As of version 7.8 MaxDB uses the segments to synchronize accesses to the LRU lists. It now uses another synchronization method for accesses to pages in the cache.

If very small regions are defined this may lead to a bad hit ratio in the cache when one region is used more often because of frequently accessed pages than others.If very big regions are defined the search in hash lists of the LRU management may have a negative impact on the performance of the system.

Data Cache Synchronization

DataCacheSyncGranularity

- Number of lock hash buckets used for page access synchronization



Version 7.8 introduced the fine granular data cache synchronization. The parameter **UseFineGranularSynchronization** should have the value YES.

The region lock management doesn't distinguish between reader and writer locks. Reader tasks can block other reader tasks.

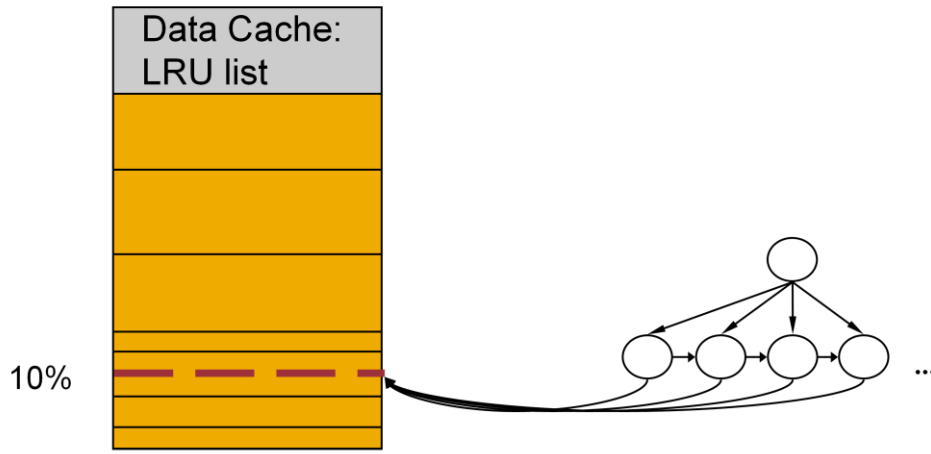
The new lock management needs less CPU resources. It distinguishes between reader and writer locks. Readers tasks can block writer tasks but not other reader tasks. This improves the scalability for multi user / multi CPU operations.

Values: Default: CacheMemorySize / 128
Min, Max: DataCacheStripes * 2 <= DataCacheSyncGranularity <= CacheMemorySize * 4
Online Change: No

Scan Optimization

UseDataCacheScanOptimization (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.

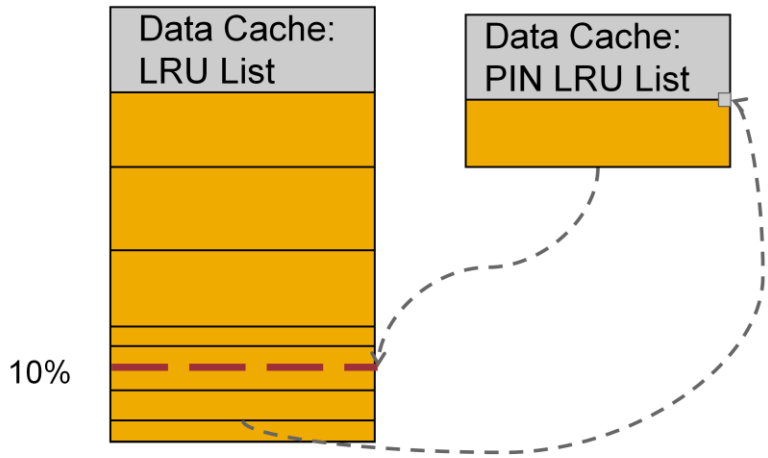
Values: Default: NO
YES: The whole data cache is used for scans.
NO: Only the last part of the data cache is used for scans.
Online change: YES

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

Cache Pinning

DataCachePinAreaThreshold

- Part of the data cache, used for tables with the CACHE attribute
- Syntax: CREATE TABLE <tablename> … CACHE
  ALTER TABLE <tablename> CACHE



As of version 7.8 MaxDB can hold pages of tables and indexes having the CACHE attribute in a LRU pin area of the data cache. The Parameter DataCachePinAreaThreshold defines the maximum size of the data cache in percent used as pin area.

Pages that have to be swapped out of the pin area will be inserted into the global LRU list at the beginning of the last 10%. The pages will be linked back into the beginning of the PIN LRU list when they are reused.

Values: Default: 5
Min: 0
Max:: 50
Online Change: YES

Use the table attribute NO CACHE if pages shell be removed from the cache after a short time. The cache management inserts the pages at the end of the LRU list when reading them into the cache. This configuration makes sense for logging tables which usually will not be read by the application after the data has been written.

Syntax:

```
CREATE TABLE <tablename> … NO CACHE
ALTER  TABLE <tablename> NO CACHE
:
```

Monitoring:

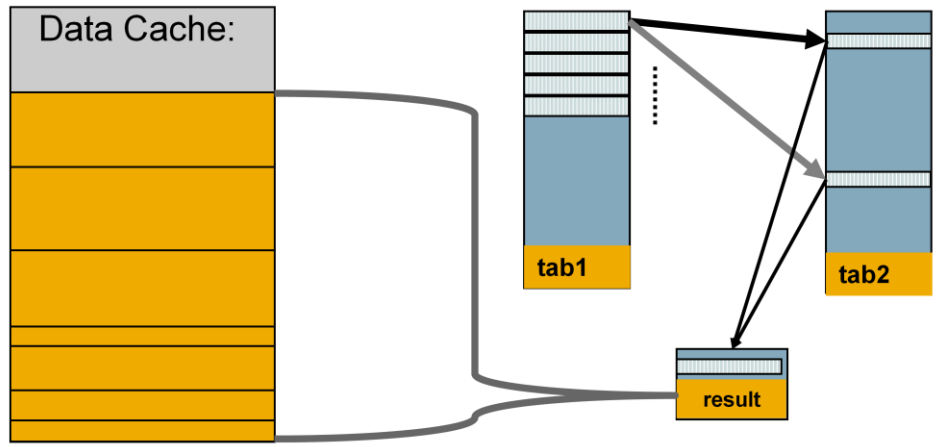
Hitrate of PIN-Area: :

```
select * from sysdba.monitor_caches where description like 'Data cache pinpage%'
```

Cache Pinning for temporary Pages

TempResultsetPinAreaThreshold

- Part of the data cache the SQL Manager can occupy in the data cache with the assurance those pages will not be swapped out
- The SQL Manager can access with direct memory pointers to the pages in the cache. This allows a fast creation of result sets.



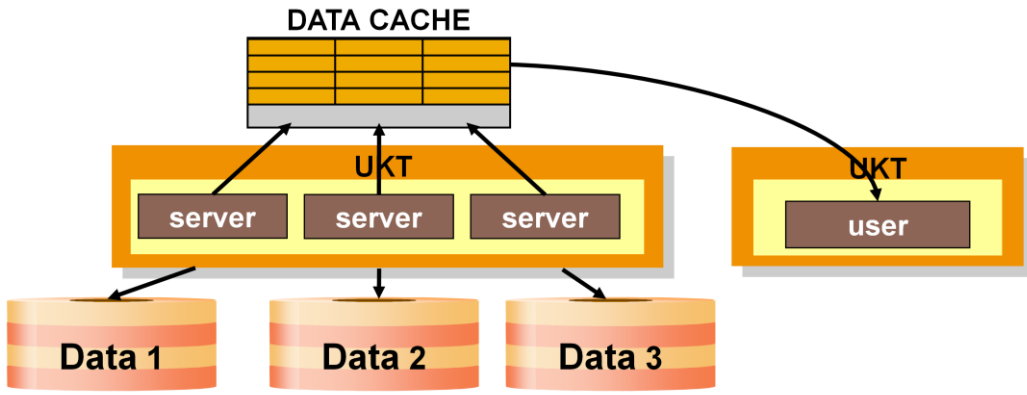
As of version 7.8 MaxDB the SQL Manager can occupy pages in the cache and access them via C++ - pointers. The pages are market as “in use” for a longer timeframe. Those pages will not be swapped out of the cache because they are in use. This allows a direct memory access by the SQL Manager to the cache pages. The creation and usage of result sets speeds up significantly.

Values: Default: 10
Min: 0
Max:: 50
Online Change: YES

Parallel Create Index

MaxTempFilesPerIndexCreation (_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.

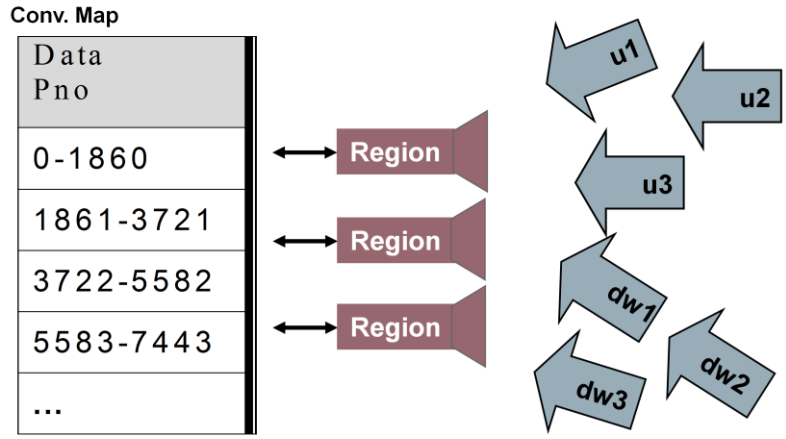
The value for MaxTempFilesPerIndexCreation should not exceed CacheMemorySize / 3.

Values: Default: Depends on Cache MemorySize if condition MaxTempFilesPerIndexCreation <= CacheMemorySize / 3 is fulfilled
Allowed: 0, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072
Online Change: YES

Converter Cache Regions

ConverterStripes (XP_CONVERTER_REGIONS)

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



The parameter ConverterStripes 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.

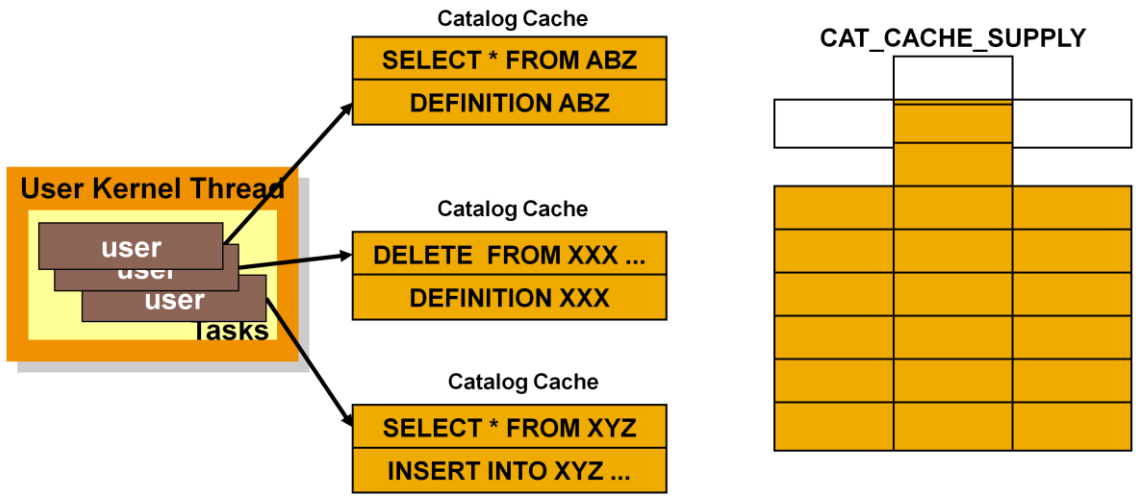
Values: Default: The value is calculated depending on the defined size of the database.
Min: 1, Max: 64
Online change: NO

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 TaskSpecificCatalogCacheMinSize (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 SESSIONS

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

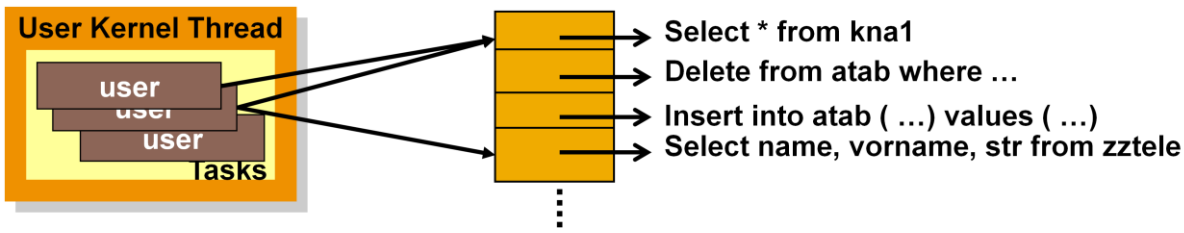
- If { (MaxUserTasks + 1) / TaskSpecificCatalogCacheMinSize * 8192 } > TaskSpecificCatalogCacheMinSize
  CAT_CACHE_SUPPLY = (MaxUserTasks + 1) / TaskSpecificCatalogCacheMinSize * 8192
- Restriction:
  _TaskSpecificCatalogCacheMinSize * 8192 / MaxUserTasks + 1 <= CAT_CACHE_SUPPLY
- No online change

The name of this parameter has not been changed in version 7.7.03 as the meaning will change soon.

Shared-SQL Cache

UseSharedSQL (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

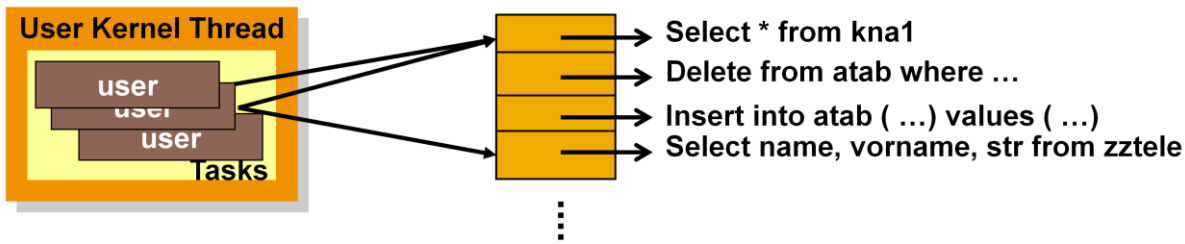
In online operation, shared SQL can be switched on and off dynamically. Changes are active for all new user sessions.

- 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 COMMANDCACHESTATISTICS

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

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

Values (in KB):

- Default: 262144 (256 MB)
- Min: 16384 (16 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.

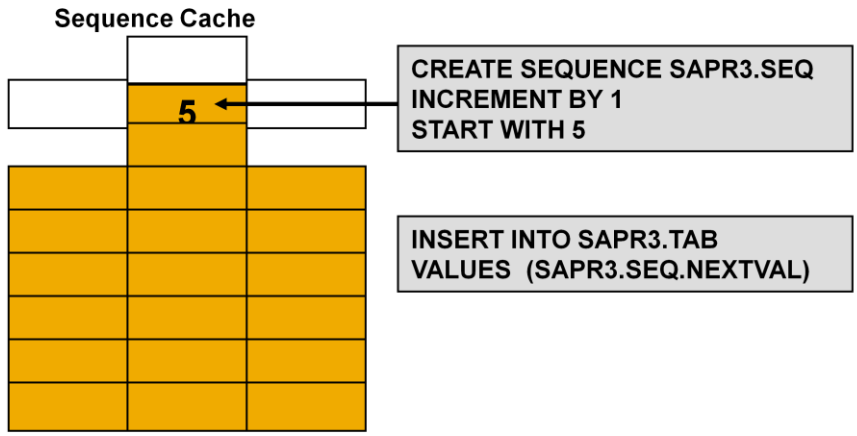
When the maximum value for the size of the Shared SQL cache has been reached, statements can still be formatted in the catalog cache. This does lead to higher execution costs, however.

The Shared SQL management maintains runtime statistics of the statement as of version 7.8. The parameter **EnableCommandMonitor** should have the value YES.

Sequence Cache

SequenceCacheSize (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
Online change: NO

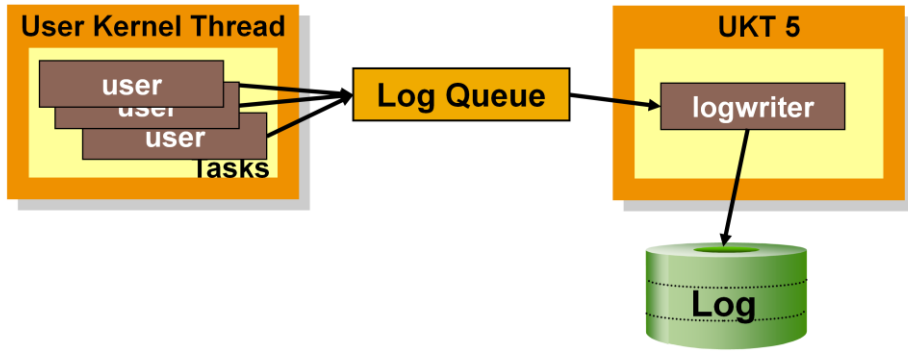
SQL statements

- SELECT * FROM MONITOR_CACHES
- MONITOR INIT executes a reset

Log Queue

LogQueueSize (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 and >= 200)
- Online change: No

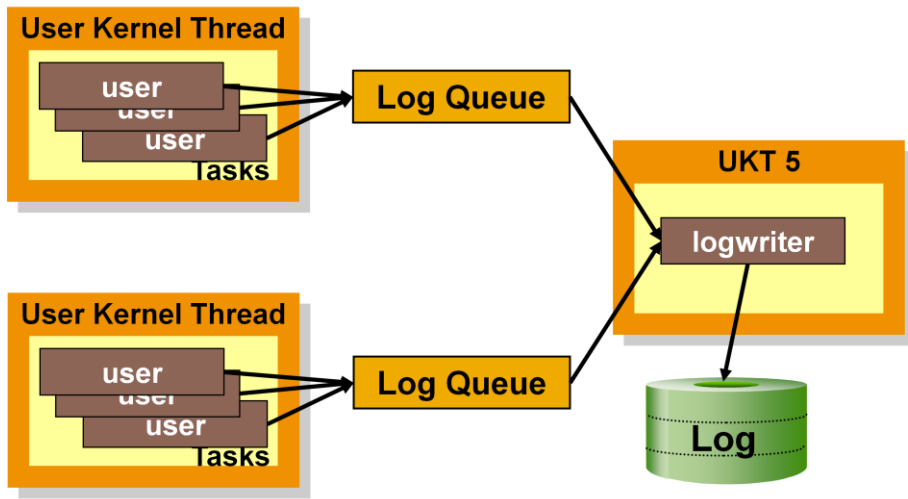
The system view LOGQUEUESTATISTICS displays the number of log queue overflows

- SELECT * FROM LOGQUEUESTATISTICS

Log Queue Count

LogQueues (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.

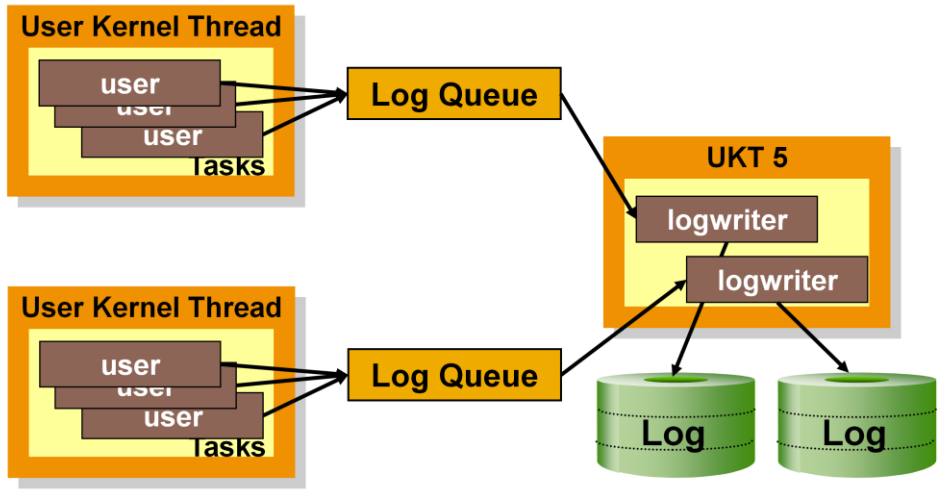
In the standard, the number of log queues is the same as the value for MaxCPUs. Thus every UKT with user tasks writes to its own log queue. So users can no longer collide with each other, but only with the log writer.

You can enlarge the value of parameter LogQueues in online mode up to the size of **MaxLogQueues.**

Values: Default: MaxCPUs
Min: 1, Max: MaxCPUs
Online change: YES

MaxLogWriterTasks

- Number of logwriter tasks within the system



If multiple log partitions are used the configuration of one logwriter task per UKT is suitable. Thus the database can prevent collisions during the access on the log queue.

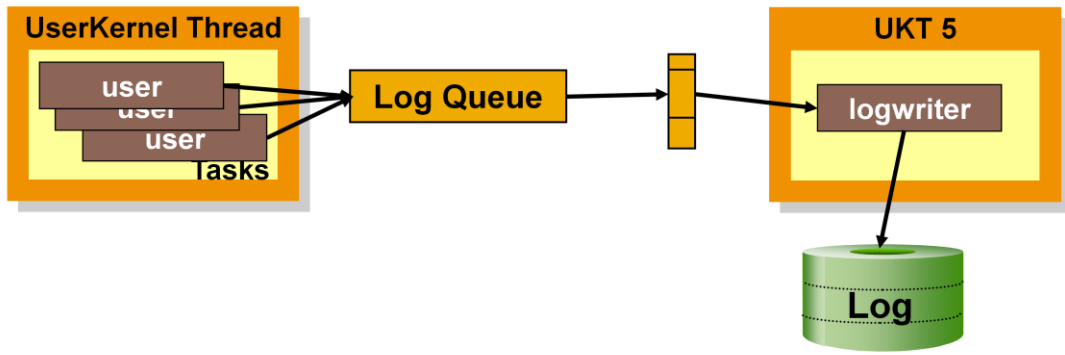
Values: Default: 1
Min: 1
Online change: Nein

This parameter was introduced temporarily to support multiple log partitions. It will be replaced by an automatic configuration in future.

Maximum Number of Log Pages per I/O

LogIOClusterSize (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.

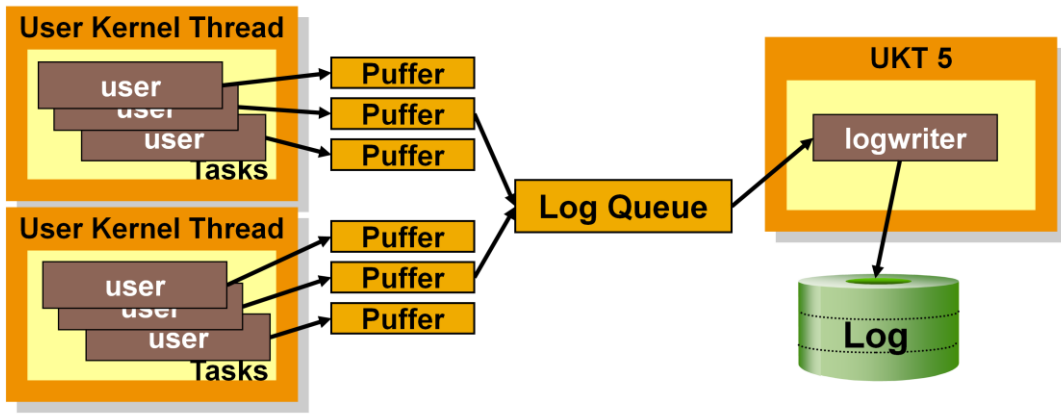


Values: Default: 4 (this amounts to 32 KB with 8 KB Pages)
Min: 4, Max: 32
Online change: NO

Transaction Buffer for Redo Log Entries

LocalRedoLogBufferSize (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 (MaxCPUs > 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.

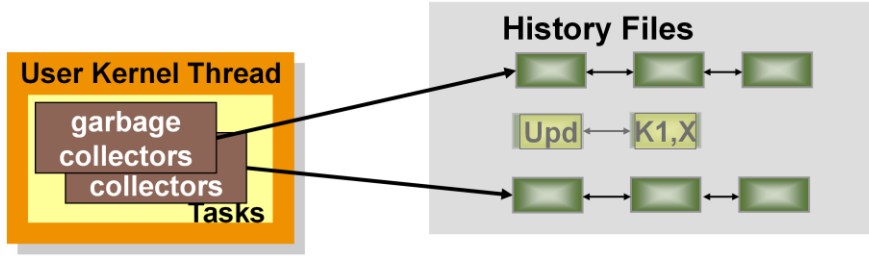
Values:	Default: 0	Transaction-dependent buffers for redo log entries are not used
	>0	Size per transaction-dependent buffer in bytes

Online change: NO

Number of Garbage Collectors

GarbageCollectors (_MAXGARBAGE_COLL)

- Number of Garbage Collectors releasing the pages of history files



Garbage collectors release history files which contain undo information that is not needed any longer.

If in a system several garbage collectors are configured then it is possible to cleanup also several history files simultaneously.

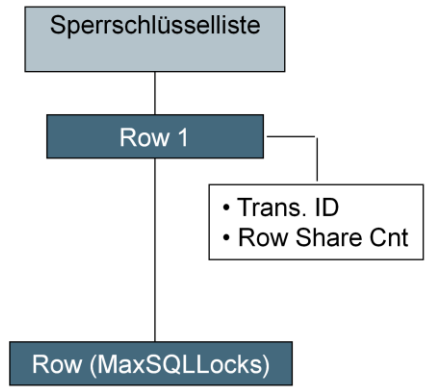
Additional garbage collectors do not influence the CPU load of the system as they are all working within the same UKT. Many garbage collectors, however, may increase the parallel I/O load in the system. It is recommended to set the parameter to 10.

Values: Default: 1
SAP Systems: 10
Online change: NO

Lock Lists

MaxSQLLocks (MAXLOCKS)

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



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

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

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

MaxSQLLocks should be increased if the following regularly occurs:

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

Alternatively, the number of write transactions can be reduced.

Values:

- Initial: 2500
- Minimum: 500
- MaxSQLLocks >= (((TransactionLockManagementStripes + TableLockManagementStripes + RowLockManagementStripes) * 100) DIV 3
- SAP systems: >= 150.000

Excessive values for MaxSQLLocks lead to longer search runs in the lock lists.

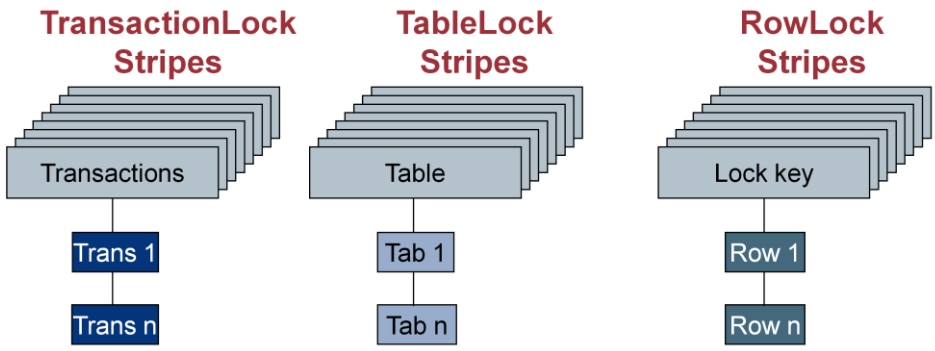
The criteria for increasing MaxSQLLocks mentioned above can be queried:

- SQL statements:
  - SELECT * FROM SYSINFO.LOCKSTATISTICS
- DBAnalyzer
- SAP system: transaction DB50

Sections of Lock List

TransactionLockManagementStripes (_TRANS_RGNS)
TableLockManagementStripes (_TAB_RGNS)
RowLockManagementStripes (_ROW_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

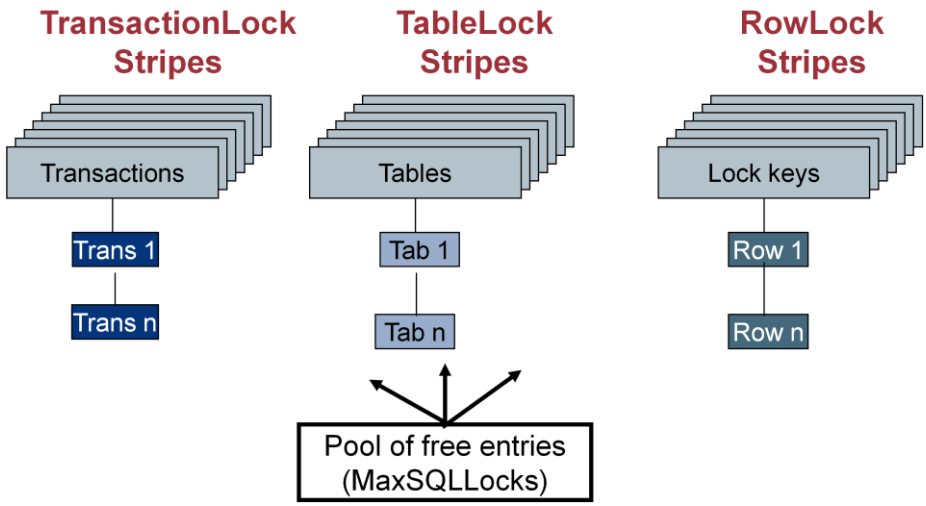
- Default: 8
- Min: 1, Max: 64

Online change: NO

Enlargement of Lock Lists in Blocks

InternalLockStructureEntries (_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 a single stripe 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 InternalLockStructureEntries.

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 MaxSQLLocks).

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

DeadlockDetectionLevel (DEADLOCK_DETECTION)

- Maximum search level for deadlock detection.

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

DeadlockDumpFilename

- Name of the file where to dump the internal structure of the lock list when a deadlock appears

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: YES (as of version 7.6.03)

MaxDB writes the internal structure of the lock list into a file with the name of the parameter value for DeadlockDumpFilename, if a value is given.

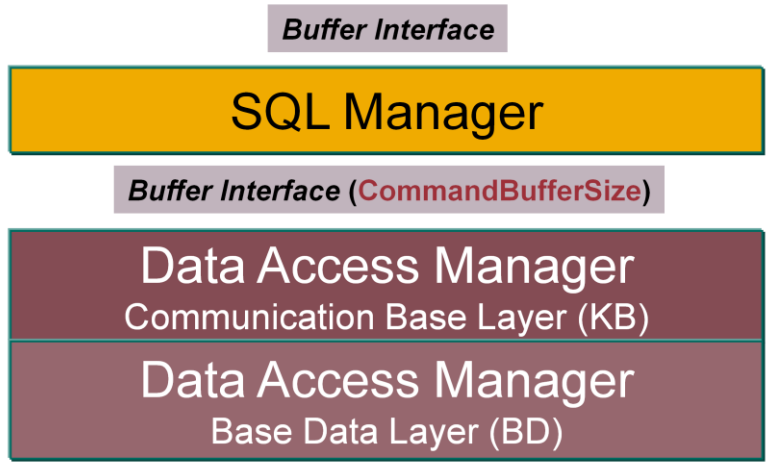
Communication, I/O

CommandBufferSize	Maximum length of communication packets
EnableSynchronousTaskIO	I/O only allowed through special threads
MaxPagerTasks	Number of tasks for writing data pages
MaxSavepointTimeInterval	Time passed between two savepoints
DataCacheIOAreaSize	Data cache area relevant for savepoints
DataCacheIOAreaFlushThreshold	Point of time when writing starts
DataCacheLRUAreaFlushThreshold	Point of time when writing starts
DataIOClusterSize	Max. number of blocks per I/O operation
ClusterWriteThreshold	Generation of physical table clusters
UseLobClustering	Generation of clusters for LOB data
VolumeIOQueuesFor<…>Priority	Number of I/O queues per volume
IOQueueFillingThreshold	Threshold value for the use of further I/O queues
ReadAheadLobThreshold	Asynchronous read of pages of a LOB value
UseAutomaticBadIndexRecreation	Automatic creation of indexes during restart
ConverterVolumeIdLayout	Maximum size or number of data volumes
UseVolumeLock	Enforce lock of volumes while attached
UseFilesystemCacheForVolume	By-passing the Linux file system cache

Communication Packet

CommandBufferSize (_PACKET_SIZE)

- CommandBufferSize limits the maximum length of the communication packets 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: 131072

- 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

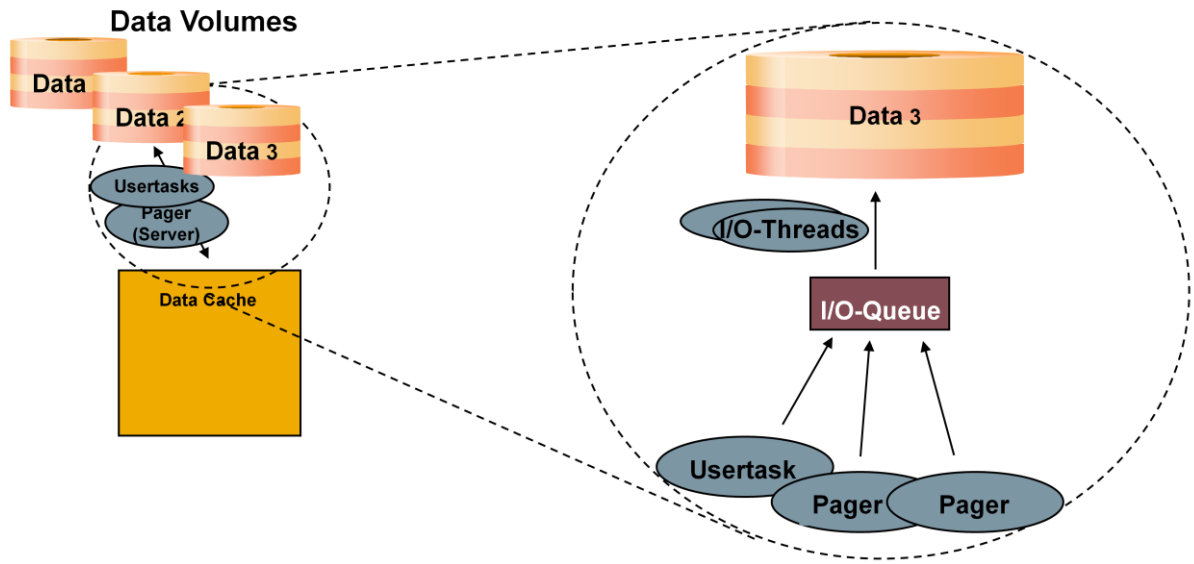
Typical errors if CommandBufferSize is too small:

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

Asynchronous I/O

EnableSynchronousTaskIO (_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 EnableSynchronousTaskIO is set to “YES" 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.

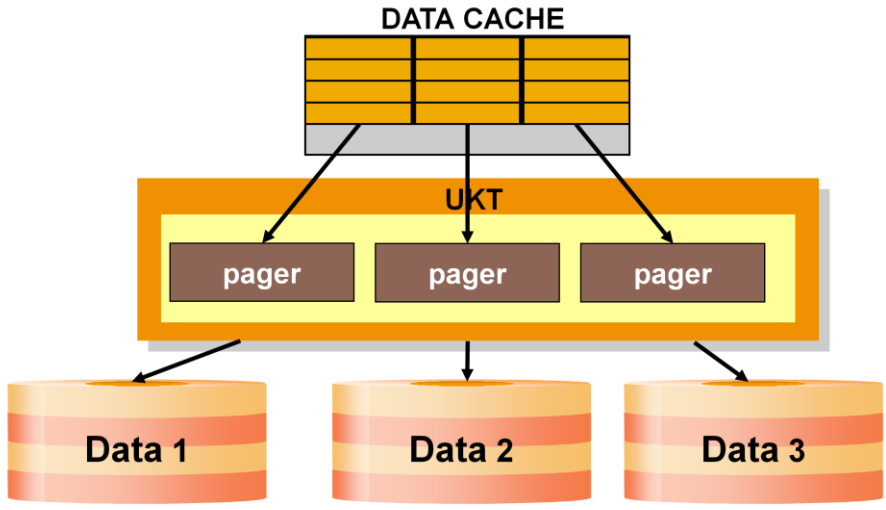
If a user task executes an I/O request itself, other tasks cannot work until it is finished. The UKT is blocked and waits for the reply to the I/O request. This option can compromise performance in parallel operation.

Values: Default: YES
Online change: YES

Pagers

MaxPagerTasks (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 EnableSynchronousTaskIO).

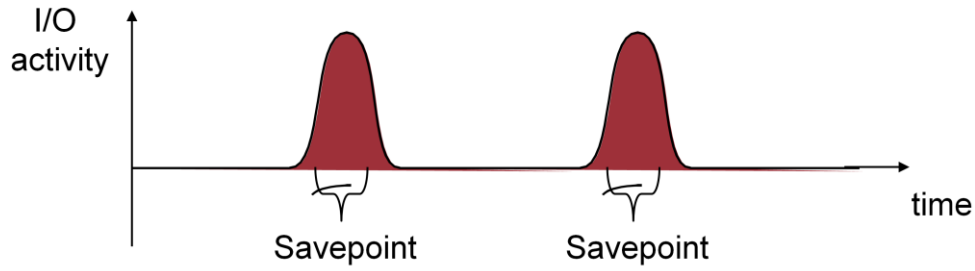
With an online restart, the pagers read the converter in parallel.

Values: Min: 1, Max: 64
Default: Maximum of DataCacheStripes, ConverterStripes and MaxDataVolumes
Online change: NO

Time between two Savepoints

MaxSavepointTimeInterval (_RESTART_TIME)

- Maximum time period in seconds between the end of a savepoint and the start of the next savepoint if pages have been modified in the data cache.



If the MaxSavepointTimeInterval 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

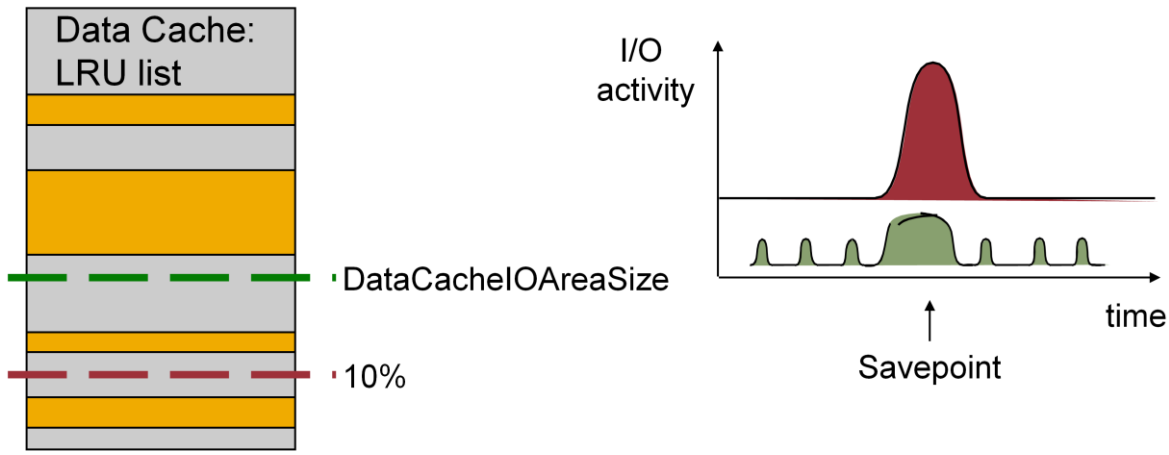
The time period calculation in this case uses the end of the prepare phase as the end of the last savepoint.

SAP does not recommend changing the value of this parameter. If desired, you can influence the behavior of savepoints with the parameters DataCacheIOAreaSize, DataCacheIOAreaFlushThreshold and DataCacheLRUAreaFlushThreshold.

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.

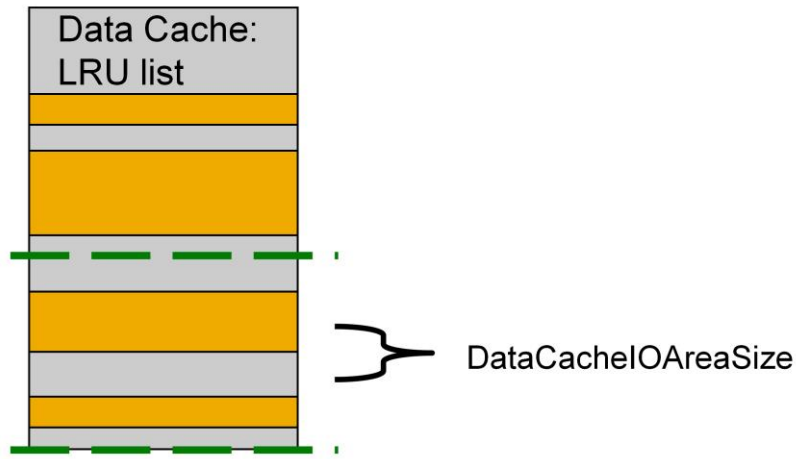
Only those pages from a part of the cache, the DW-IO-AREA, are written. The size of this area is controlled by a parameter.

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

DataCacheIOAreaSize (_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 DataCacheIOAreaSize.

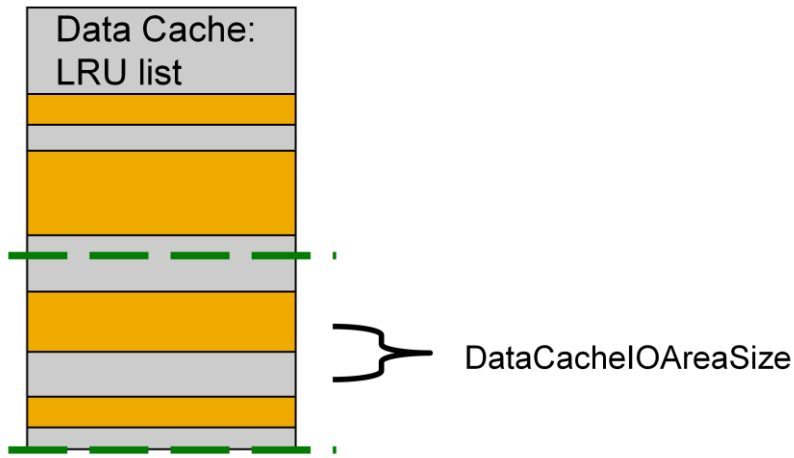
A large value reduces the savepoint I/O load more, but increases the current I/O load.

Values: Default: 50
Min: 10, Max: 90
Online change: NO

Threshold Value for Activation of Pagers (I)

DataCacheIOAreaFlushThreshold (_DW_IO_AREA_FLUSH)

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



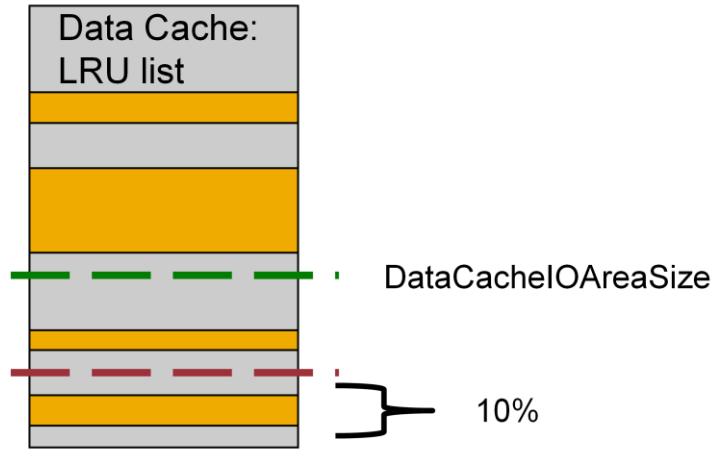
This parameter specifies an event at which the pagers become active.
A large value reduces the savepoint I/O load more, but increases the current I/O load.

Values: Default: 50
Min: 30, Max: 80
Online change: NO

Threshold Value for Activation of Pagers (II)

DataCacheLRUAreaFlushThreshold (_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.

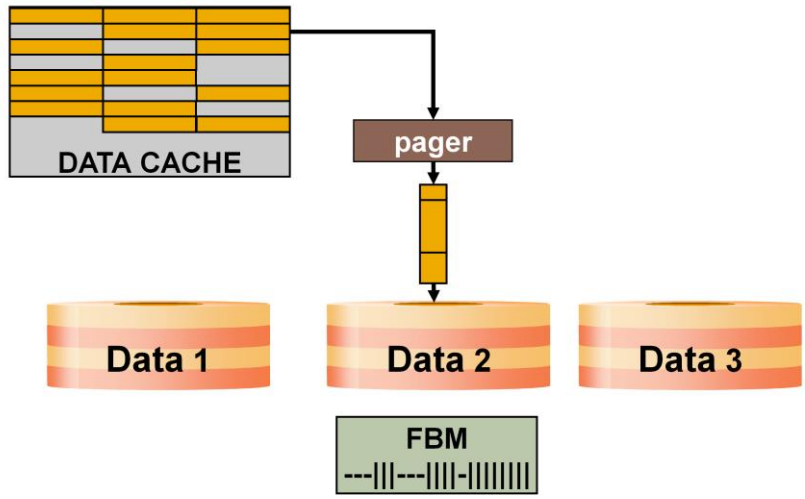
The purpose of this mechanism is to ensure that there are always enough free pages at the end of the LRU that user tasks are not forced to displace pages from the data cache.

Values: Default: 25
Min: 10, Max: 80
Online change: NO

Blocked I/O

DataIOClusterSize (DATA_IO_BLOCK_COUNT)

- 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).

If for a table the cluster flag is switched on the database builds groups of pages that belong together according to the B* tree chains before writing it to the disks. Thus data pages are kept together logically to improve the use of prefetch algorithms of the storage systems during scan operations. The cluster flag is set with the CREATE TABLE or ALTER TABLE statement.

This parameter also influences the block sizes for read / write operations to data backups. Backup templates provide a mechanism to adapt block sizes used for writing to the backup media.

If the cluster flag is used for tables the block size for writing a backup should be set according to the DataIOClusterSize or should be a multiple of it. Otherwise the clusters would be dissected during restore.

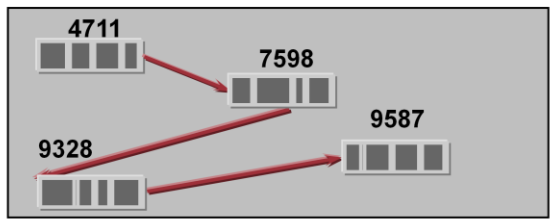
Values: Default: 64
Min: 4, Max: 128
Online change: No

Use of Cluster Areas (I)

ClusterWriteThreshold (CLUSTER_WRITE_THRESHOLD)

- Value in percent beginning with which cluster size the data of cluster tables is written to separate FBM sectors even if the sector is not completely filled.

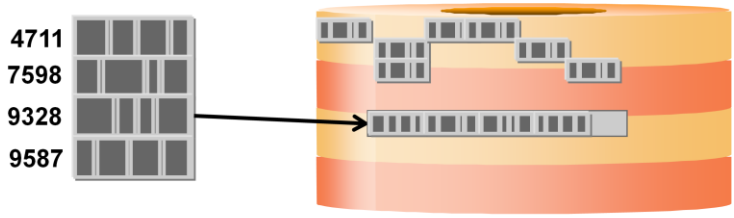
Data Cache



Converter

Page	Volume	Offset
4711	1	9857
...		
7598	1	9858
...		
9328	1	9859
...		
9587	1	9860

Data



MaxDB builds clusters for tables with the cluster flag to improve read performance for scans.

If blocks are written for cluster tables the pager tasks are looking for logically clustered blocks. Logically clustered blocks are those with successive cluster keys. The cluster key is defined by the primary key or another logical key which must not be unique on application side (f.e. time characteristic). Pager tasks write those blocks adhesively to the data area.

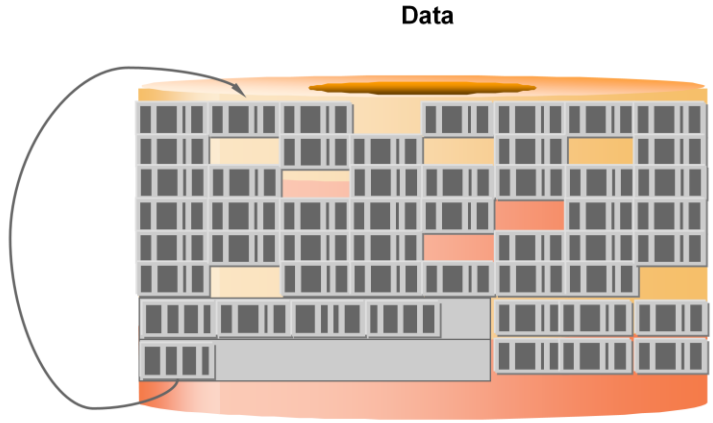
A cluster builded by pager tasks is only written to a separate FBM section if the number of blocks within the cluster is at least ClusterWriteThreshold % of DataIOClusterSize and a free section in the data volumes is available. During backup and restore the clustering is not lost. If the percentage falls below ClusterWriteThreshold and no more free section is available the cluster is splitted and written to different free blocks.

Values: Default: 80%
Min: 0, Max: 100
Online Änderung: Ja

Use of Cluster Areas (II)

ClusterCompressionFillThreshold

- If a cluster only holds ClusterCompressionFillThreshold % of a FBM section then this cluster is read, marked as changed within the cache and will be written to another position with the next savepoint. This generates space for larger clusters that can use a section in a better way.



If the database is filled to a high amount there is increased risk of writing too small clusters because there are no more free FBM sections for bigger clusters. So the scan performance of the system will be restricted.

FBM sections are released if they are only filled with a few blocks.

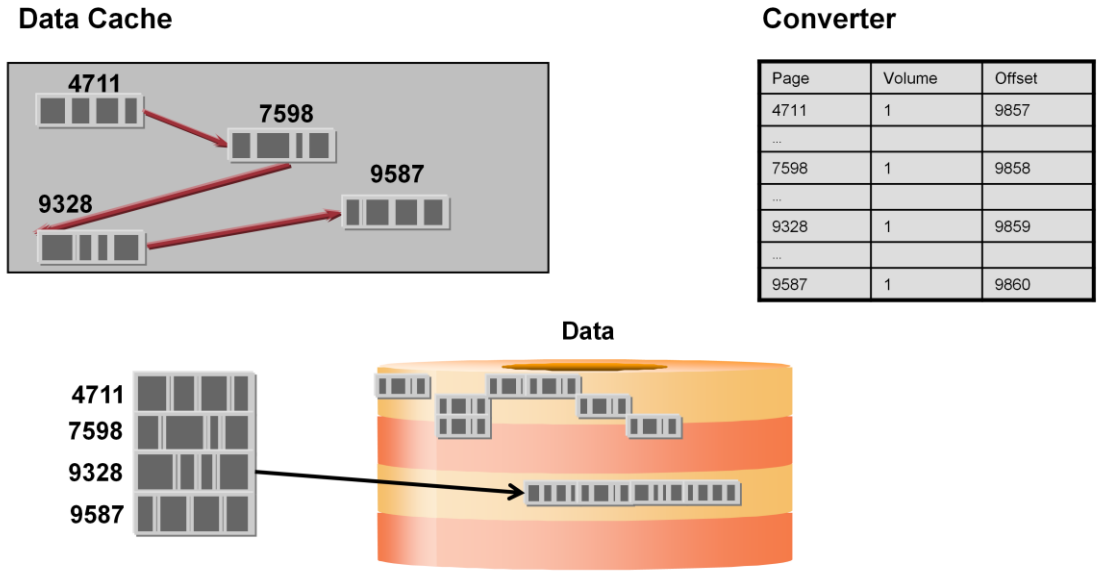
At the end of a savepoint it is checked by pager tasks if there are FBM sections with a low filling grade. Server tasks read the affected blocks to the data cache and mark it as modified. The blocks are written to other positions in the data area at the latest with the next savepoint. The FBM sections are now free for large table clusters.

Values: Default: 10%
Min: 0, Max: 50
Online Change: YES

Building Clusters for LOBs

UseLobClustering (CLUSTERED_LOBS)

- Storage of blocks with LOB values in clusters.



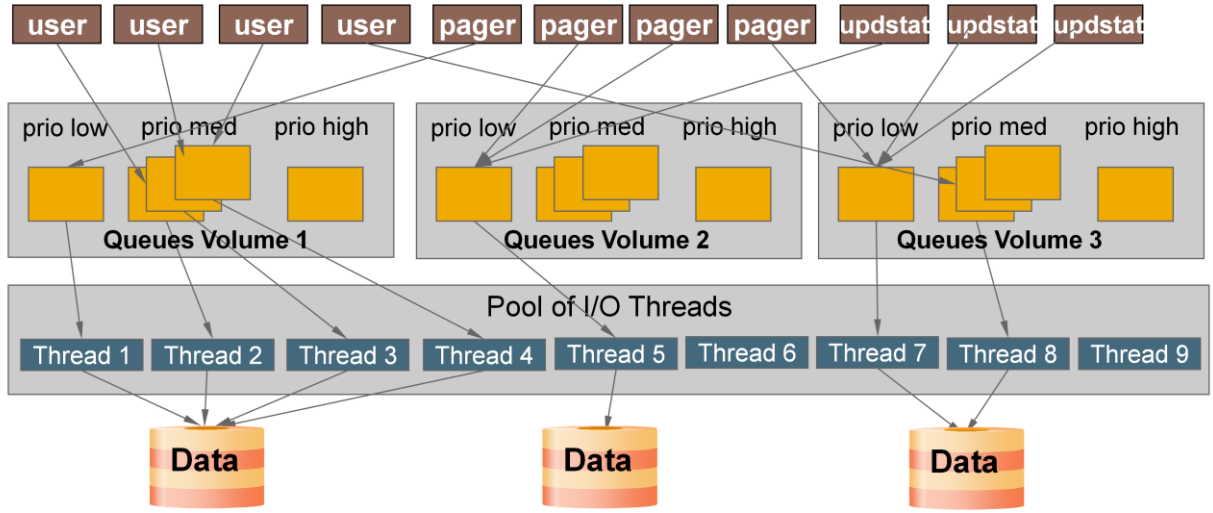
Pager tasks also build clusters for LOB values if the parameter UseLobClustering is set to YES.

The storage of blocks for LOB data is also influenced by the parameter ClusterWriteThreshold.

Values: Default: YES
Online Change: YES

I/O Thread Implementation

Scalability of asynchroneous I/O has been improved in version 7.7 significantly. In older versions the I/O threads were directly associated with the volumes. As of version 7.7 I/O threads can send their requests to different volumes. There is a configurable number of queues per volume. It is possible to assign priorities to I/O requests. Tasks don‘t have to wait for the result of the I/O but can send the request asynchroneously and continue their work.



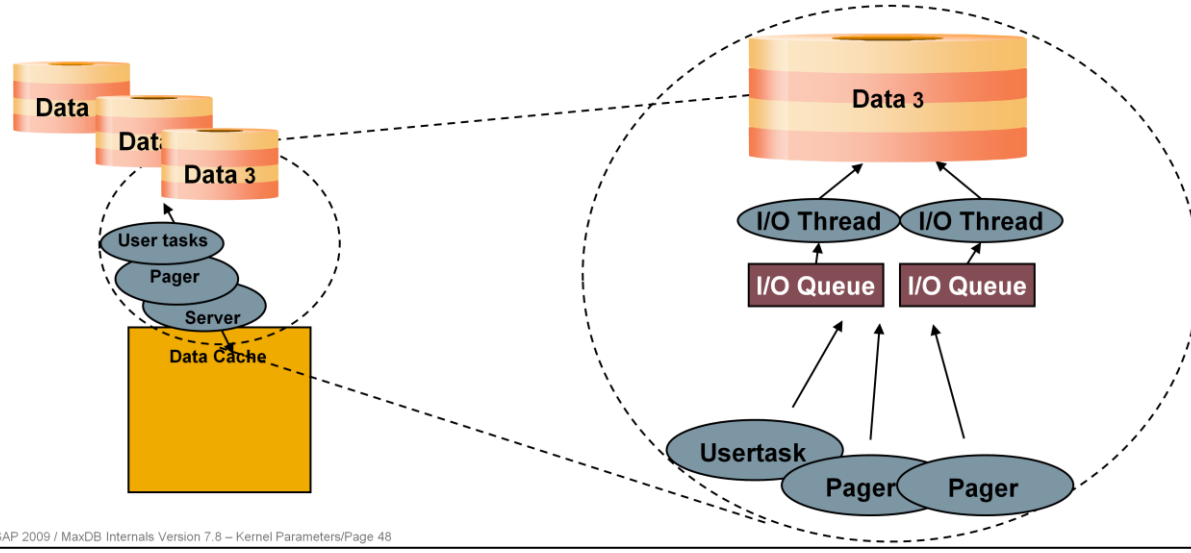
With version 7.7 the I/O interface to the operating system has been reimplemented. Version 7.7 uses different parameters than version 7.6. The new I/O system has the following essential advantages:

- No direct assignment of a I/O worker thread to a volume. This implies a better scalability of I/O.
- I/O worker threads can be started on request. This prevents the use of unnecessary resources.
- The synchronization of accesses to the I/O queues has been changed. The access is done collision free. This additionally improves the scalability of I/O.
- Prioritization of special I/O requests. Dedicated jobs within the database (f.e. CHECK DATA) can run with lower priority. Online operation is stressed less.
- Tasks can send I/O requests asynchroneously to the I/O system. They don‘t have to wait until the I/O request has been fulfilled but can continue their work.
- Support of multiple database instances.

I/O Threads per Database

EnablePreAllocateIOWorker
MinIOPoolWorkers, MaxIOPoolWorkers
IOPoolIdleTimeout

- Minimum and maximum number of I/O worker threads within a database and their maximum duration if they are not used.



The parameters shown above were introduced with the multiple database concept. This allows the use of several MaxDB databases within one instance. The parameters can be used to restrict I/O resources per database within one instance.

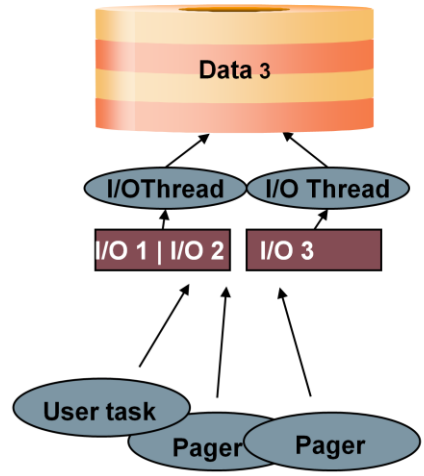
If you use a database per instance then it is not necessary to adapt these parameters. On request the database can start additional I/O worker threads. It is possible to restrict the number of I/O worker threads by setting the parameter MaxIOPoolWorkers.

Multi DB concept is not yet available.

I/O Queues per volume

VolumeIOQueuesForLowPriority
VolumeIOQueuesForMediumPriority
VolumeIOQueuesForHighPriority

- Number of I/O queues for requests with low, medium and high priority per volume



By defining the number of queues per volume and per priority you can influnce the priorities of I/O for certain requests.

VolumeIOQueuesForLowPriority:
Default: 1
Min: 1
Max: 10
Online Change: NO

VolumeIOQueuesForMediumPriority:
Default: 4
Min: 0
Max: 20
Online Änderung: NO

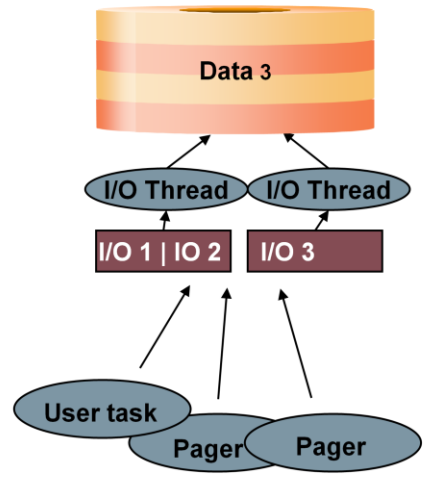
VolumeIOQueuesForHighPriority:
Default: 5
Min: 0
Max: 10
Online Änderung: NO

You can watch the states of current I/O requests in the system by the use of the console (x_cons) and the system view IOJOBS.

Threshold value for the use of additional I/O queues

IOQueueFillingThreshold (_IOPROCS_SWITCH)

- Defines the threshold value from which number of I/O requests it is changed to another I/O queue.



As soon as there are more than IOQueueFillingThreshold requests in the queue of an I/O thread the system tries to put each additional I/O request to another I/O queue.

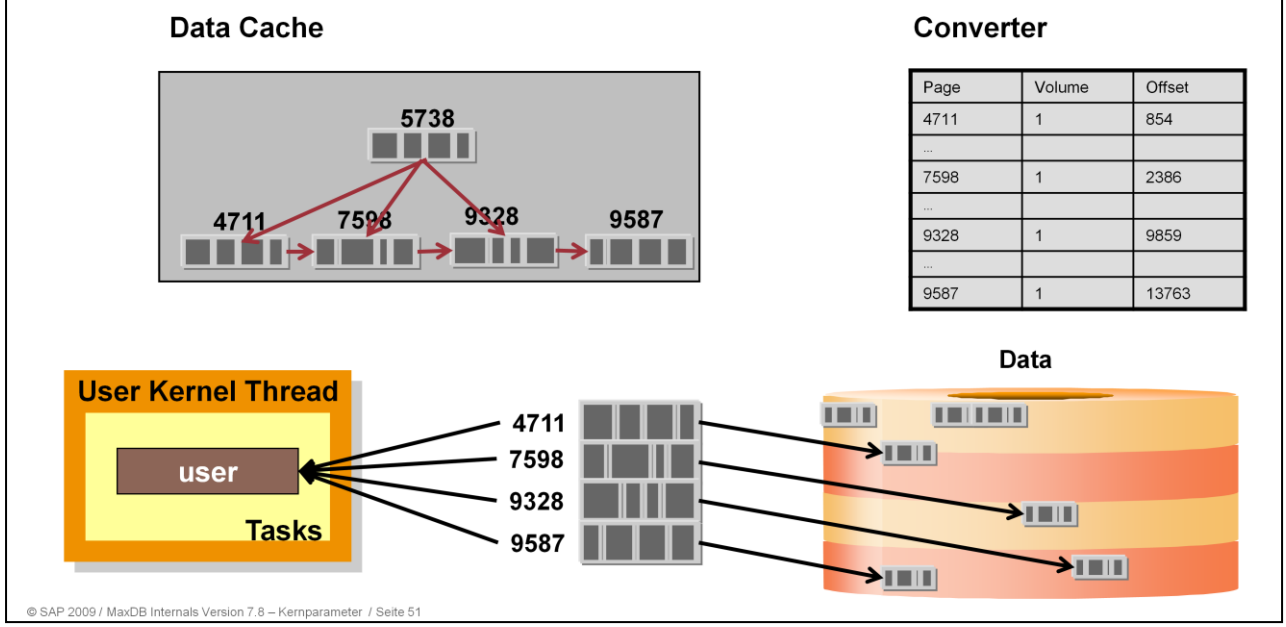
Values: Default: 1 (recommended)
Min: 0
Online change: YES

If there are not enough I/O worker threads available to handle all filled queues then the system automatically starts an additional thread.

Asynchronous Read of table Pages

ReadAheadTableThreshold

- User tasks utilize asynchronious read requests to scan tables or ranges of tables with parallel I/O



As of verison 7.8 MaxDB uses parallel I/O request to speed up table scans and table range scans. User tasks read index level 1 pages into the cache, determine the volume block positions of the pages stored in the separators by reading the converter and send asynchronious I/O requests to the I/O system. The user task doesn‘t wait for every single I/O before sending the next I/O request.

User tasks use asynchronious I/O requests if the size of the scan acceeds the number of pages specified as value of the parameter **ReadAheadTableThreshold.** The query optimizer evaluates the range size before the statement execution starts.

The database uses asynchronious I/O for scans only, if the number of current running I/O requests is below the value of **MaxDataAreaReadAheadRequests**. The determination of the current running I/O requests happens during the operation on the index level 1 page. This operation prevents the system from I/O overload situations. I/O orders for short SQL commands should not be blocked by the aysnchronious parallel I/O.

Asynchronious I/O read requests have the priority low.

Values:

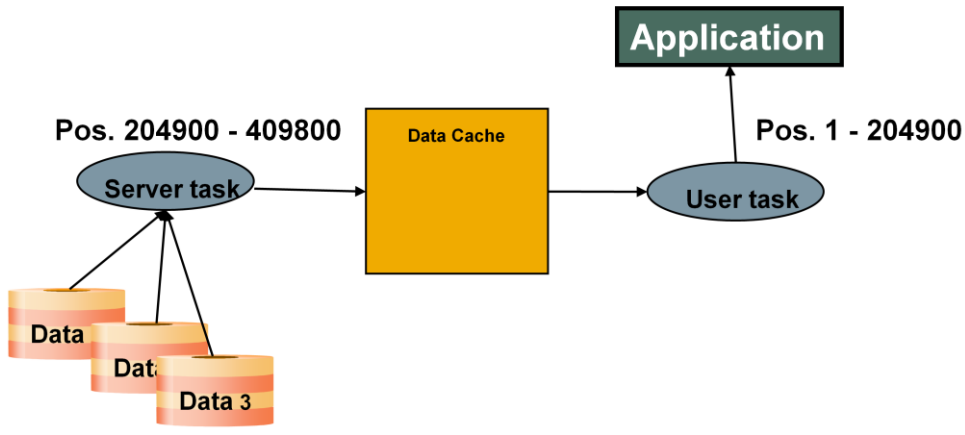
ReadAheadTableThreshold:	Default: 100 Pages Min: 0 Max: 2147483647 Online Change: Yes
MaxDataAreaReadAheadRequests:	Default: 2 x Number of Data Volumes Min: 0 Max: 2147483647 Online Änderung: Yes

Monitoring: select * from monitor_pages where description like 'Read ahead%'

Asynchronous Read of LOB Pages

ReadAheadLobThreshold (_READAHEAD_BLOBS)

- 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 LOB 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 LOB value to be read exceeds the value of the parameter ReadAheadLobThreshold.

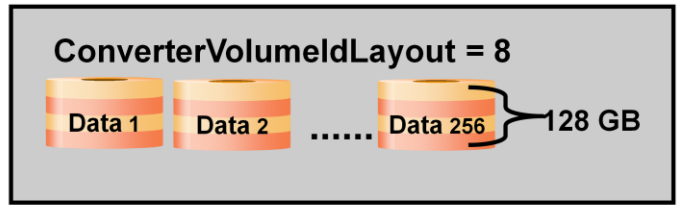
Values: Default: 25

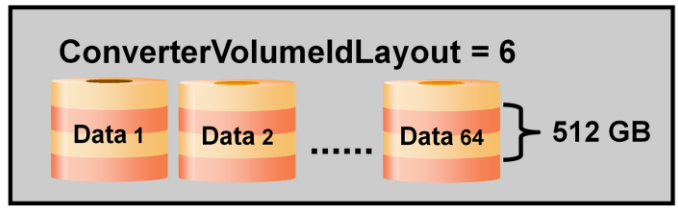
- Min: 2 * CommandBufferSize / 8196 Max: 262144
- Online change: YES

Maximum Size or Number of Data Volumes

ConverterVolumeIdLayout (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

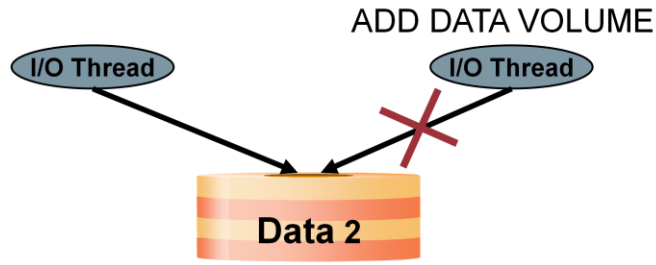
Online change: No

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

Lock on Attached Volumes

UseVolumeLock (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 UseVolumeLock.

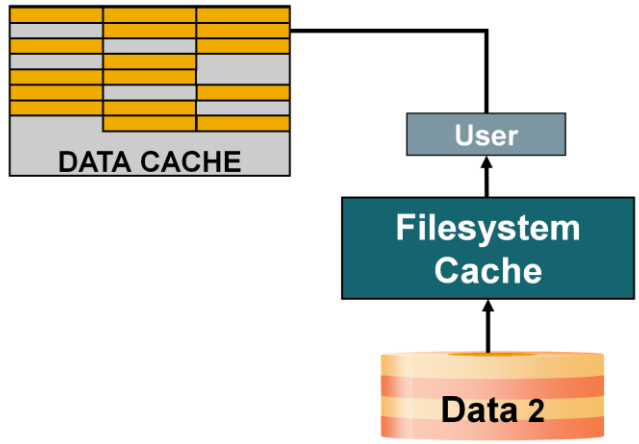
Values: Default: YES The database requests a lock when a volume is opened.

Online change: NO

Bypassing the File System Cache

UseFilesystemCacheForBackup
UseFilesystemCacheForVolume (USE_OPEN_DIRECT)

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



If the parameter UseFilesystemCacheForVolume is set to NO then the database opens the volumes by using the flag O_DIRECT.

If the parameter UseFilesystemCacheForBackup is set to NO then – for a backup - the database opens the volumes and backup media (in case of files) by using the flag O_DIRECT.

The database kernel cannot open the files in the volumes upon starting if one of the parameters is set to NO although the option O_DIRECT is not supported by the file system. The mount options should force direct I/O for the file system in those cases.

Please additionally have a look at note 993848 which gives recommendations concerning mount options for different file systems. Note 977515 describes file system behaviors during backups and provides special recommendations for the settings of these parameters.

Attention: By renaming the parameter it now got the inverse meaning.

Values:

UseFilesystemCacheForVolume

Default: NO
Online change: NO

UseFilesystemCacheForBackup

Default: YES
Online change: NO

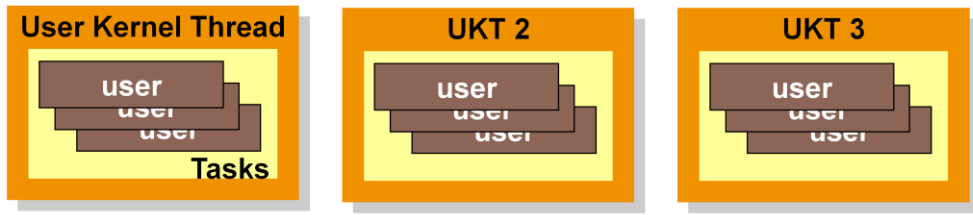
Process Structure, CPU Usage

TaskCluster01 ... 03	Distribution of tasks to threads
MaxCPUs	Maximum number of intensely used processors
UsableCPUs	Maximum number of currently intensely used processors
LoadBalancingCheckInterval	Load balancing between UKTs
LoadBalancingWorkloadThreshold	Threshold value for load balancing
LoadBalancingWorkloadDeviation	Precision of measurement for load balancing
EnableMultipleServerTaskUKT	Distribution of server tasks to UKTs
ExclusiveLockRescheduleThreshold	Stop running tasks
MaxExclusiveLockCollisionLoops	Busy Waiting (BW) during collisions
TaskSchedulerRetryLoops	Dispatcher loops
Task…Priority	Base priority per task state
TaskNiceElevationValue	Additional priorities for lockholders
ForceSchedulePrioritizedTask	Interruption of tasks

Maximum Number of Intensively Used CPUs

MaxCPUs (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, MaxCPUs 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 server with up to 7 CPUs: 100% of available CPUs

Dedicated database server with more than 7 CPUs: 100% of available CPUs -1

Online change: NO

Distribution of Tasks to User Kernel Threads

TaskCluster01 bis 03 (_TASKCLUSTER_01 bis _03)

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

- Example:
  - MaxUserTasks = 40, MaxCPUs = 2
  - TaskCluster01 = 'tw;lw;ut;2000*sv;10*ev,10*gc'
    TaskCluster02 = 'ti,100*pg;20*us;'
    TaskCluster03 = '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

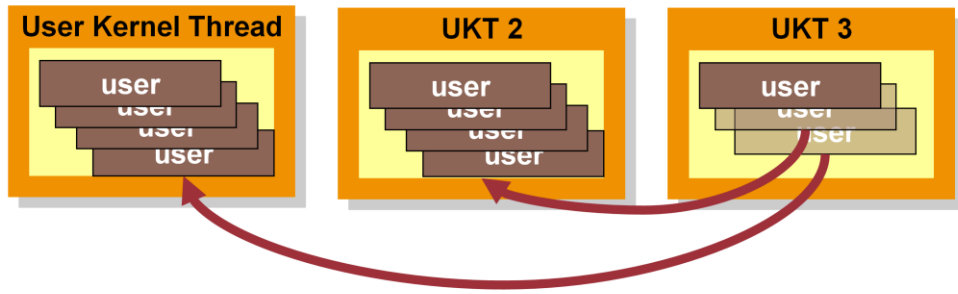
Warning:

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

Maximum Number of currently used CPUs

UseableCPUs

- Limitation of the number of User Kernel Theads currently used by user tasks
- The dispatcher moves user tasks from inactive User Kernel Threads to active UKT if the values of UseableCPUs is smaller than MaxCPUs



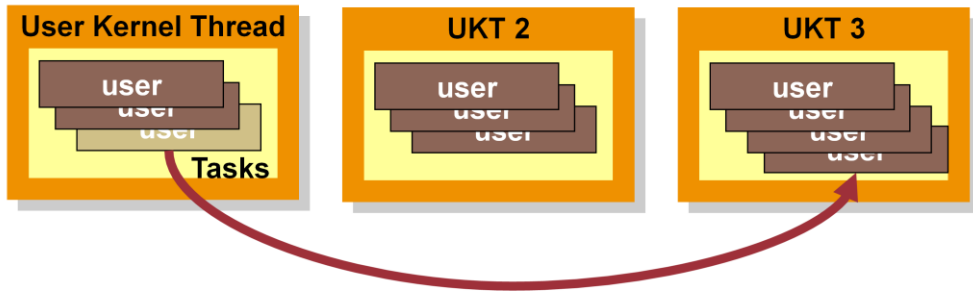
As of version 7.8 MaxDB can dynamically adjust the number of CPU cores to use. The dispatcher moves user tasks out of the inactive user kernel threads when the tasks become active.

Values: Default: MaxCPUs
Min: 1
Max: MaxCPUs
Online Change: Yes

Load Balancing Between UKTs

LoadBalancingCheckInterval (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 LoadBalancingCheckInterval 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 LoadBalancingCheckInterval 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.

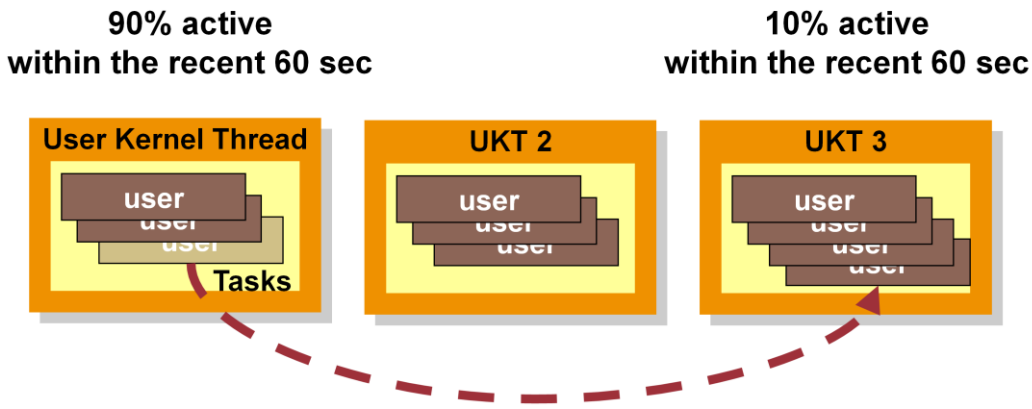
In OTLP operation, unbalanced load distribution among the UKTs is usually due to poorly-optimized statements with high CPU loads for a single job. For this reason, such statements should be identified and optimized before load balancing is employed.

Values: Default: 4
Min: 0, Max: 600
Online change: Yes

Threshold Value for Load Balancing

LoadBalancingWorkloadThreshold (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 LoadBalancingWorkloadThreshold 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:

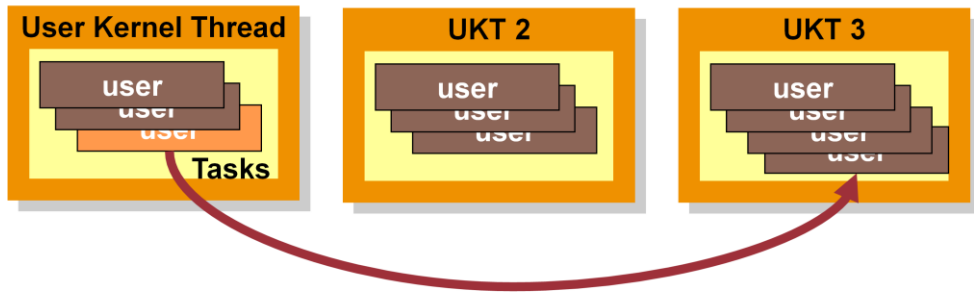
```
x_cons <dbname> show moveinfo
x_cons <dbname> show t_move t<taskid>
dbmcli –d <dbname> -u <dbmuser,passwd> -n <server> show …
```

Values: Default: 10
Min: 0, Max: 99
Online change: Yes

Precision of Measurement for Load Balancing

LoadBalancingWorkloadDeviation (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.

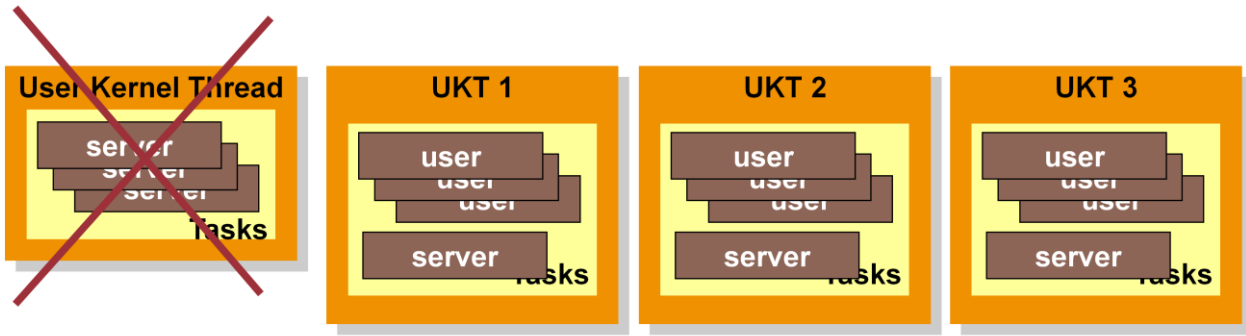
Recorded time data for activities are considered equal if the difference between them does not exceed a certain percentage. The parameter LoadBalancingWorkloadDeviation defines this percentage.

Values: Default: 5
Min: 0, Max: 50
Online change: Yes

Distribution of Server Tasks to UKTs

EnableMultipleServerTaskUK (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



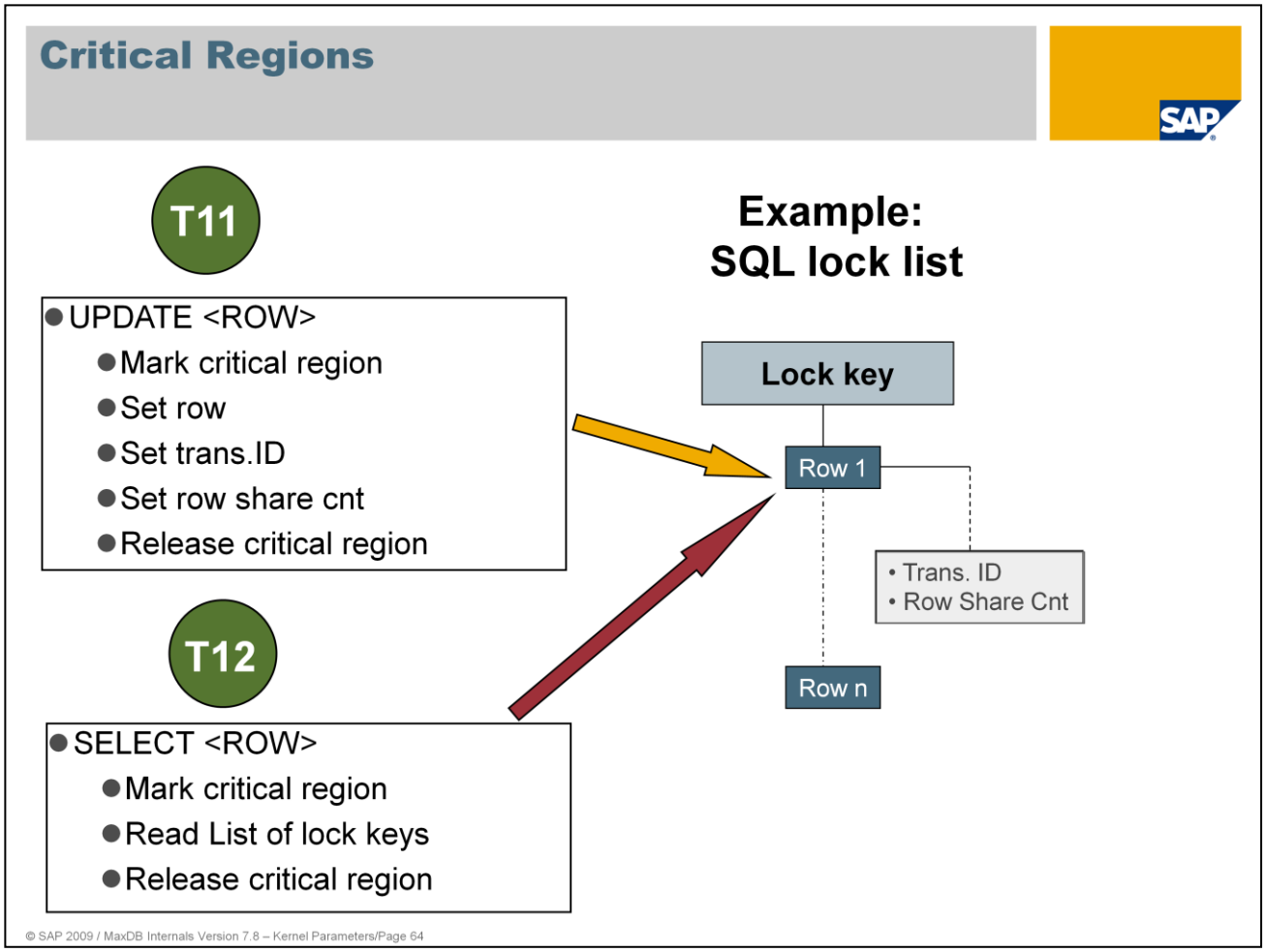
With the standard setting, server tasks are assigned to a UKT. In systems with many data volumes and fast I/O, the CPU load generated by the server tasks can lead to a bottleneck within the thread.

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

With the setting EnableMultipleServerTaskUKT=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.

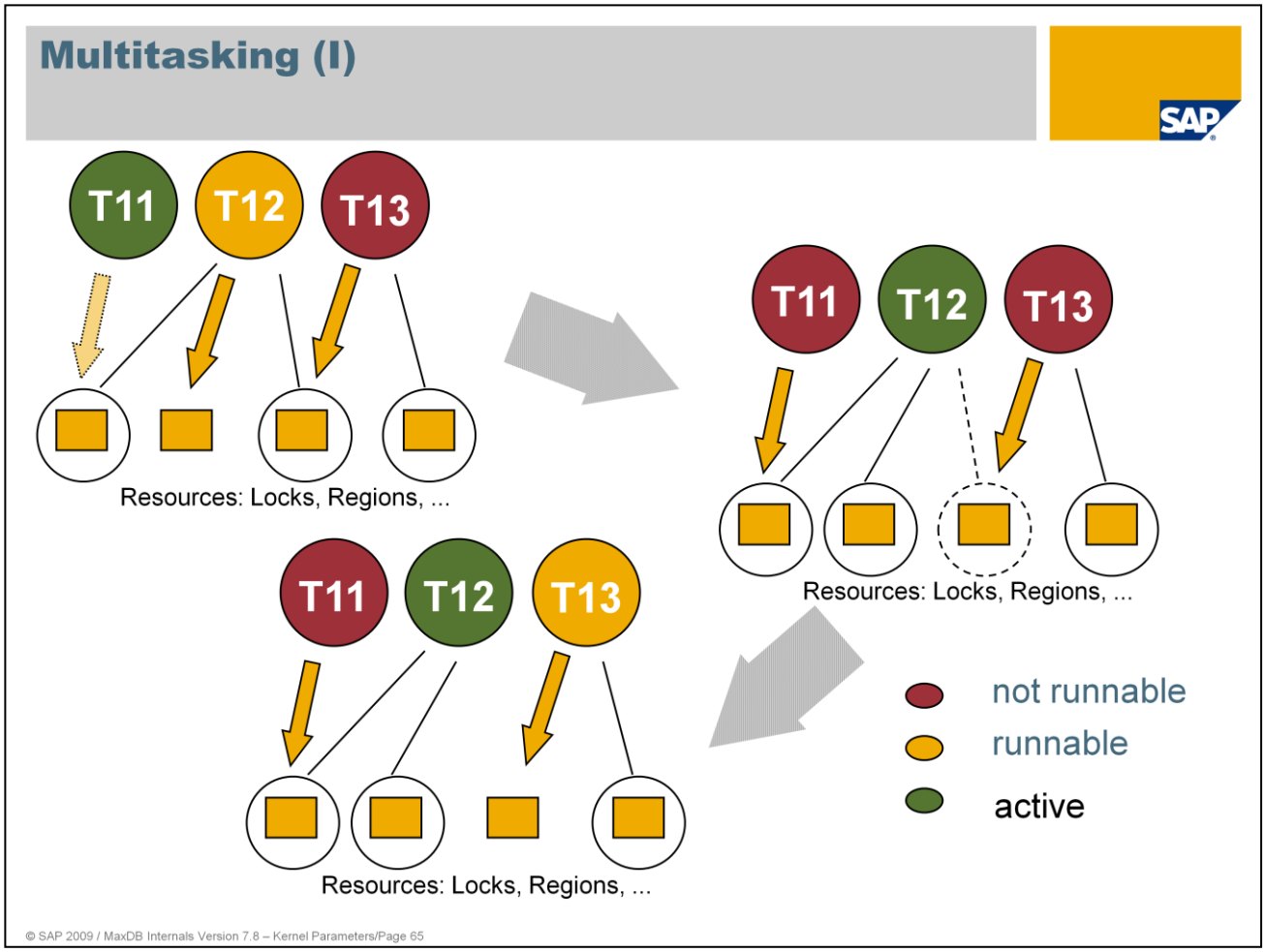
The source code, which can only be run by one task at a time, is designated a critical region. Defining regions enables synchronized access to resources that can be read or changed by multiple tasks in parallel.

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.

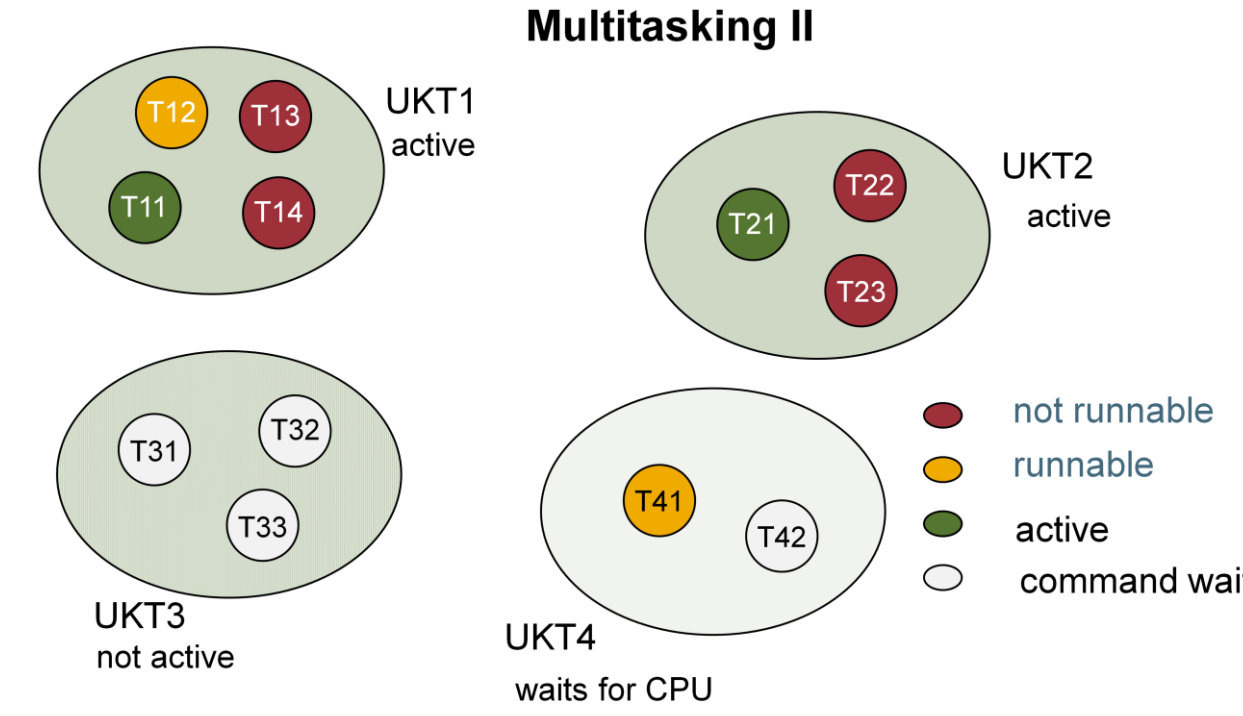
The illustration:

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.

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.

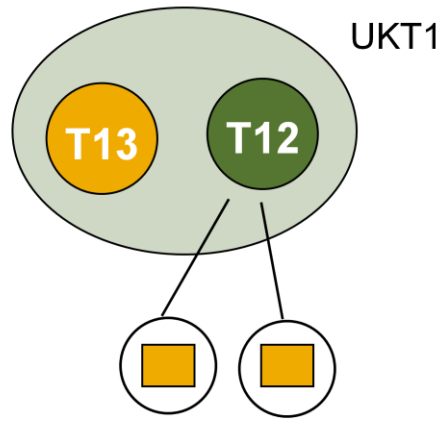
UKT4, if it were granted CPU time by the operating system, would take off and activate task T41.

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

Stop Running Tasks

ExclusiveLockRescheduleThreshold (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 ExclusiveLockRescheduleThreshold, 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.

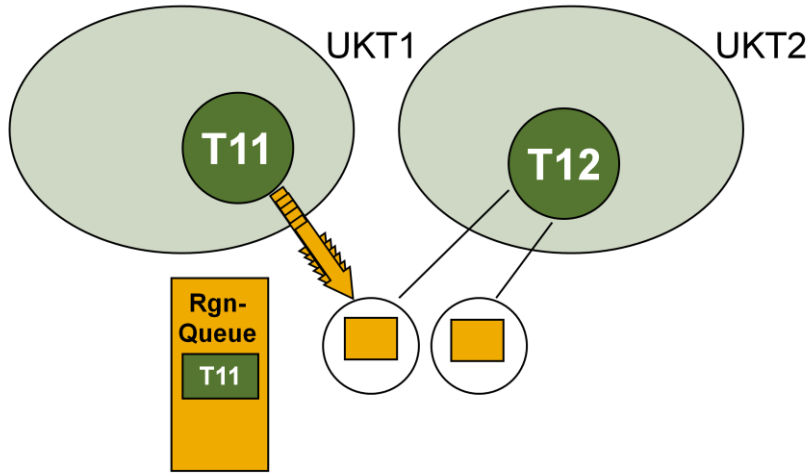
This parameter is especially important for single-processor systems.

Values Default: MaxCPUs = 1: 300; MaxCPUs > 1: 3000
Min:100, Max: 100000
Online change: YES

Busy Looping

MaxExclusiveLockCollisionLoops (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 MaxExclusiveLockCollisionLoops 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 MaxCPUs = 1
100 if MaxCPUs = 2-7
10000 if MaxCPUs > 7 (starting in Version 7.6.02)

MaxExclusiveLockCollisionLoops should not be > 0 , if MaxCPUs = 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.

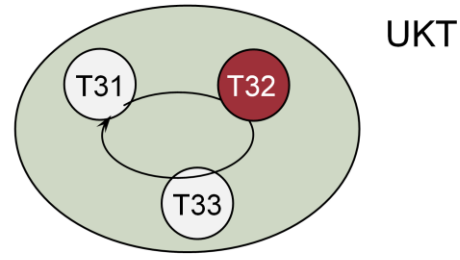
The parameters _MP_RGN_QUEUE, _MP_RGN_DIRTY_READ and _MP_RGN_BUSY_WAIT have been removed with version 7.7. They haven‘t been changed in the systems for a long time. The removal saves some efforts for dispatching.

Dispatcher Loops

TaskSchedulerRetryLoops (_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.

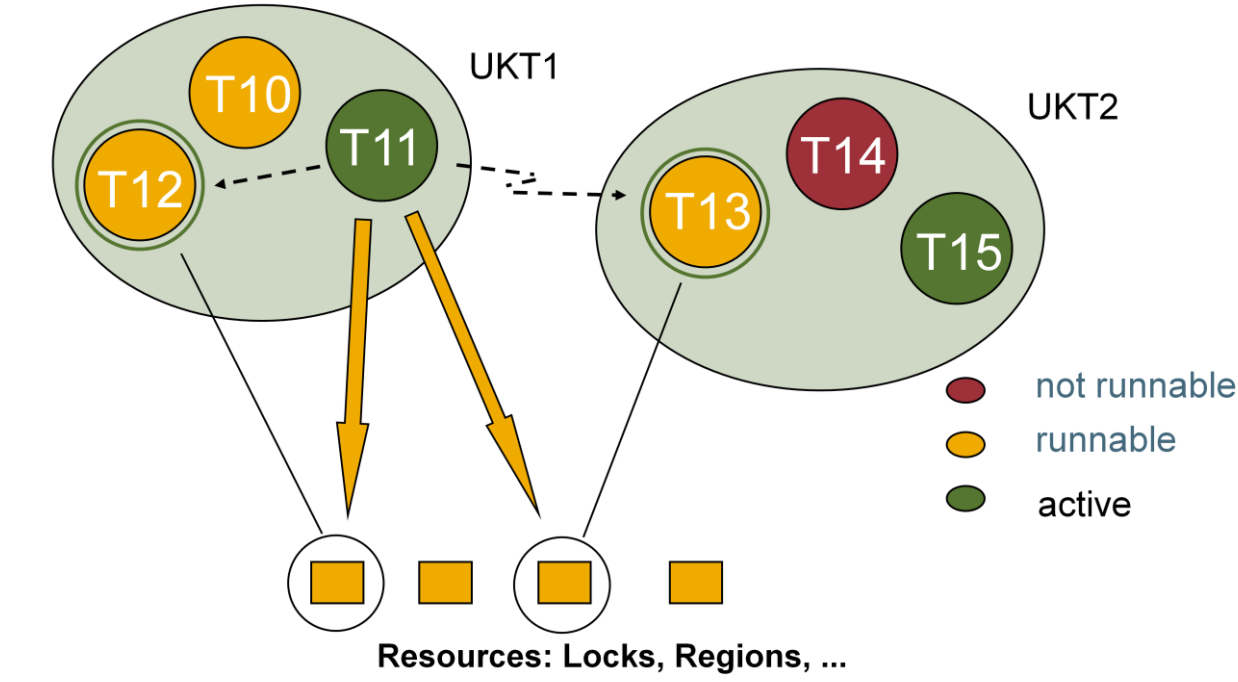
- Values:
  - Initial: 2
  - 2 – 10,000



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

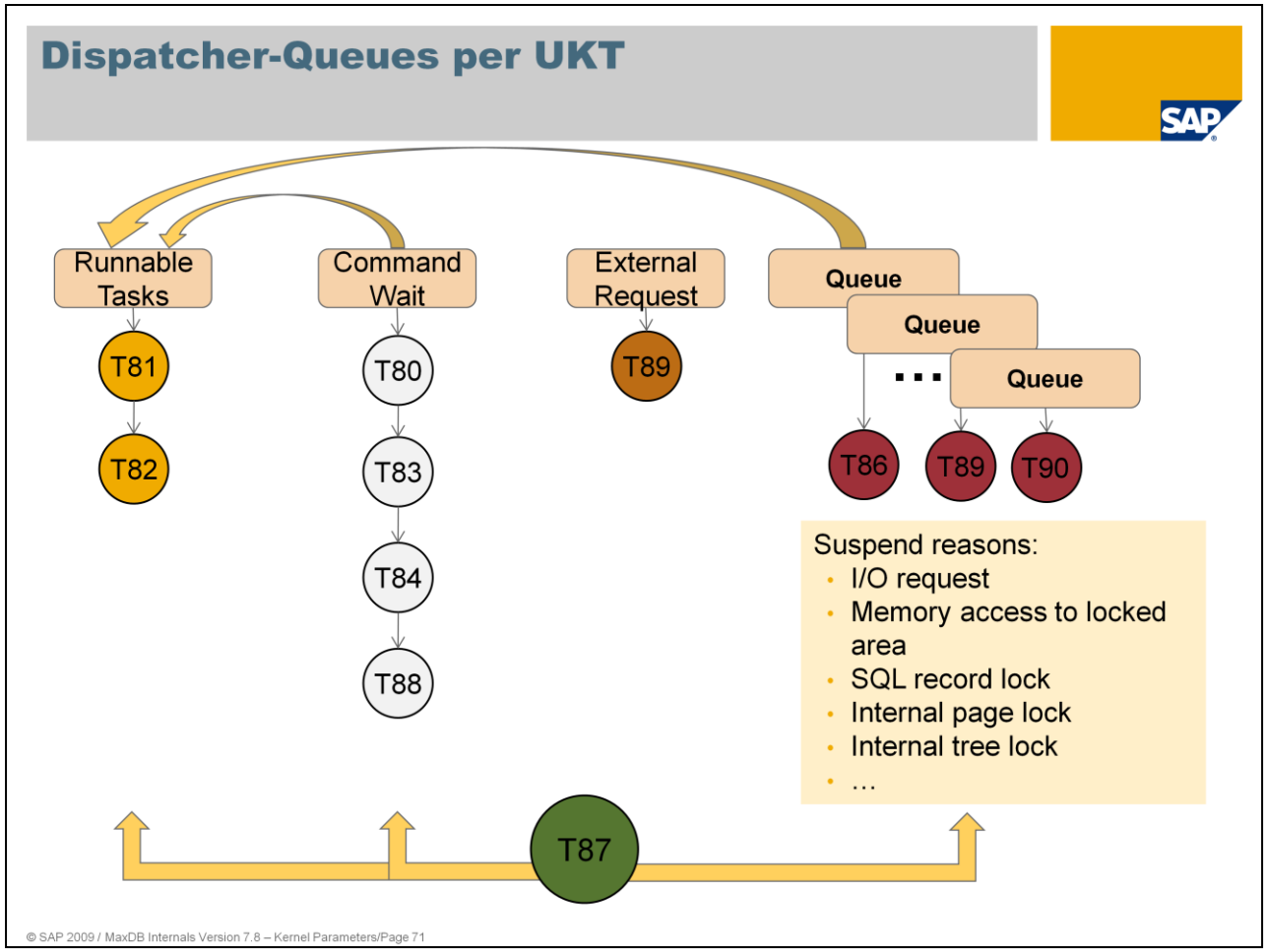
Online change: YES

Prioritization of Tasks



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.



Tasks are assigned to certain queues existing per UKT.

Runnable tasks are waiting in a specific queue. They wait because one other task is running in this UKT. Each runnable tasks has a priority. The running task increases the priority of the runnable tasks every time it passes the dispatcher code. The runnable task with the highest priority will become the next running task. Accesses to the lists and queues inside the UKT are locking free because only one task can run at one time.

Tasks waiting for a request from the database client are assigned to the command wait queue. Tasks which are suspended by any reason are waiting in a respective queue.

A task from another UKT or another thread can put the task id of a waiting task into the external request queue notifying that the waiting task can continue. With this method synchronized access are only needed on the external request queue. The accesses to all other queues are locking free. The dispatcher of a the UKT first tries a locking free dirty read to the external request queue and requests a lock if the queue is filled.

A running task releases control if

- it has finished it’s work and sends back the result to the client; i.e. it puts itself into the command wait queue
- it has to wait for another task or thread. It puts itself into a queue according to the suspend reason
- another task is runnable and the running task has exceeded the maximum number of runs through critical sections. It put’s itself to the queue of runnable tasks.

Base Priority Depending on the State

Base Priority	Previous State	Default
TaskCommunicationRequestPriority	Command Wait	Low
TaskYieldRequestPriority	Busy Loop	AboveLow
TaskResumeRequestPriority	Selbst zurückgetreten	BelowHigh
TaskLockGrantRequestPriority	Wartezustand (Suspend)	AboveHigh

Priority Level	Default
TaskRequestPriorityLow	10
TaskRequestPriorityAboveLow	30
TaskRequestPriorityBelowNormal	50
TaskRequestPriorityNormal	80
TaskRequestPriorityAboveNormal	90
TaskRequestPriorityBelowHigh	100
TaskRequestPriorityHigh	240
TaskRequestPriorityAboveHigh	400

The dispatcher assigns base priorities to the runnable tasks according to their previous state.

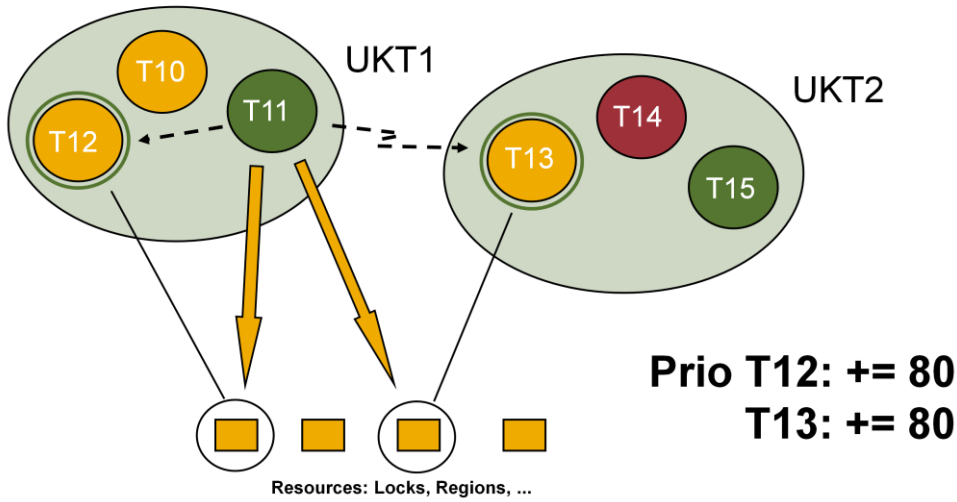
Values: Possible values or base priorities are:
Low, AboveLow, BelowNormal, Normal, AboveNormal, BelowHigh, High, AboveHigh

Online Change: YES (Base priorities)

Additional Priorities for Lockholder

TaskNiceElevationValue (_PRIO_FACTOR)

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



A task that holds 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

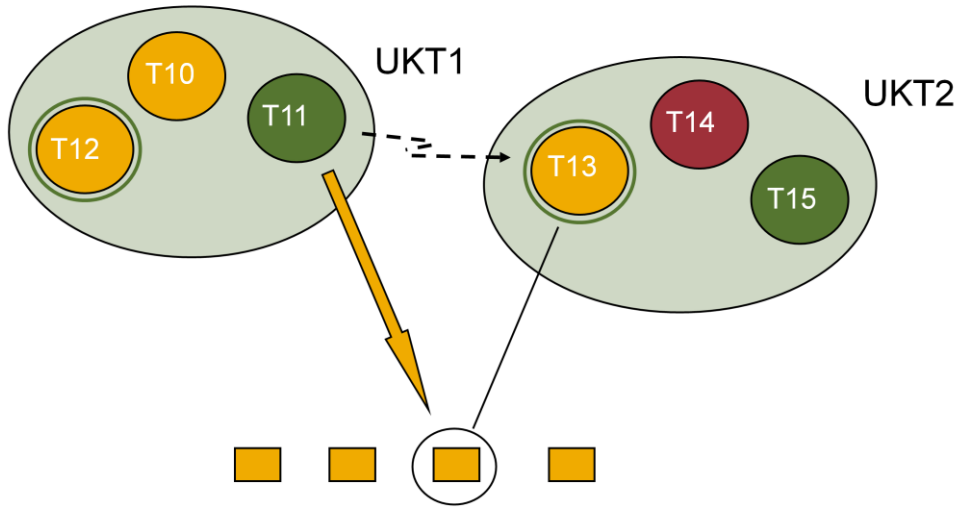
Online change: NO

Interruption of Tasks

ForceSchedulePrioritizedTask (_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 TaskPriorityFactor is set).

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

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

Online change: YES

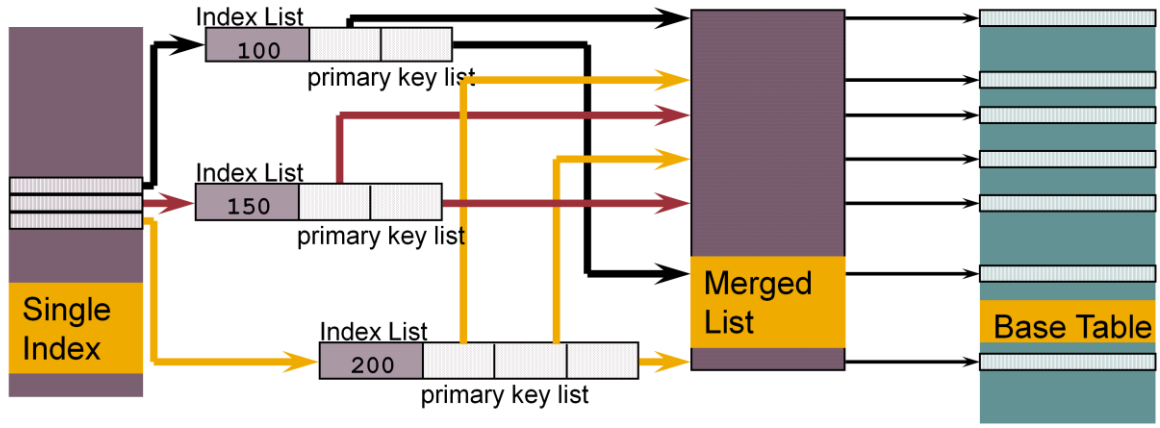
Optimizer

IndexlistsMergeThreshold	Max. number of index pages for the creation of a merge list
EnableIndexOnlyStrategy	Index Only Strategy
EnableMinMaxAggregationOptimizer	Min and max optimization
EnableAggregationOptimization	Aggregation through BD layer
EnableFetchReverseOptimization	Allow reverse index scan
UseStrategyCache	Determination of a strategy per statement
UseHashedResultset	Create aggregates via hash tables
JoinSearchLevel JoinSearchTableThreshold4 JoinSearchTableThreshold9	Join sequence algorithms dependent on the number of join tables
EnableOnePhaseJoin	Aggregation during join processing
ParallelJoinServerTasks	Parallel I/O for index transitions
EnableOuterJoinOptimization	Determination of the table order
EnableJoinHashTableOptimization HashJoinTotalMemorySize	Use of hash joins
EnableQueryRewrite	Use of Query Rewrite
UpdateStatParallelServerTask	Number of server tasks for Update Statistics
UpdateStatSampleAlgorithm	Sampling algorithm for determining statistics

Creation of Merge Lists

IndexlistsMergeThreshold (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 IndexlistsMergeThreshold, 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

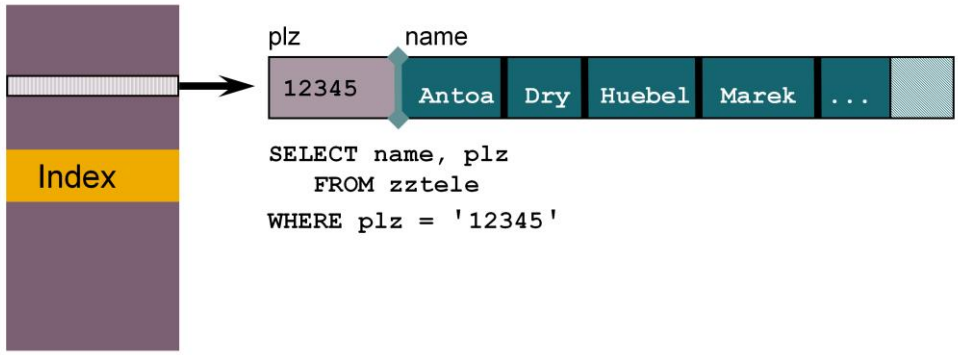
1 <= IndexlistsMergeThreshold

Online change: Yes

Index Only Strategy

EnableIndexOnlyStrategy (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 EnableIndexOnlyStrategy is set to YES, the INDEX ONLY is employed.

Values:

Default: YES

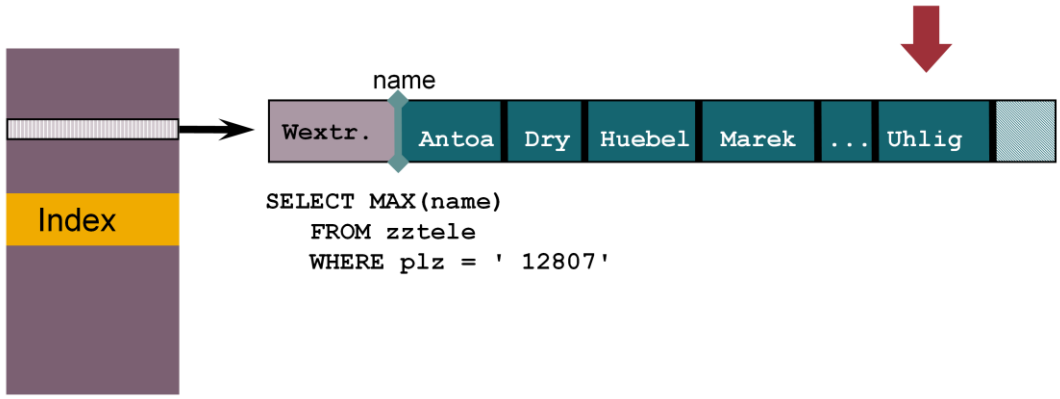
NO

Online change: YES

Min and Max Optimization

EnableMinMaxAggregationOptimizer (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

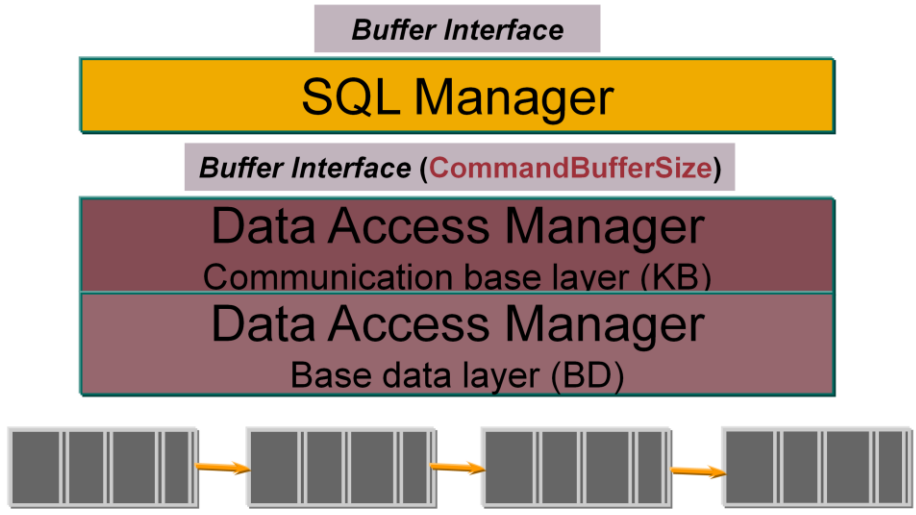
NO

Online change: YES

Aggregate Optimization

EnableAggregationOptimization (OPTIMIZE_AGGREGATION)

- With an Index Only Strategy, the BD layer calculates 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

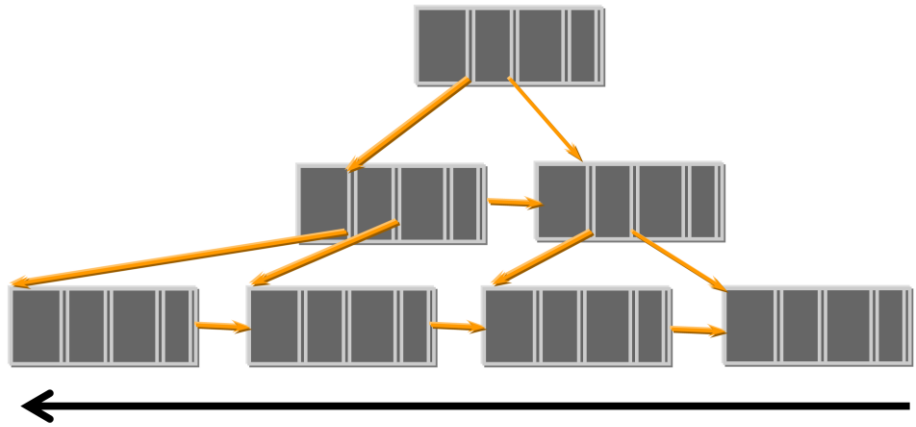
NO

Online change: Yes

Reverse Index Access

EnableFetchReverseOptimization (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 EnableFetchReverseScan=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

UseStrategyCache (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 UseStrategyCache.

```
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 UseStrategyCache.

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.

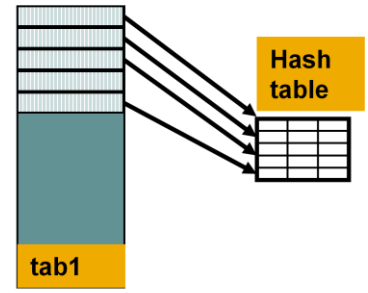
Online change: Yes

Use of Hash Joins

UseHashedResultset (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 often faster than creating a temporary B*Tree.

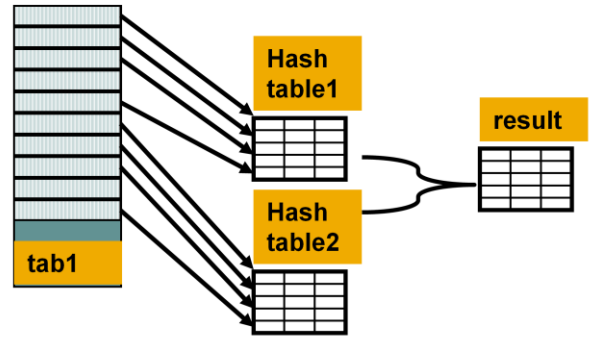
Online change: Yes

Maximum Cache Size per Hash Table

HashedResultsetCacheSize (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 HashedResultsetCacheSize 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

JoinSearchLevel (JOIN_SEARCH_LEVEL)

- Specifies the algorithm for the join sequence search to find the best order for table processing.
- Values:
  - 0: The parameters JoinSearchTableThreshold4 & JoinSearchTableThreshold9 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 JoinSearchLevel 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 JoinSearchLevel to 0.

Online change: YES

Join Algorithm Depending on Number of Tables

JoinSearchTableThreshold4 (JOIN_MAXTAB_LEVEL4)

JoinSearchTableThreshold9 (JOIN_MAXTAB_LEVEL9)

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

- If (# tables < JoinSearchTableThreshold9)
  then perform permutation
- If (# tables > JoinSearchTableThreshold4)
  then perform Greedy Algorithm
- If (JoinSearchTableThreshold9 < # tables < JoinSearchTableThreshold4)
  then perform Internal Algorithm
- JoinSearchTableThreshold9: Default = 5, Values: 2-64
- JoinSearchTableThreshold4: Default = 16, Values: 2-64

The standard settings are as follows:

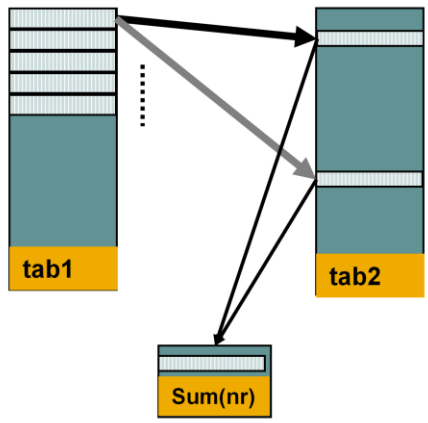
If the number of join tables

- 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 Aggregate Without Temporary Result Sets

EnableOnePhaseJoin (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

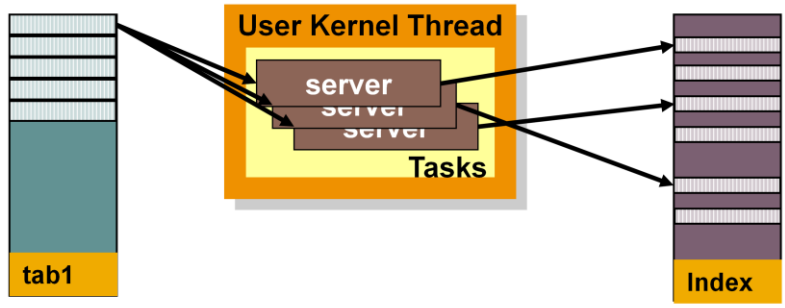
NO: Multi-phase processing

Online change: Yes

Parallel I/O for Index Transitions

ParallelJoinServerTasks (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 ParallelJoinServerTasks should by lower than the number of configured data volumes.

The parameter **ParallelJoinMinThreshold** (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

EnableOuterJoinOptimization (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'
```

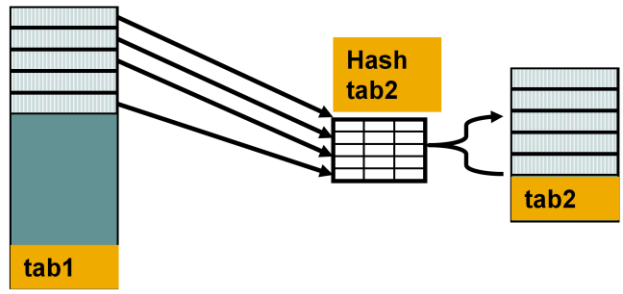

With outer joins, the result can vary depending on the processing sequence. For this reason, the parameter EnableOuterJoinOptimization allows you to specify whether the database should choose the best sequence or if the sequence is to be taken from the Select statement.

Online change: Yes

Hash Joins

EnableJoinHashTableOptimization (OPTIMIZE_JOIN_HASHTABLE)

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



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

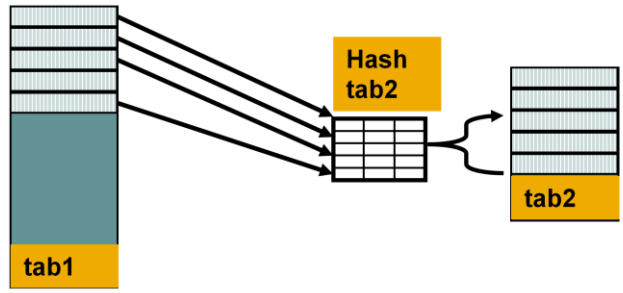
The parameter **JoinHashMinimalRatio** (OPTIMIZE_JOIN_HASH_MINIMAL_RATIO) defines the minimum ratio of output amount to the table for the next join step, after which hash joins are used. The deafult value is 1%.

Online change: Yes

Maximum Cache Size for Hash Joins

HashJoinTotalMemorySize (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

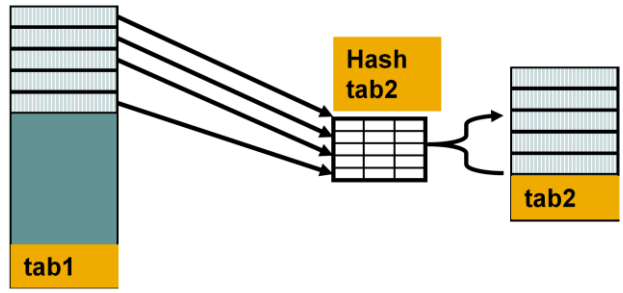


Online change: YES

Maximum Table Size for Hash Joins

HashJoinSingleTableMemorySize (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, Max: HashJoinTotalMemorySize



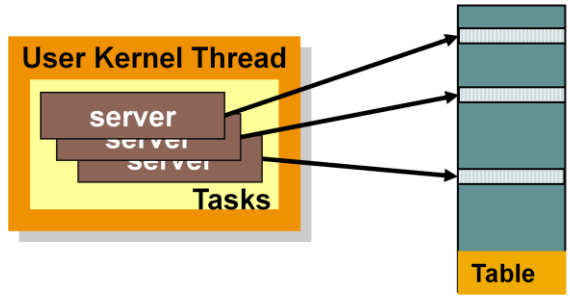
It is only sensible to generate temporary hash tables for join tables below a certain size. With large tables, you lose more time with the scan than you gain through the better search algorithm on the temporary hash table.

Online change: YES

Number of Server Tasks for Update Statistics

UpdateStatParallelServerTask (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 UpdateStatParallelServerTask to reduce the number of parallel I/O for Update Statistics.

Online change: Yes

Query Rewrite

EnableQueryRewrite (OPTIMIZE_QUERYREWRITE)

- Query Rewrite rebuilds SQL statements after the syntactical analysis by using predefined rules to allow the optimizer to create a better strategy.
- Rules can be activated/deactivated by updating table QUERYREWRITERULES

Simple example:
select a from (select a,b,c from tab)

select a from tab

As of version 7.6 Query Rewrite is activated by default.

You can completely deactivate Query Rewrite by setting the parameter EnableQueryRewrite to NO.

Table QUERYREWRITERULES contains all rules. You can switch off single rules with the following statement:

```
UPDATE QUERYREWRITERULES
SET ACTIVE = 'NO'  WHERE RULENAME = '<rulename>'
```

Online change: YES

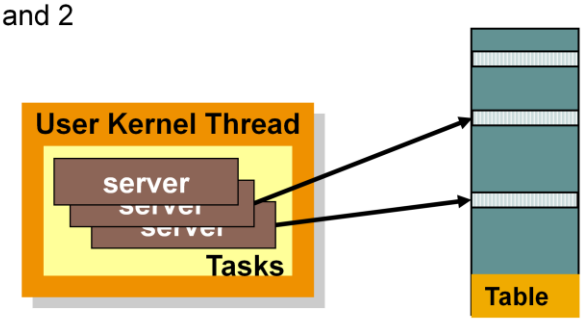
QUERYREWRITERULES in version 7.7.03:

RULENAME	ACTIVE	COMMENT
AddLocalPredicates	YES	Add Local Predicates for Joins with OR-Predicates
ConvertExistentialSubquery	NO	Convert a correlated existential subquery to an IN clause
ConvertOrToIn	YES	Convert OR to IN
ConvertToExistentialSubquery	NO	Convert INTERSECT or EXCEPT to an existential subquery
DistinctForSubqueries	YES	Set Distinct for existential and all subqueries
DistinctPullUp	YES	Remove distinct elimination in a select if all fromselects are distinct
DistinctPushDownFrom	YES	Distinct push down from
DistinctPushDownTo	YES	Distinct push down to
EliminateGroupByOrDistinct	YES	Remove unnecessary GROUP BY or DISTINCT
EliminateOrderBy	YES	Remove unnecessary ORDER BY
EliminateSubqueries	YES	EliminateSubqueries
MergeExistentialSubquery	YES	Merge a select with an existential subquery
MergeFromSelectOrView	YES	Merge a select with a fromselect or view
NormalizePredicates	NO	Normalize Predicates
OptimizeSubqueries	YES	OptimizeSubqueries
PushDownPredicates	YES	Push down predicates
PushDownProjection	YES	Push down projection
PushDownQuantifier	NO	Push down quantifier
RemoveConstFromGroupOrOrderBy	YES	Remove unnecessary constants from GROUP BY or ORDER BY
SimplifyPredicates	YES	Simplify Predicates

Algorithm for Update Statistics

UpdateStatSampleAlgorithm (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.

Query Rewrite

EnableQueryRewrite (OPTIMIZE_QUERYREWRITE)

- Query Rewrite transforms SQL Statement after a syntax analysis according to predefined rules. It simplifies the optimization and/or the execution of the command.
- Updates on the table QUERYREWRITERULES activate or deactivate certain rules.

Example:
select a from (select a,b,c from tab)
⇨ **select a from tab**

You can turn of Query Rewrite with the parameter EnableQueryRewrite.

The table QUERYREWRITERULES contains all rules. The following command disables a certain rule:

```
UPDATE QUERYREWRITERULES
SET ACTIVE = 'NO'  WHERE RULENAME = '<rulename>'
```

RULENAME	ACTIVE	COMMENT
AddLocalPredicates	YES	Add local predicates
CombineExternalSelects	YES	Combine external selects
CombineToAnyOrAll	YES	Combine ORed/ANDed predicates to ANY/ALL predicate
ConvertExceptToAntiSemiJoin	YES	Convert Except to anti-semi-join
ConvertIntersectToSemiJoin	YES	Convert Intersect to semi-join
ConvertUnionToORQuery	YES	Convert Union to OR-query
DistinctPullUp	YES	Pull up distinct information
DistinctPushDown	YES	Push down distinct information
FlattenSubqueries	YES	Flatten subqueries
MergeQueries	YES	Merge queries
OptimizeAggregates	YES	Optimize aggregates
OptimizeExpressions	YES	Optimize expressions
OptimizeJoins	YES	Optimize joins
OptimizePredicates	YES	Optimize predicates
OptimizeSubqueries	YES	Optimize subqueries
PushDownJoins	YES	Push down joins
PushDownPredicates	YES	Push down predicates
PushDownProjection	YES	Push down projection
RemoveDispensableConstants	YES	Remove dispensable constants
RemoveDispensableGroupBy	YES	Remove dispensable GroupBy-columns
RemoveDispensableOrderBy	YES	Remove dispensable OrderBy-columns
ReorderJoins	YES	Reorder joins
ReorderPredicates	YES	Reorder predicates
ReorderUnions	YES	Reorder unions
SubstituteBushyJoins	YES	Substitute right side of a bushy join by a from-select
SubstituteViews	YES	Substitute complex views and join views by the corresponding from-select

Online change: Yes

Thank you!