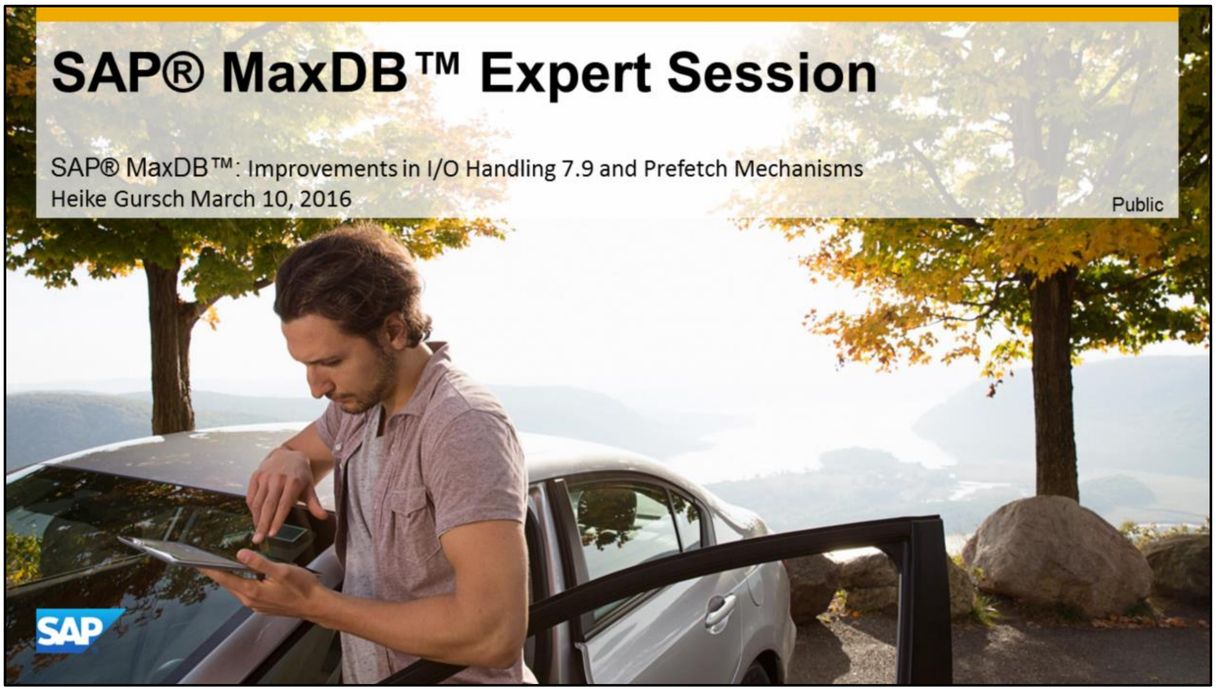


SAP® MaxDB™ Expert Session

SAP® MaxDB™: Improvements in I/O Handling 7.9 and Prefetch Mechanisms

Heike Gursch March 10, 2016

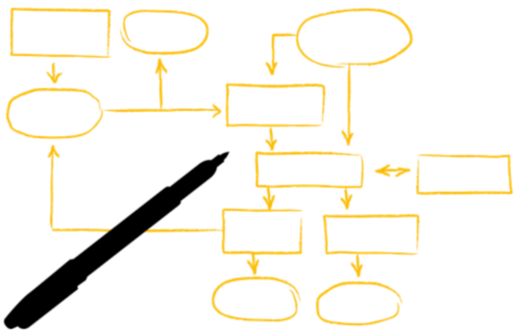
Public

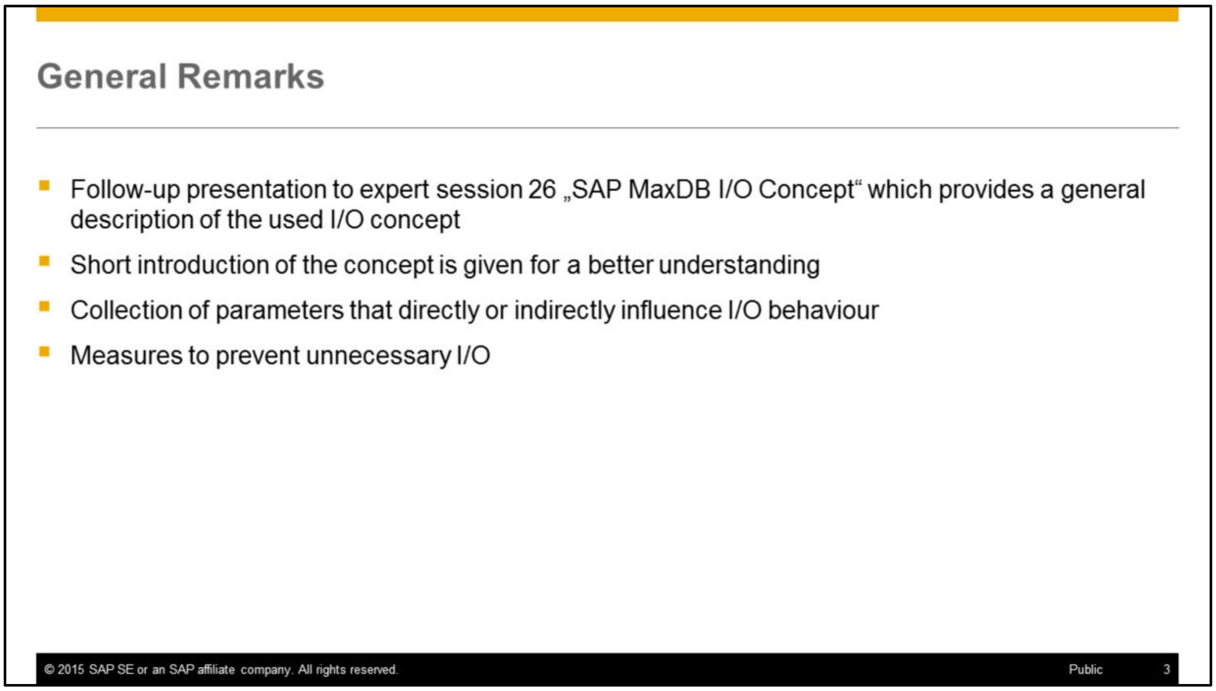
SAP® MaxDB™ Expert Session

Improvements in I/O Handling 7.9 and Prefetch Mechanisms

Public

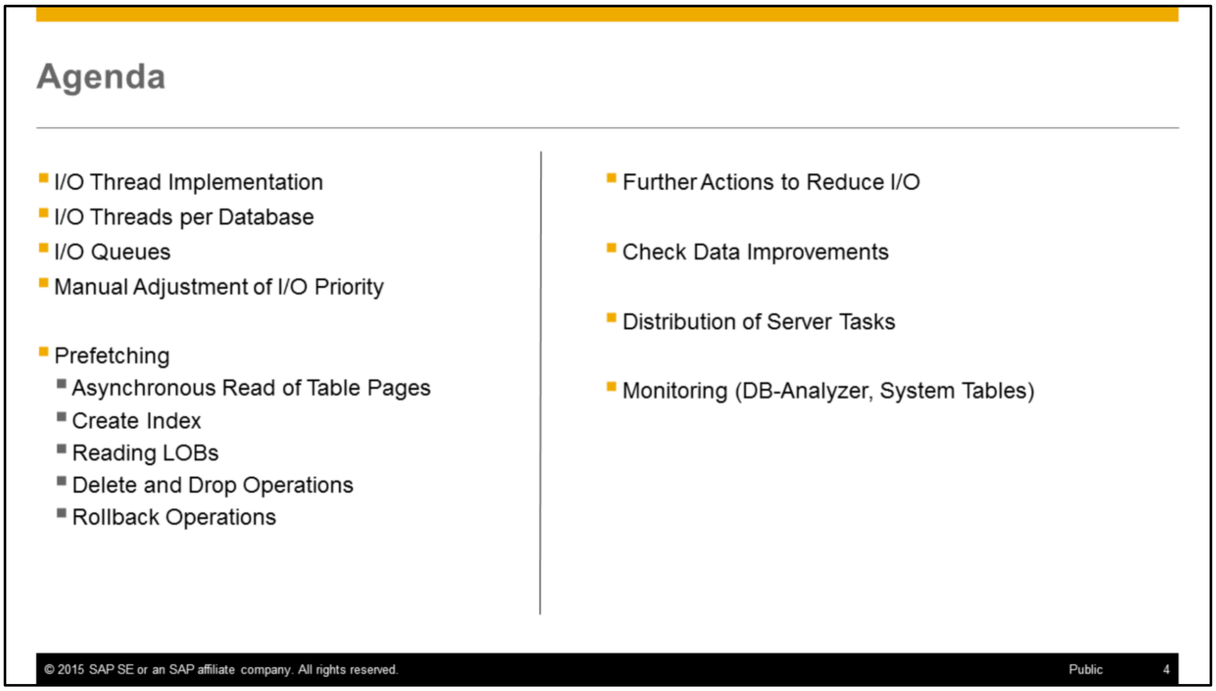
Heike Gursch
Christiane Hienger
IMS MaxDB/liveCache Development Support
March 10, 2016





General Remarks

- Follow-up presentation to expert session 26 „SAP MaxDB I/O Concept“ which provides a general description of the used I/O concept
- Short introduction of the concept is given for a better understanding
- Collection of parameters that directly or indirectly influence I/O behaviour
- Measures to prevent unnecessary I/O



Agenda

- I/O Thread Implementation
- I/O Threads per Database
- I/O Queues
- Manual Adjustment of I/O Priority

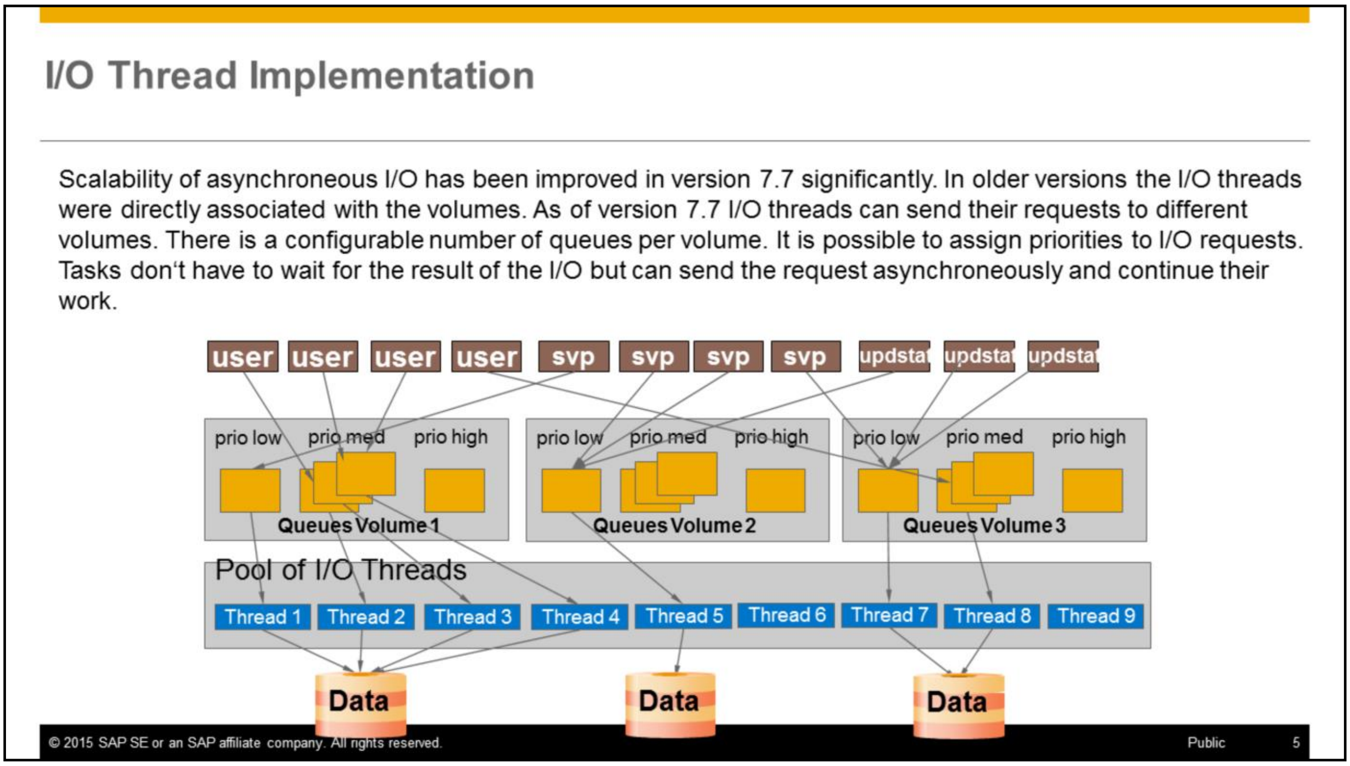
- Prefetching
  - Asynchronous Read of Table Pages
  - Create Index
  - Reading LOBs
  - Delete and Drop Operations
  - Rollback Operations

- Further Actions to Reduce I/O

- Check Data Improvements

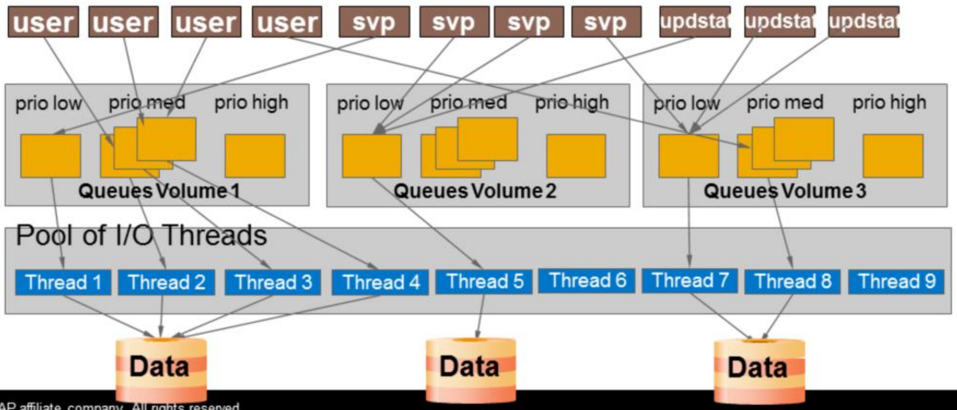
- Distribution of Server Tasks

- Monitoring (DB-Analyzer, System Tables)



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.



Public

With version 7.7 the I/O interface to the operating system has been reimplemented. As of version 7.7 different parameters than in version 7.6 are used. The improved I/O system has the following essential advantages:

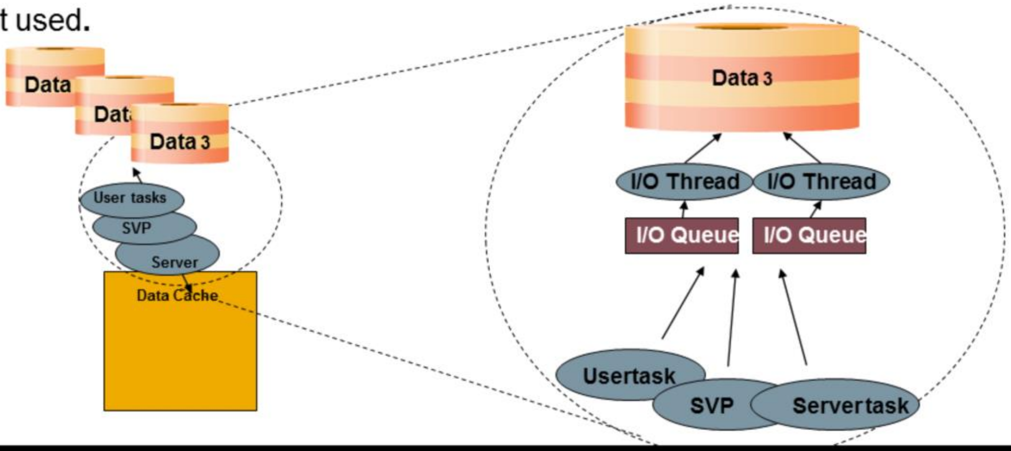
- No direct assignment of a I/O worker thread to a volume. This implies a better scalability of I/O.
- I/O worker threads can be started on request. This prevents the use of unnecessary resources.
- The synchronization of accesses to the I/O queues has been changed. The access is done collision free. 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.

The former pager tasks have been replaced in 7.9 by the special savepoint tasks.

I/O Threads per Database

EnablePreAllocateIOWorker
MinIOPoolWorkers, MaxIOPoolWorkers
IOPoolIdleTimeout, IOWorkerStackSize

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



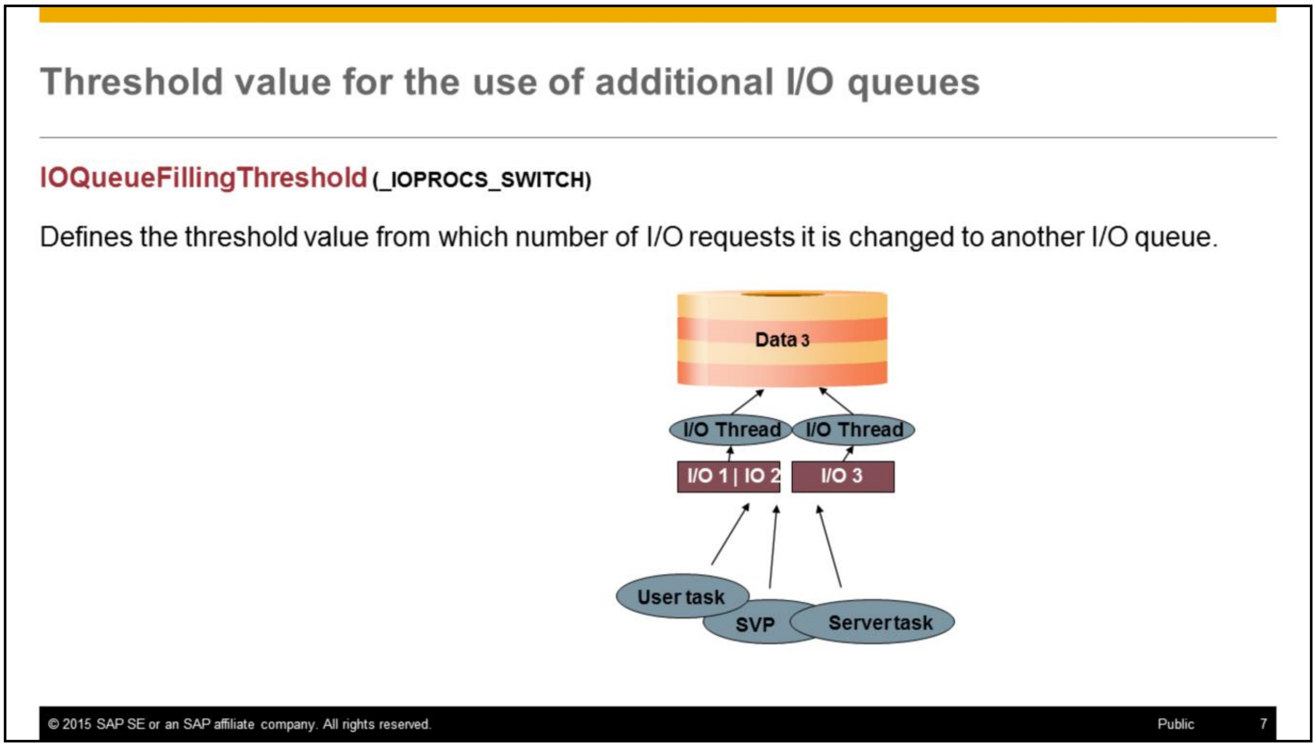
Usually it is not necessary to adapt these parameters. The database can start additional I/O worker threads on request.
The parameter **EnablePreAllocateIOWorker** defines if I/O worker threads are already generated during startup phase. As a default it is set to NO meaning that threads are only started when needed. This is usually more effective. Be aware that if the configuration in near machine resource limits it may happen that I/O worker thread resources are not available during runtime. F.e. this might prevent the execution of a successful backup.

MinIOPoolWorkers defines the minimum number of I/O worker threads that were allocated during the startup phase. If the parameter is set to a value smaller than the number of priorities, then at least as many workers are started as priorities are defined.
With setting the parameter **MaxIOPoolWorkers** it is possible to restrict the number of I/O worker threads.

(The value for **MaxIOPoolWorkers** is identical to **MinIOPoolWorkers** if **EnablePreAllocateIOWorker** is set to YES.)

IOPoolIdleTimeout describes the maximum time in seconds an I/O pool worker is allowed to be idle before it is released and the thread resources are returned to the operating system.
IOWorkerStackSize specifies the stack size for I/O worker threads in kilobytes which is as default the platform-specific minimum stack size.

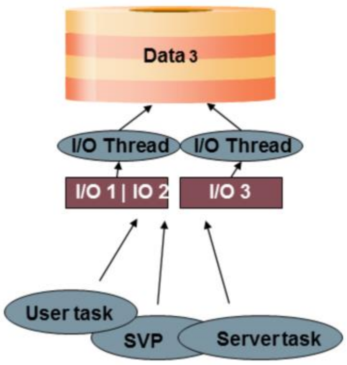
(The parameters shown above were introduced with the implementation of a multiple database concept. This allows the use of several MaxDB databases within one instance. The parameters can be used to restrict I/O resources per database within one instance. This concept did not come into effect so far.)



Threshold value for the use of additional I/O queues

IOQueueFillingThreshold (_IOPROCS_SWITCH)

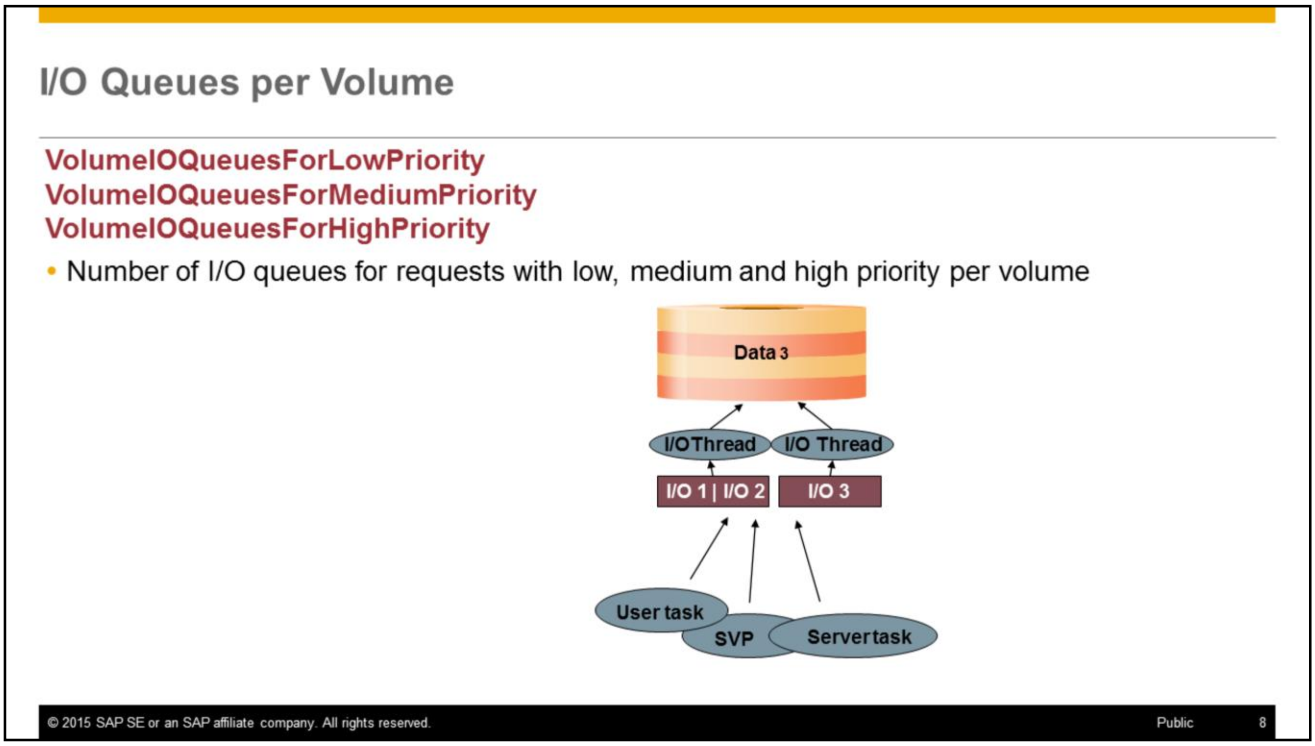
Defines the threshold value from which number of I/O requests it is changed to another I/O queue.



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

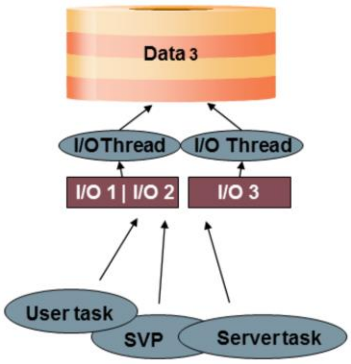
If there are not enough I/O worker threads available to handle all filled queues then the system automatically starts an additional thread.



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 influence 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

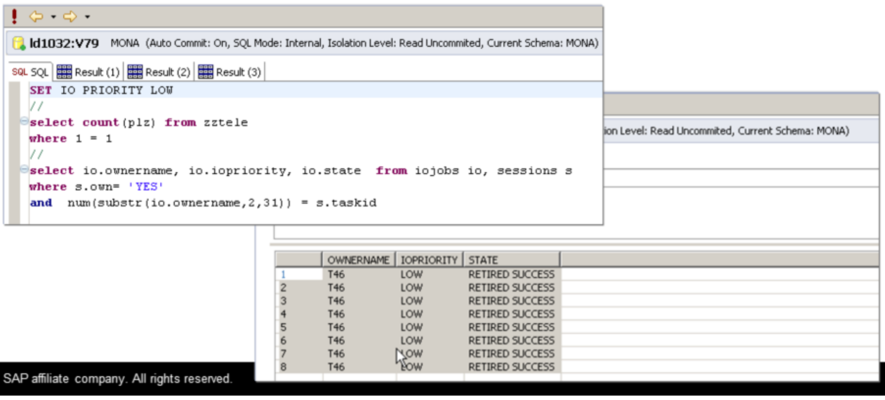
You can watch the states of current I/O requests in the system by the use of the console (x_cons) and the system view IOJOBS.

Manual Adjustment of I/O Priority

A session can modify the priority for its own disk I/O orders to Low and Medium

The I/O Subsystem supports the priorities LOW, MEDIUM, HIGH

Syntax: SET IO PRIORITY   LOW | DEFAULT



ld1032:V79   MONA (Auto Commit: On, SQL Mode: Internal, Isolation Level: Read Uncommited, Current Schema: MONA)

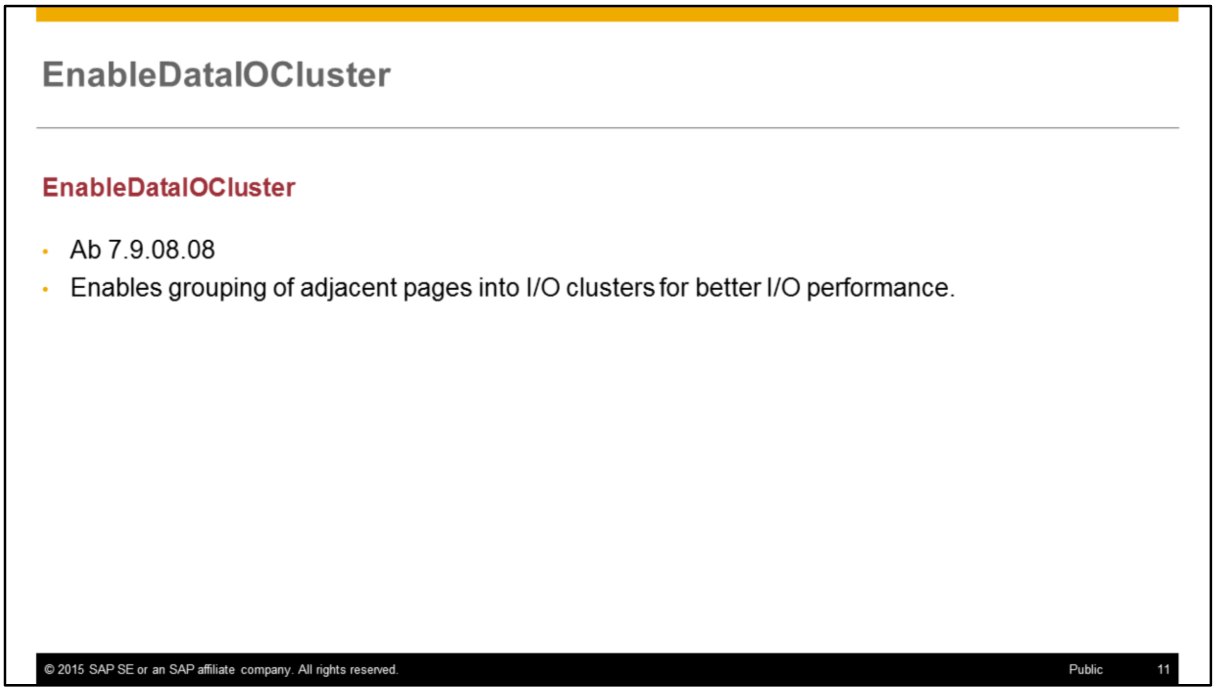
```
SET IO PRIORITY LOW
//
select count(plz) from zztele
where 1 = 1
//
select io.ownername, io.iopriority, io.state  from iojobs io, sessions s
where s.own= 'YES'
and  num(substr(io.ownername,2,31)) = s.taskid
```

	OWNERNAME	IOPRIORITY	STATE
1	T46	LOW	RETIRED SUCCESS
2	T46	LOW	RETIRED SUCCESS
3	T46	LOW	RETIRED SUCCESS
4	T46	LOW	RETIRED SUCCESS
5	T46	LOW	RETIRED SUCCESS
6	T46	LOW	RETIRED SUCCESS
7	T46	LOW	RETIRED SUCCESS
8	T46	LOW	RETIRED SUCCESS

The statement can be used to control the I/O behaviour of the affected task. If no TASK ID is given the current task shall be prioritized.
The I/O priority can be reduced to LOW and raised to MEDIUM but not to HIGH. Server tasks inherit the I/O priority from the client user task.

```
SET IO PRIORITY LOW
//
select count(str) from zztele
where 1 = 1
//
select io.ownername, io.iopriority, io.state from iojobs io, sessions s
where s.own= 'YES‘
and  num(substr(io.ownername,2,31)) = s.taskid
```

http://pts:1080/webpts?wptsdetail=yes&ErrorType=1&ErrorID=1175203



EnableDataIOCluster

EnableDataIOCluster

- Ab 7.9.08.08
- Enables grouping of adjacent pages into I/O clusters for better I/O performance.

© 2015 SAP SE or an SAP affiliate company. All rights reserved. Public 11

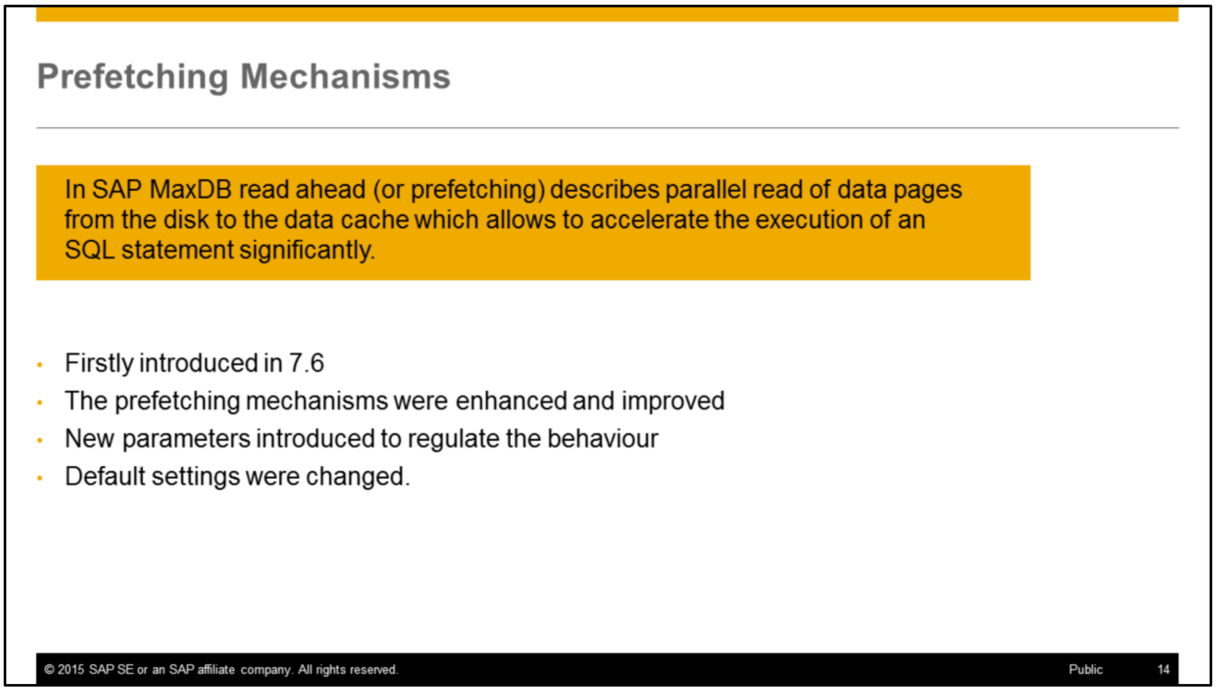
The parameter **EnableDataIOCluster** allows to cluster adjacent pages and thus improves I/O performance by processing larger units.

The kernel parameter **EnableDataIOCluster** is set to NO for liveCache instances.

Set this parameter to NO if you experience long running savepoints , especially when one CPU is used to 100% and there is no I/O. When this parameter is set to NO, the savepoint will not attempt to write data pages in cluster, which might lead to worse I/O performance when this data is read back in.

Online Change: YES

"YES" enables I/O clustering
"NO" prevents I/O clustering



Prefetching Mechanisms

In SAP MaxDB read ahead (or prefetching) describes parallel read of data pages from the disk to the data cache which allows to accelerate the execution of an SQL statement significantly.

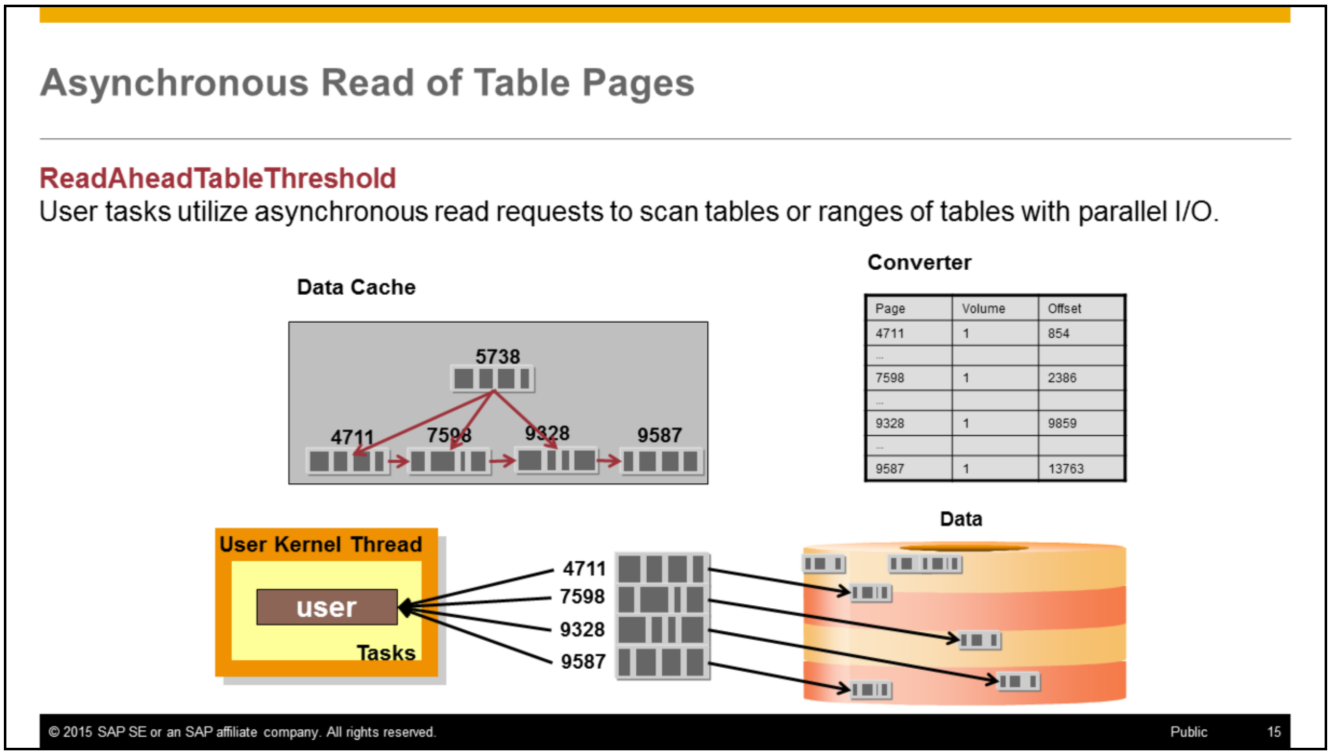
- Firstly introduced in 7.6
- The prefetching mechanisms were enhanced and improved
- New parameters introduced to regulate the behaviour
- Default settings were changed.

© 2015 SAP SE or an SAP affiliate company. All rights reserved. Public 14

The first prefetch mechanisms were introduced with version 7.6. As some 7.7 versions were developed at the same time, there are some 7.7 versions which do not support prefetching.

Further information can be found in the corresponding FAQ note:
1327874 - FAQ: SAP MaxDB Read Ahead/Prefetch

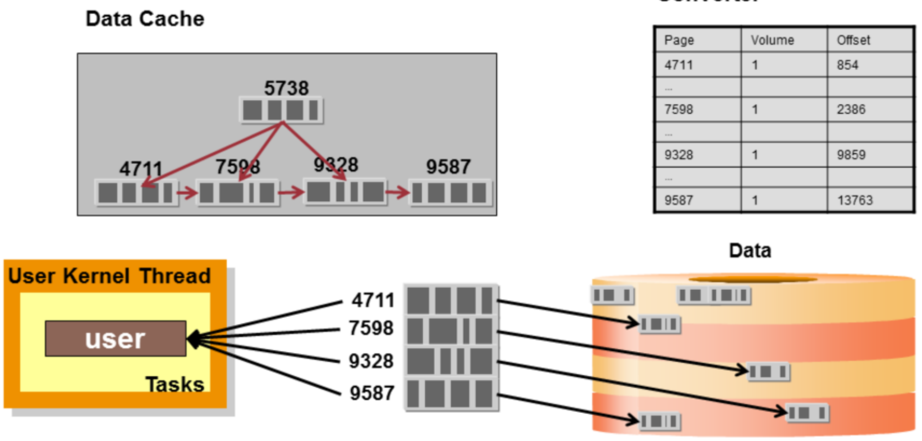
The following descriptions are related to 7.8 and 7.9. As the functionality is continueously enhanced you have to check in older patches of these versions to which extent prefetching can be used.



Asynchronous Read of Table Pages

ReadAheadTableThreshold

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



Converter

Page	Volume	Offset
4711	1	854
...		
7598	1	2386
...		
9328	1	9859
...		
9587	1	13763

As of version 7.8 MaxDB uses parallel I/O requests 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 asynchronous 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 asynchronous I/O requests if the size of the scan exceeds 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 asynchronous 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 asynchronous parallel I/O.

Asynchronous I/O read requests have the priority low.

Values:

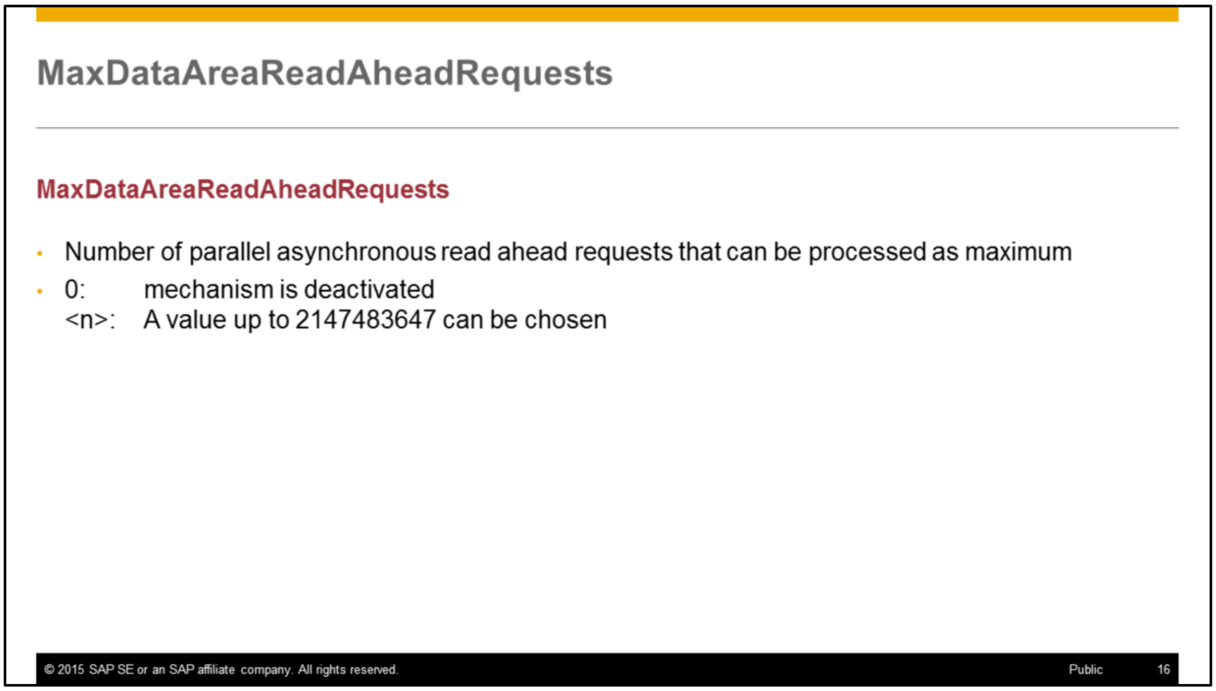
ReadAheadTableThreshold:

Default: 100 Pages
Min: 0
Max: 2147483647
Online Change: Yes

MaxDataAreaReadAheadRequests:

Default: 2 x Number of Data Volumes
Min: 0
Max: 2147483647
Online Change: Yes

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



MaxDataAreaReadAheadRequests

MaxDataAreaReadAheadRequests

- Number of parallel asynchronous read ahead requests that can be processed as maximum
- 0: mechanism is deactivated
  <n>: A value up to 2147483647 can be chosen

© 2015 SAP SE or an SAP affiliate company. All rights reserved. Public 16

This parameter is used to control the number of parallel and asynchronously processed read ahead (prefetch) requests.Typically read ahead is used during range access and complete scan. This parameter will influence I/O behaviour and main memory consumption.

A huge number of requests may harm the I/O performance if the storage system is not efficient enough. A huge number of requests will also consume data cache memory. The effects can be monitored with the suitable system views.

If **MaxDataAreaReadAheadRequests** is zero the feature is disabled.

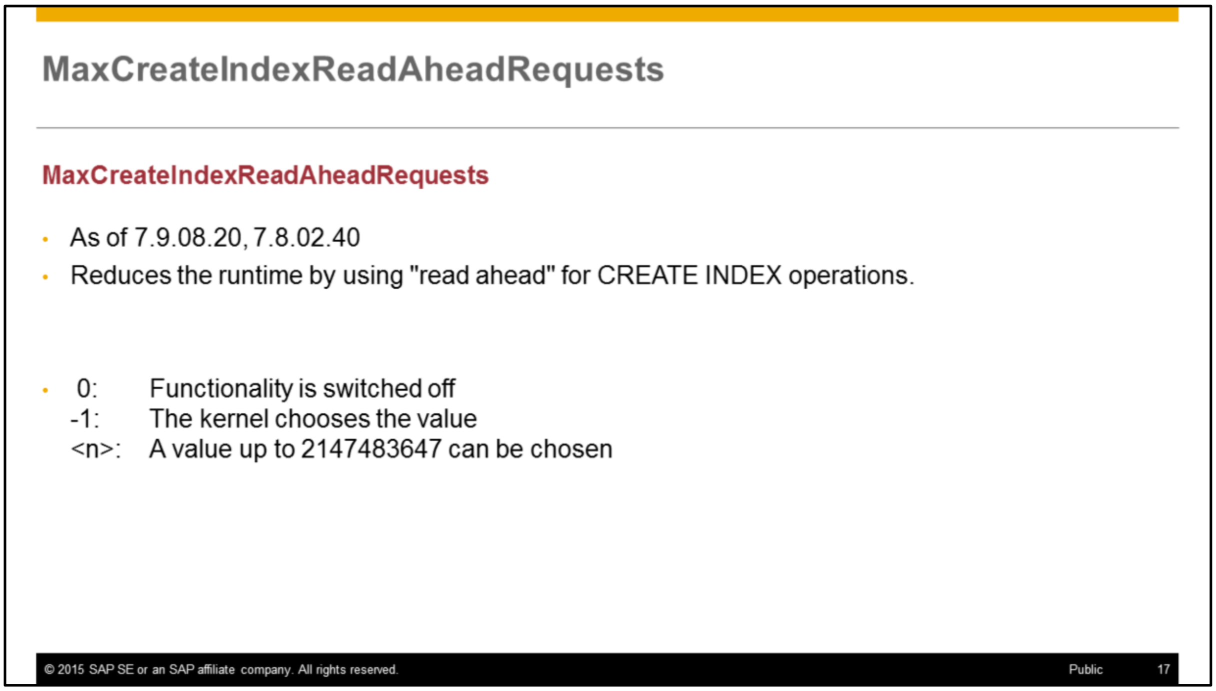
The performance of table scans using read ahead was improved by minimizing the number of page faults caused by the user task. Instead of reacting to a read I/O by the user task, now 'read ahead' is triggered when skipping to the next level-1 node.

Online Change: YES

The lower and upper limits are:
    0 <= **MaxDataAreaReadAheadRequests** <= 2147483647

10 is set as default value but will be overruled by the formula: data volumes * 2

Attention: If the parameter is set to 0, the whole read ahead mechanism is deactivated!



MaxCreateIndexReadAheadRequests

MaxCreateIndexReadAheadRequests

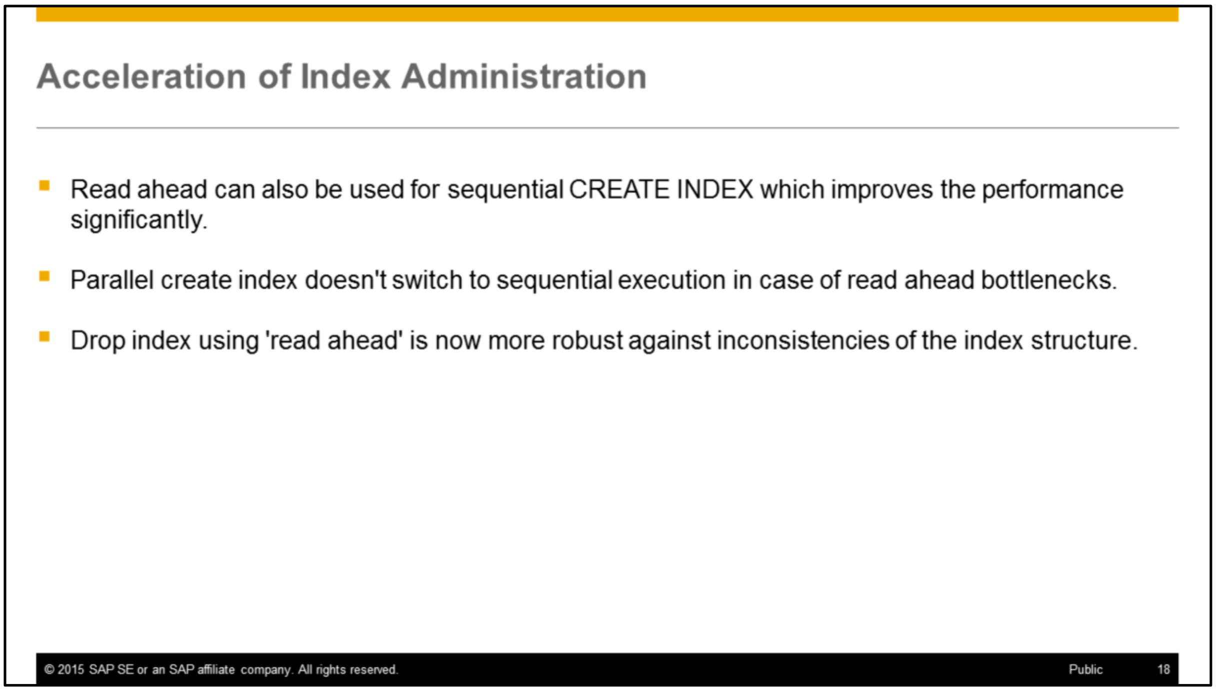
- As of 7.9.08.20, 7.8.02.40
- Reduces the runtime by using "read ahead" for CREATE INDEX operations.

0:	Functionality is switched off
-1:	The kernel chooses the value
<n>:	A value up to 2147483647 can be chosen

Public 17

This parameter **MaxCreateIndexReadAheadRequests** is used to control the number of parallel and asynchronously processed read ahead (prefetch) requests produced by create index operations. This parameter will influence I/O behaviour and main memory consumption. A huge number of requests may harm the I/O performance if the storage system is not efficient enough. A huge number of requests will also consume main memory.

The effects can be monitored by checking the DB-Analyzer logfiles or directly with the suitable system views.



Acceleration of Index Administration

- Read ahead can also be used for sequential CREATE INDEX which improves the performance significantly.
- Parallel create index doesn't switch to sequential execution in case of read ahead bottlenecks.
- Drop index using 'read ahead' is now more robust against inconsistencies of the index structure.

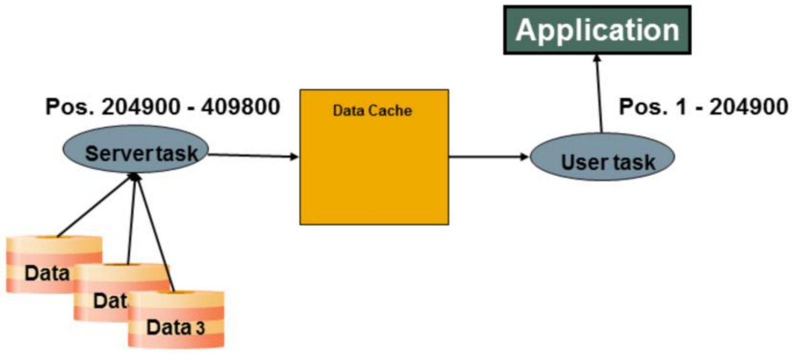
Public 18

If no **DataCache_ReadAsyncIOContext** is available, create index switches to sequential mode. This should be avoided.

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.



The prefetch mechanism for LOBs is operated with the additional parameter **ReadAheadLobThreshold.**

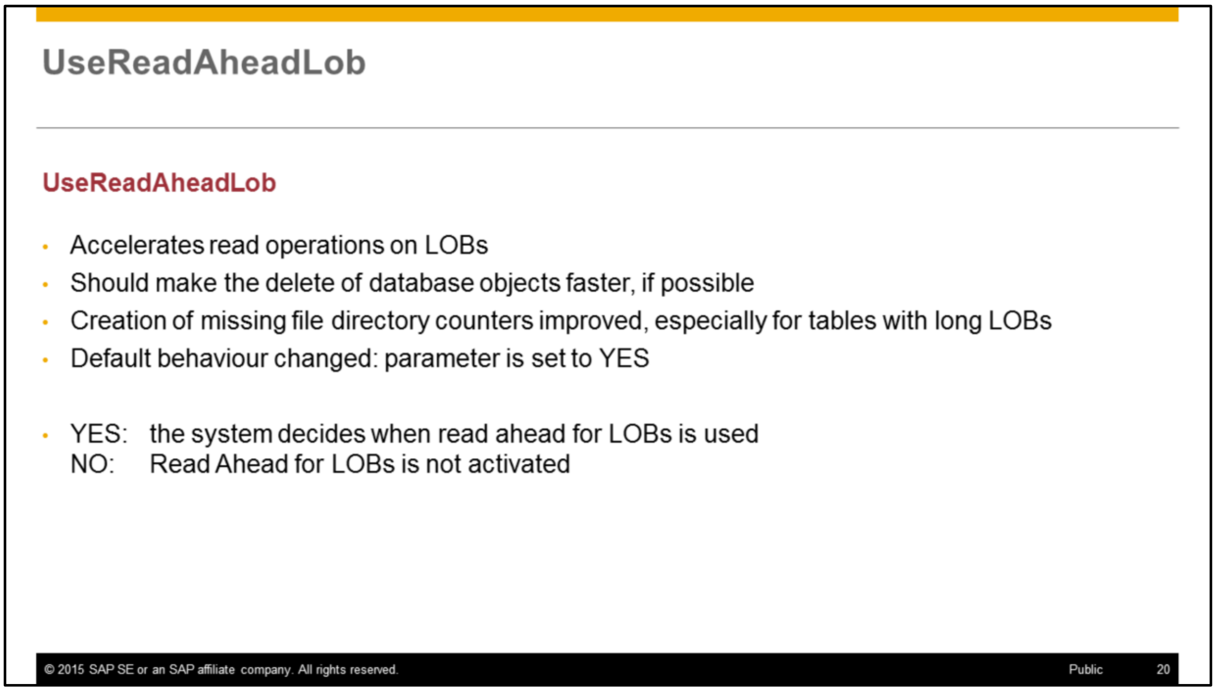
When reading larger LOB values, it is a good idea to continue reading asynchronously from the volumes while sending a read package to the application. This is done by a server task if the length of the LOB value to be read exceeds the value of the parameter **ReadAheadLobThreshold**.

Values:

Default: 25

Min: 2 * CommandBufferSize / 8196   Max: 262144

Online change: YES



UseReadAheadLob

UseReadAheadLob

- Accelerates read operations on LOBs
- Should make the delete of database objects faster, if possible
- Creation of missing file directory counters improved, especially for tables with long LOBs
- Default behaviour changed: parameter is set to YES

- YES: the system decides when read ahead for LOBs is used
  NO: Read Ahead for LOBs is not activated

© 2015 SAP SE or an SAP affiliate company. All rights reserved. Public 20

This parameter defines whether read operations of LOBs are accelerated by reading them in advance asynchronously. A prefetching mechanism is used for deletion of LOB leaf pages if the parameter is set to YES.

In current versions 'Read Ahead' of LOBs does not only read the first four LOB pages per LOB (as it was before) but uses 'read ahead' for the complete LOB.

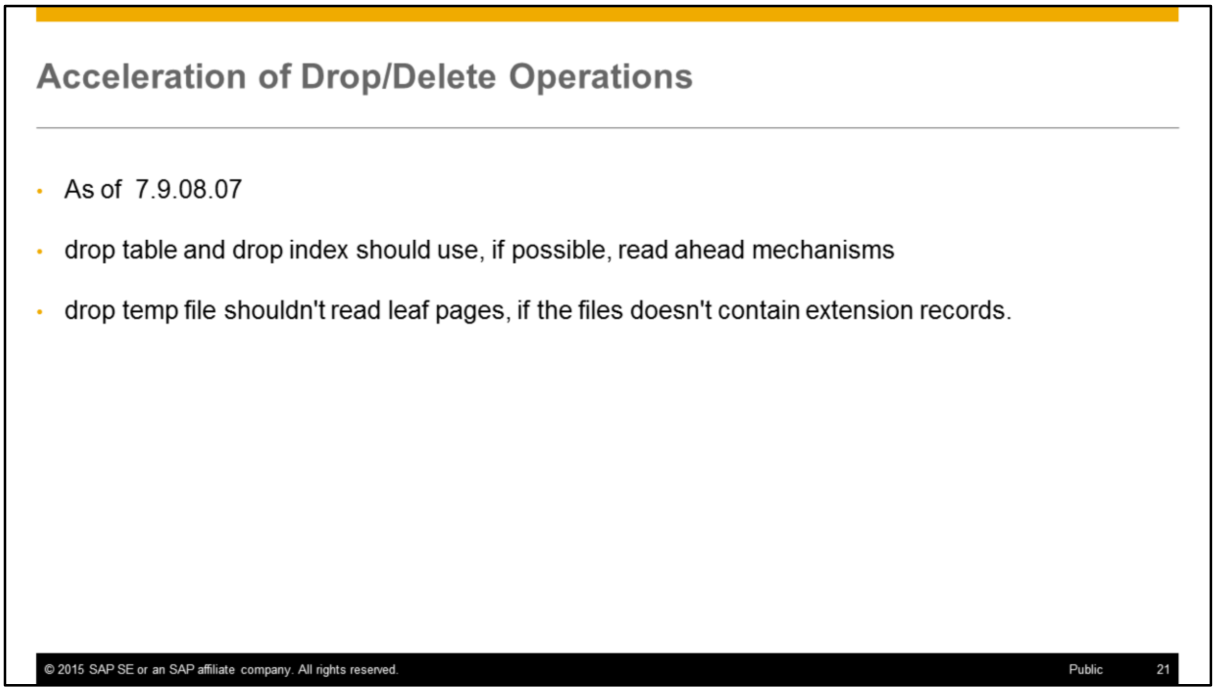
The performance of DROP LOB was improved by

1. Using 'Read Ahead' depending on the parameter **UseReadAheadLob**
2. Avoiding file directory updates if LOBs are dropped in the course of a DROP TABLE statement.

The update of file directory counters has been accelerated for tables having long LOB columns through read ahead mechanism and by just reading the root of LOBs.

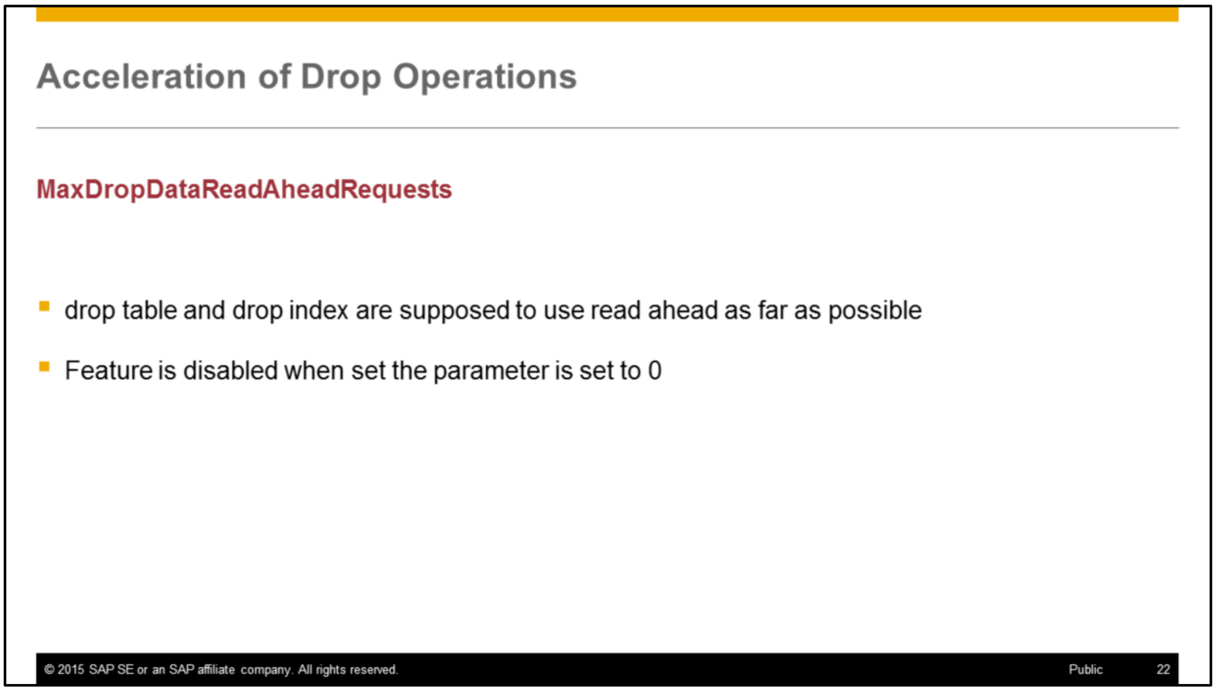
Online Change: YES

NO: read ahead of LOBs never takes place
YES: the system decides when to use read ahead for LOBs



Acceleration of Drop/Delete Operations

- As of 7.9.08.07
- drop table and drop index should use, if possible, read ahead mechanisms
- drop temp file shouldn't read leaf pages, if the files doesn't contain extension records.



Acceleration of Drop Operations

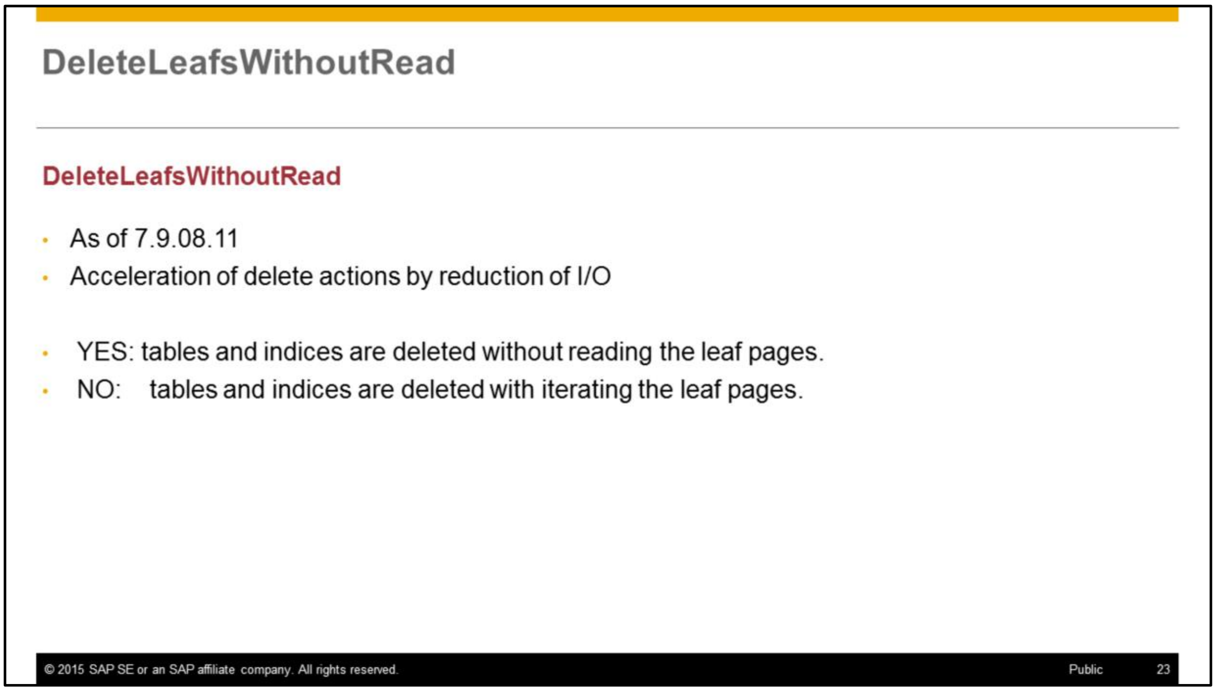
MaxDropDataReadAheadRequests

- drop table and drop index are supposed to use read ahead as far as possible
- Feature is disabled when set the parameter is set to 0

© 2015 SAP SE or an SAP affiliate company. All rights reserved. Public 22

This parameter is used to control the number of parallel and asynchronous processed read ahead (prefetch) requests produced by drop operations. This parameter will influence I/O behaviour and main memory consumption. A huge number of requests may harm the I/O performance if the storage system is not efficient enough. A huge number of requests will also consume main memory. The effects can be monitored with the suitable system views.

If **MaxDropDataReadAheadRequests** is zero the feature is disabled



DeleteLeafsWithoutRead

DeleteLeafsWithoutRead

- As of 7.9.08.11
- Acceleration of delete actions by reduction of I/O

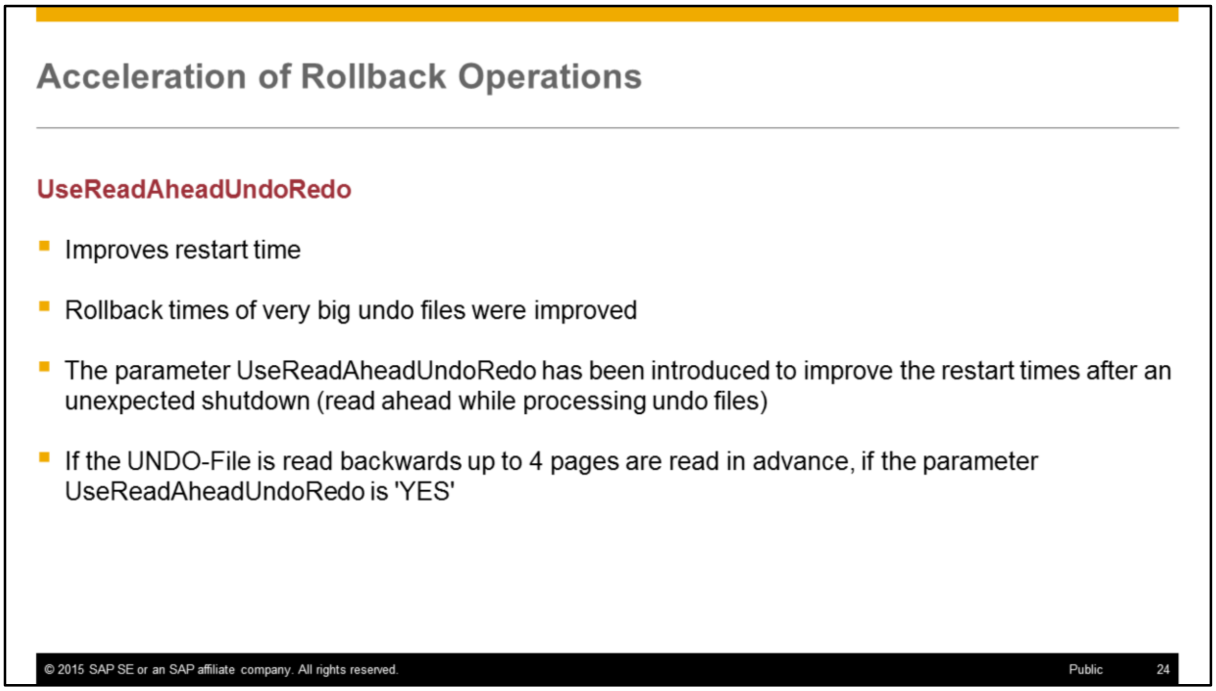
- YES: tables and indices are deleted without reading the leaf pages.
- NO: tables and indices are deleted with iterating the leaf pages.

© 2015 SAP SE or an SAP affiliate company. All rights reserved. Public 23

Online change: yes

Dropping tables via the garbage collector caused by TRUNC TABLE or unqualified delete are executed without reading the leaf pages, if the following is true

1. **DeleteLeafsWithoutRead** = 'YES'
2. The table doesn't contain any LOB columns
3. The row definiton length doesn't exceed one page



Acceleration of Rollback Operations

UseReadAheadUndoRedo

- Improves restart time
- Rollback times of very big undo files were improved
- The parameter UseReadAheadUndoRedo has been introduced to improve the restart times after an unexpected shutdown (read ahead while processing undo files)
- If the UNDO-File is read backwards up to 4 pages are read in advance, if the parameter UseReadAheadUndoRedo is 'YES'

© 2015 SAP SE or an SAP affiliate company. All rights reserved. Public 24

Info from http://pts:1080/webpts?wptsdetail=yes&ErrorType=1&ErrorID=1253445

1. Read ahead of insert, update, delete via server tasks

2. Detect rollback of 'insert select' and enable page wise read ahead in this case. Read ahead is done on tree level, i.e. several leaf pages are read asynchronously in advance from the right side. The following Undo (delete operations) will find the relevant data in the cache, then.

3. Fixed bug : the size and state of undo files may have been lost, if the kernel crashed or was stopped via 'db_stop'. Therefore at restart time even huge undo files have not been dropped via a garbage collector. This caused too long restart runtimes. The size and state now can be determined, even if normal kernel shutdown has not been reached.

4. New rows ("Rollback Read Ahead Counter" and "Rollback Read Ahead Key Counter") have been added to SYSDBA.MONITOR.

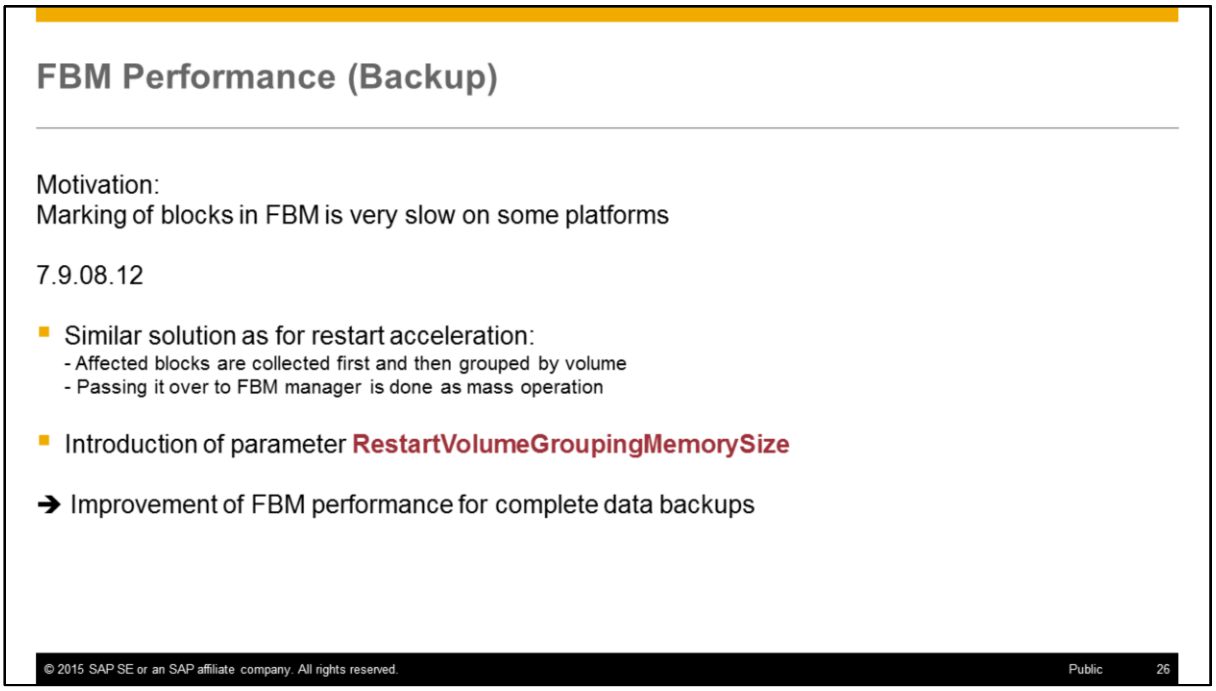
5. Progress messages for long running undo operations using read ahead are written to the KnlMsg file.

To which amount the read ahead mechanism can be used?

Read ahead on the undo file itself is implemented. When reading backwards, 4 pages are read asychronously at the same time. More than 4 pages did not fit on the existing page structure.

Read ahead for undo operations is implemented, too. When an undo file is traversed the resulting operations (Insert, Update, Delete) are not executed immediately but a server task is instructed to look for the corresponding key which implies that the data is read to the cache. The undo operation itself is started with delay because then the data is already expected in the cache. During this time delay there is no waiting situation as the same processing is already started for the next undo entry.

The read ahead mechanism is not suited if there is no I/O. So it is only started if there is I/O at all.


FBM Performance (Backup)

Motivation:
Marking of blocks in FBM is very slow on some platforms

7.9.08.12

- Similar solution as for restart acceleration:
  - Affected blocks are collected first and then grouped by volume
  - Passing it over to FBM manager is done as mass operation
- Introduction of parameter **RestartVolumeGroupingMemorySize**

➔ Improvement of FBM performance for complete data backups

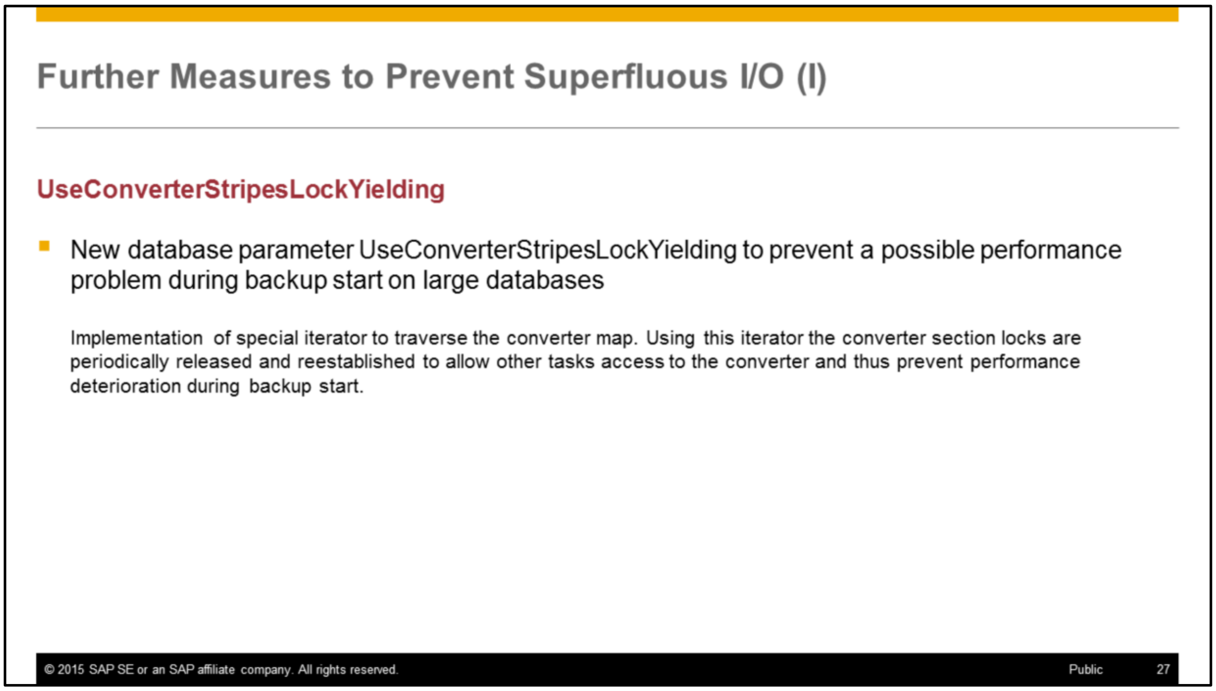
© 2015 SAP SE or an SAP affiliate company. All rights reserved. Public 26

As in case of restart the marking for backup within FBM is very slow on some platforms. A solution has been designed which is similar to the restart behaviour. The affected blocks are collected first, grouped by volume in the cache and then passed to FBM manager as a mass operation.

The size of the cache is defined by setting the parameter '**RestartVolumeGroupingMemorySize**'. If it is set to 0, the old and not optimized processing is done.

“This parameter potentially improves restart performance, but only in those (rare) cases when the system is CPU bound during restart (we would normally expect it to be IO bound).
Increasing this parameter will allocate more memory in one chunk, which we have seen to be beneficial on Solaris OS.”

Explain:
This value defines the amount of memory (in KB) used for grouping of used blocks by volume in the course of a system restart. Using this kind of grouping may speed up the restart in case of huge databases.



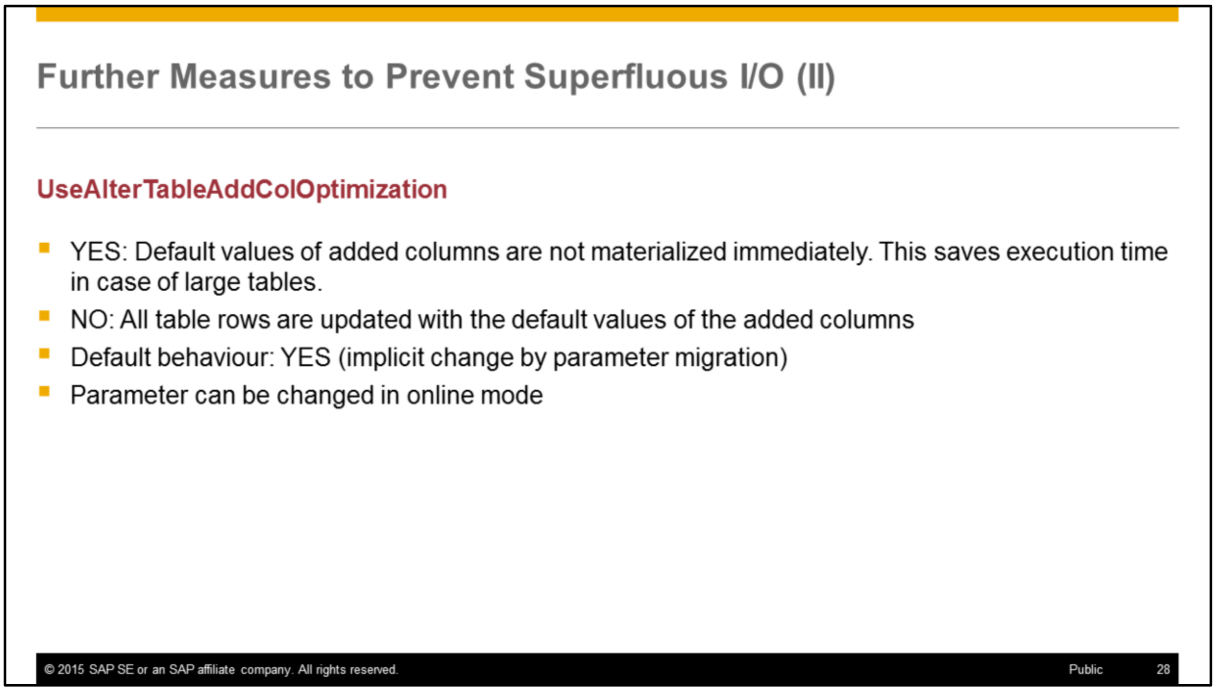
Further Measures to Prevent Superfluous I/O (I)

UseConverterStripesLockYielding

- New database parameter UseConverterStripesLockYielding to prevent a possible performance problem during backup start on large databases

Implementation of special iterator to traverse the converter map. Using this iterator the converter section locks are periodically released and reestablished to allow other tasks access to the converter and thus prevent performance deterioration during backup start.

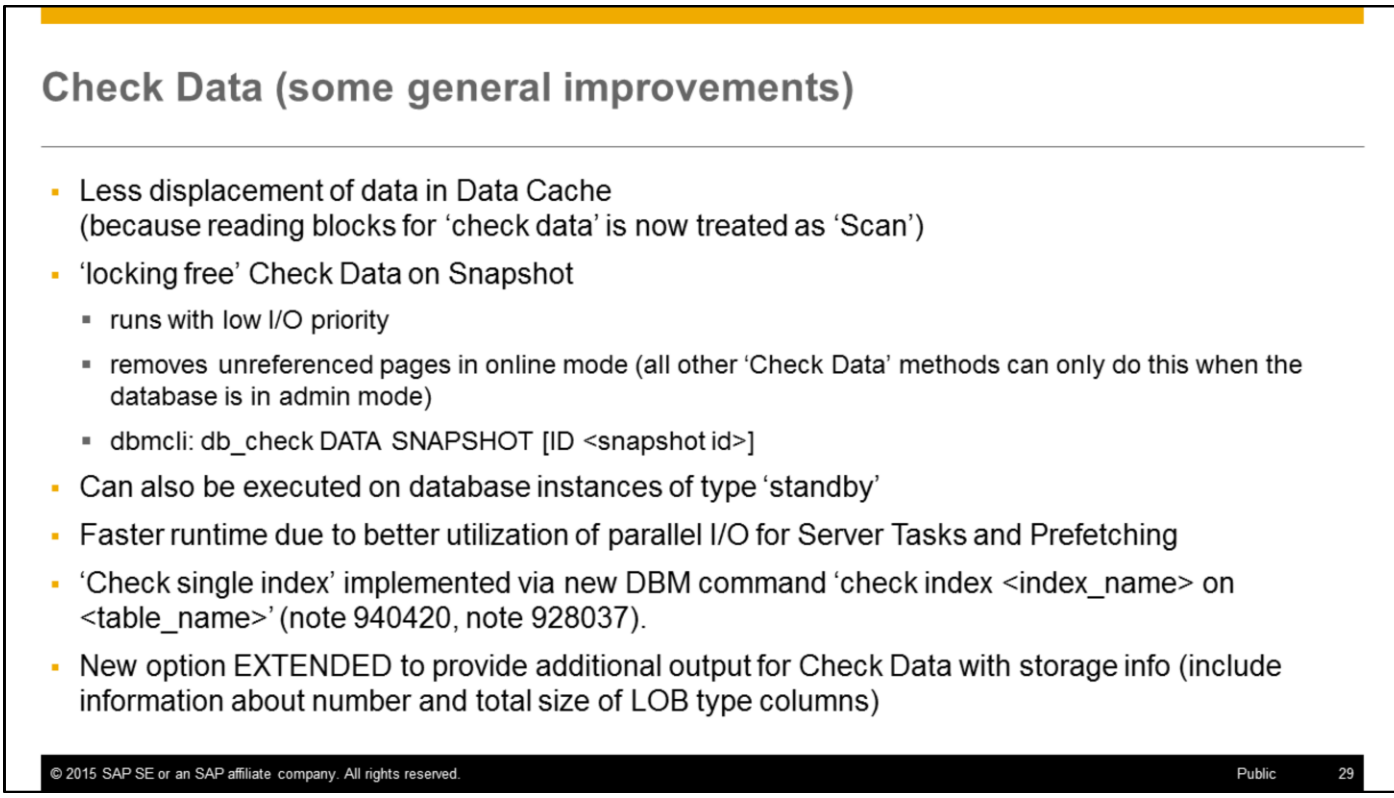
Public 27



Further Measures to Prevent Superfluous I/O (II)

UseAlterTableAddColOptimization

- YES: Default values of added columns are not materialized immediately. This saves execution time in case of large tables.
- NO: All table rows are updated with the default values of the added columns
- Default behaviour: YES (implicit change by parameter migration)
- Parameter can be changed in online mode

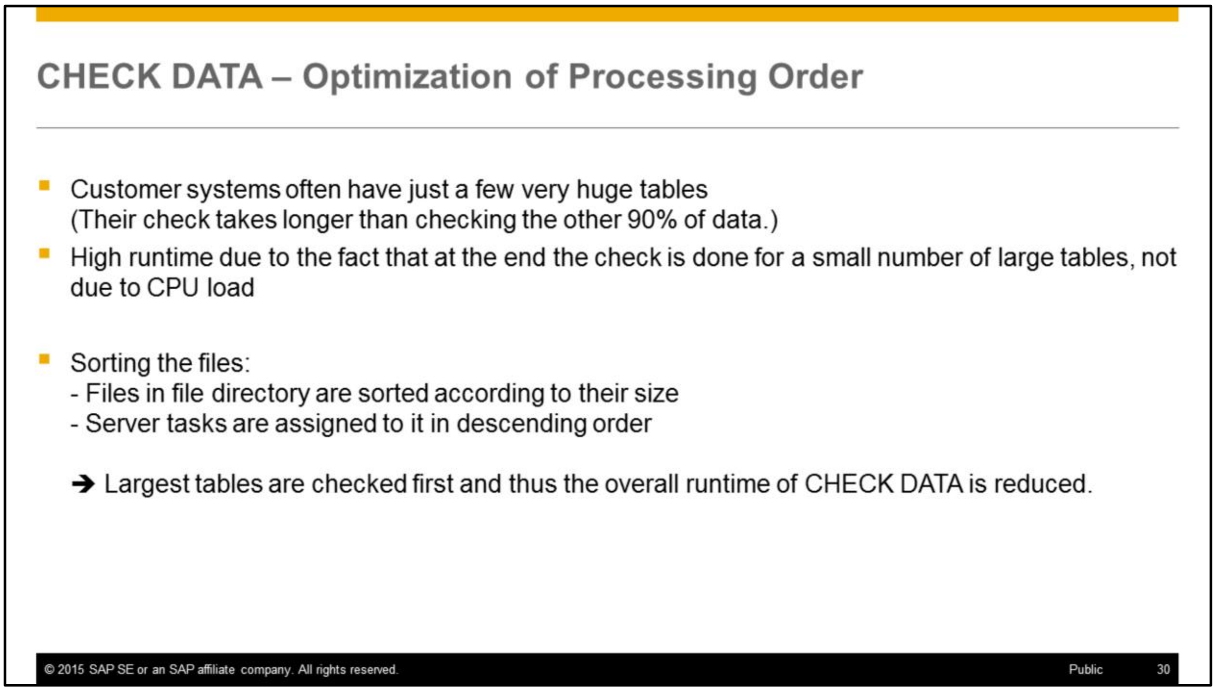


Check Data (some general improvements)

- Less displacement of data in Data Cache
  (because reading blocks for ‘check data’ is now treated as ‘Scan’)
- ‘locking free’ Check Data on Snapshot
  - runs with low I/O priority
  - removes unreferenced pages in online mode (all other ‘Check Data’ methods can only do this when the database is in admin mode)
  - dbmcli: db_check DATA SNAPSHOT [ID <snapshot id>]
- Can also be executed on database instances of type ‘standby’
- Faster runtime due to better utilization of parallel I/O for Server Tasks and Prefetching
- ‘Check single index’ implemented via new DBM command ‘check index <index_name> on <table_name>’ (note 940420, note 928037).
- New option EXTENDED to provide additional output for Check Data with storage info (include information about number and total size of LOB type columns)

The collection of exact table information in TABLESTORAGEDETAILS is disadvantageous as the table has to be locked.

Thus, it makes sense to combine the collection with Check Data.



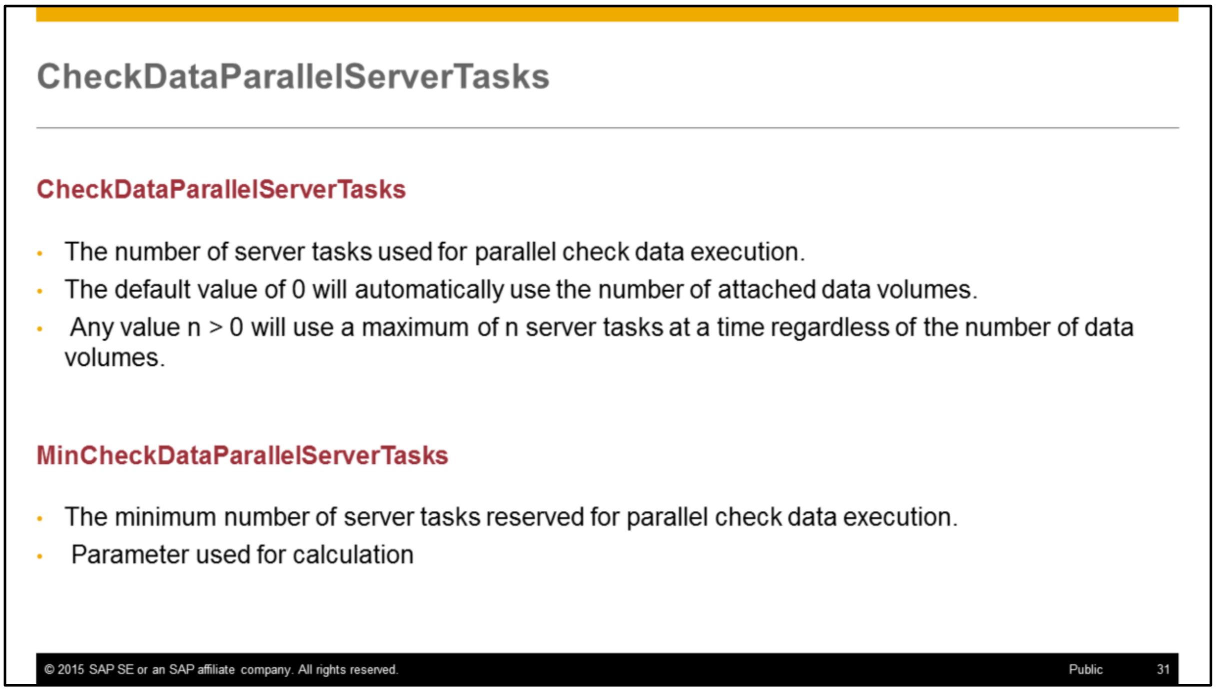
CHECK DATA – Optimization of Processing Order

- Customer systems often have just a few very huge tables
  (Their check takes longer than checking the other 90% of data.)
- High runtime due to the fact that at the end the check is done for a small number of large tables, not due to CPU load

- Sorting the files:
  - Files in file directory are sorted according to their size
  - Server tasks are assigned to it in descending order

  ➔ Largest tables are checked first and thus the overall runtime of CHECK DATA is reduced.

Public 30



CheckDataParallelServerTasks

CheckDataParallelServerTasks

- The number of server tasks used for parallel check data execution.
- The default value of 0 will automatically use the number of attached data volumes.
- Any value n > 0 will use a maximum of n server tasks at a time regardless of the number of data volumes.

MinCheckDataParallelServerTasks

- The minimum number of server tasks reserved for parallel check data execution.
- Parameter used for calculation

Server tasks are responsible for importing data to be checked and the execution of the check.
A server task takes on the job of the coordinator and starts additional server tasks for the delivered files (from the file directory that have to be checked). These server tasks import the data into the cache where they execute the check.

One server tasks deals with the import and check of the data for a file (f.e. table, index, catalog object).

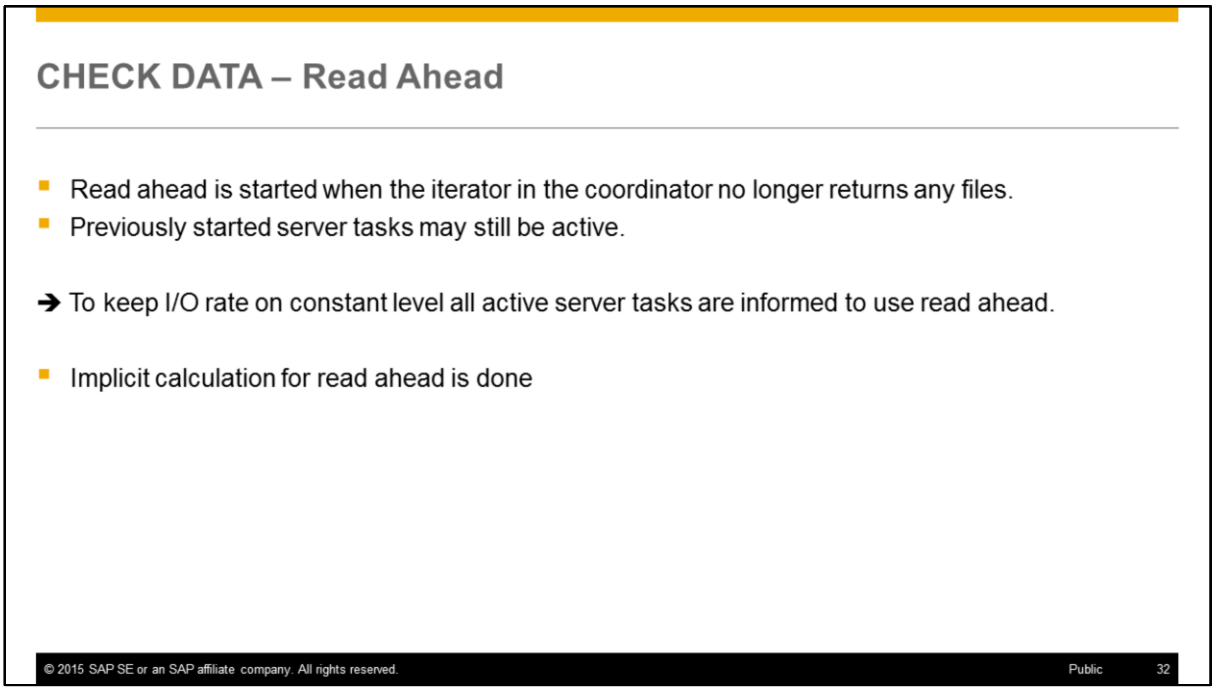
By setting the parameter you can enhance the number of server tasks used for CHECK DATA and thus the amount of read ahead operations.

A configuration with **CheckDataParallelServertasks** > 0 is not suitable for every system. It should only be set if a powerful I/O system is used.

The database parameter **CheckDataParallelServertasks** belongs to the group of support database parameters and is changeable in online mode.

Detailed information can be found in note
1837280 - SAP MaxDB: CHECK DATA runtime optimization

There is no guarantee that as many server tasks as given will actually be used as there may be other database jobs competing for the server tasks available.



CHECK DATA – Read Ahead

- Read ahead is started when the iterator in the coordinator no longer returns any files.
- Previously started server tasks may still be active.

➔ To keep I/O rate on constant level all active server tasks are informed to use read ahead.

- Implicit calculation for read ahead is done

The implicit calculation of the number of pages used for read ahead depends on the configuration.

Generally the calculation should make sure that a similar I/O rate is achieved as if all server tasks were still active. Be aware this calculated bandwidth is shared with all still active server tasks.

Default configuration:

The number of pages is determined by the number of configured server tasks. If parameter **CheckDataParallelServerTasks** is set to 0, the maximum number of pages read in advance by read ahead) corresponds to the number of volumes.

Enhanced configuration:

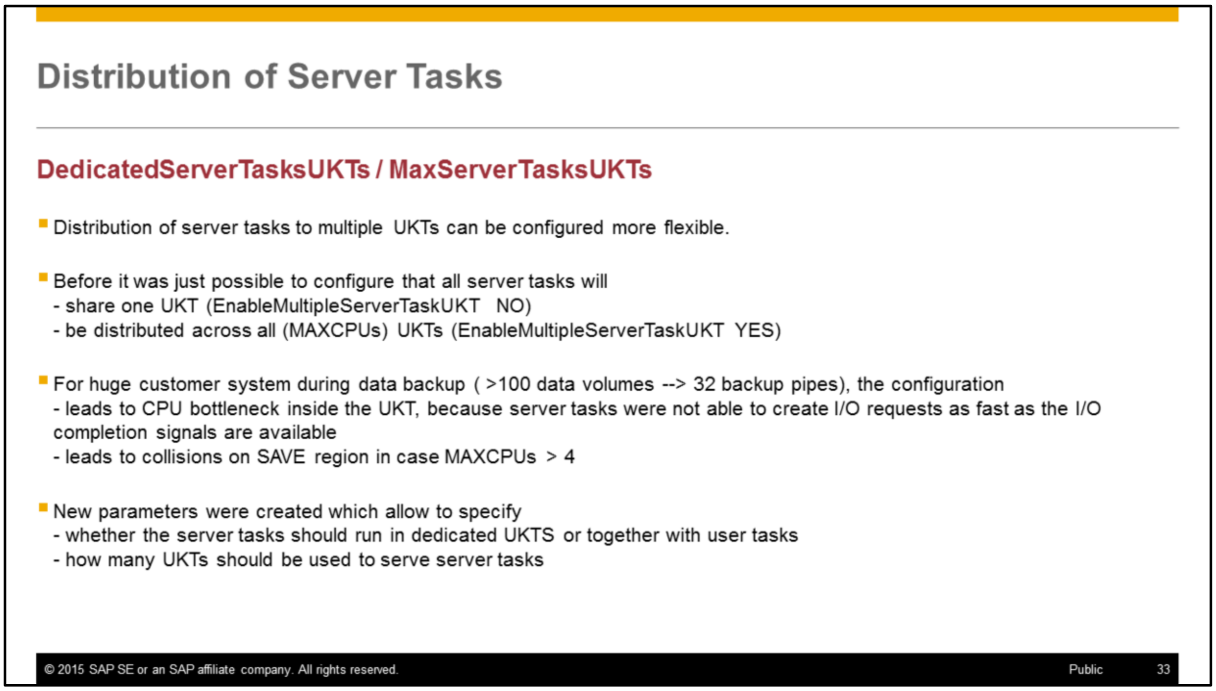
Increase the bandwidth for read ahead by setting the parameter.

Example:
- **CheckDataParallelServerTasks** is set to 100
- 10 server tasks are still running

➔ each server task can do a read ahead of 10 pages

After completion of the work of one or more server tasks the bandwidth is redistributed.

Monitoring:
- use operating system monitoring to check the load of the I/O system
- if not all capacity is used this may indicate that the parameter can be set to a higher value
- consider other parallel running actions causing I/O



Distribution of Server Tasks

DedicatedServerTasksUKTs / MaxServerTasksUKTs

- Distribution of server tasks to multiple UKTs can be configured more flexible.
- Before it was just possible to configure that all server tasks will
  - share one UKT (EnableMultipleServerTaskUKT NO)
  - be distributed across all (MAXCPUs) UKTs (EnableMultipleServerTaskUKT YES)
- For huge customer system during data backup (>100 data volumes --> 32 backup pipes), the configuration
  - leads to CPU bottleneck inside the UKT, because server tasks were not able to create I/O requests as fast as the I/O completion signals are available
  - leads to collisions on SAVE region in case MAXCPUs > 4
- New parameters were created which allow to specify
  - whether the server tasks should run in dedicated UKTS or together with user tasks
  - how many UKTs should be used to serve server tasks

Special improvements for Check Data have been explained before. Now we take a more general look to server tasks which are also used for data backup and create index.

The configuration of server tasks has become more flexible.

Motivation:

In very big systems (a lot of CPUs, fast I/O) the UKTs with special tasks may become CPU-bound.

- During backup the server tasks could not generate I/O requests as fast as the I/O system has delivered.
- The same might happen during savepoints.
- For garbage collectors the distribution to several threads might be advantageous.
- During Check Data or index creation the thread with server tasks was CPU-bound and thus too slow (f.e. because too many server tasks were in one thread or the disk‘s performance is very good.

If **DedicatedServerTaskUKTs** is set to YES then the server tasks run in special thread(s). This is the default behaviour.

MaxServerTasksUKTs defines the number of threads to which the server tasks are distributed.

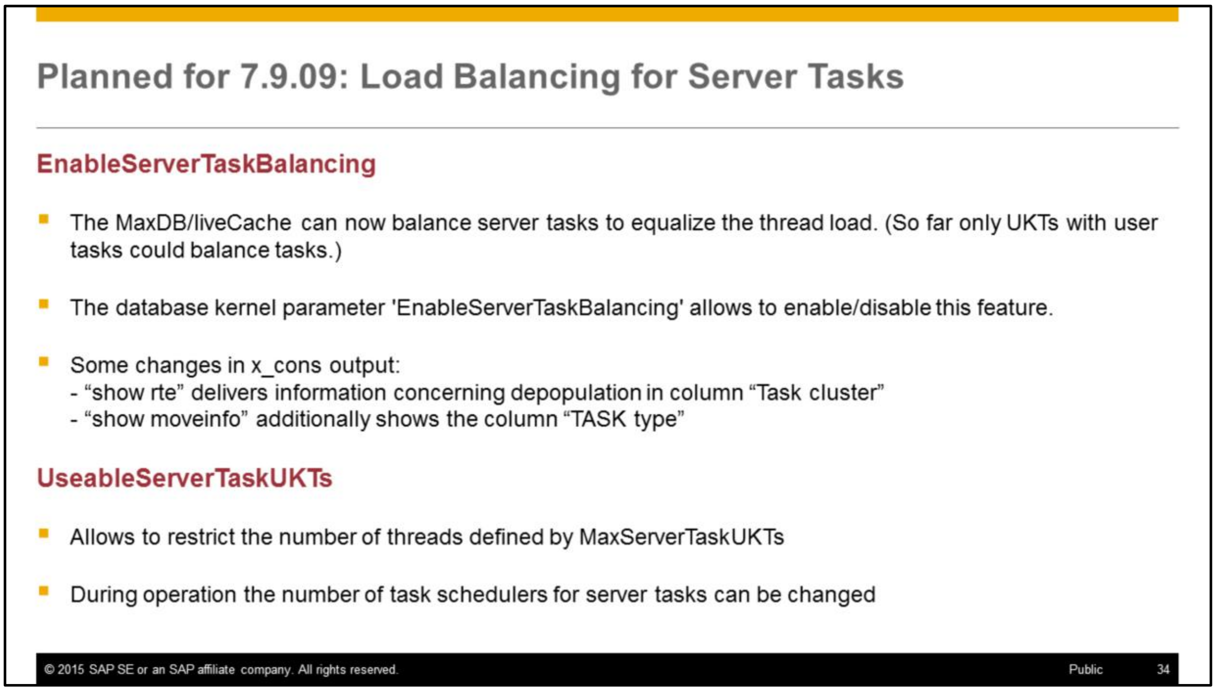
If **DedicatedServerTaskUKTs** is set to NO the server tasks are configured as running together with user tasks in UKTs. As default **MaxServerTasksUKTs** is equal to **UseableCPUs**, then.

Detailed information can be found in note 1672994.
1672994 - SAP MaxDB: configuration of server tasks

Consider DB-Analyzer files DBAN_SV and DBAN_UKT_CPU_UTILIZATION to check the behaviour.

The mechanism can be used in 7.8 and 7.9 versions.

(A problem with the parameter migration from 7.8 to 7.9 has been solved with PTS 1251378.)



Planned for 7.9.09: Load Balancing for Server Tasks

EnableServerTaskBalancing

- The MaxDB/liveCache can now balance server tasks to equalize the thread load. (So far only UKTs with user tasks could balance tasks.)
- The database kernel parameter 'EnableServerTaskBalancing' allows to enable/disable this feature.
- Some changes in x_cons output:
  - “show rte” delivers information concerning depopulation in column “Task cluster”
  - “show moveinfo” additionally shows the column “TASK type”

UseableServerTaskUKTs

- Allows to restrict the number of threads defined by MaxServerTaskUKTs
- During operation the number of task schedulers for server tasks can be changed

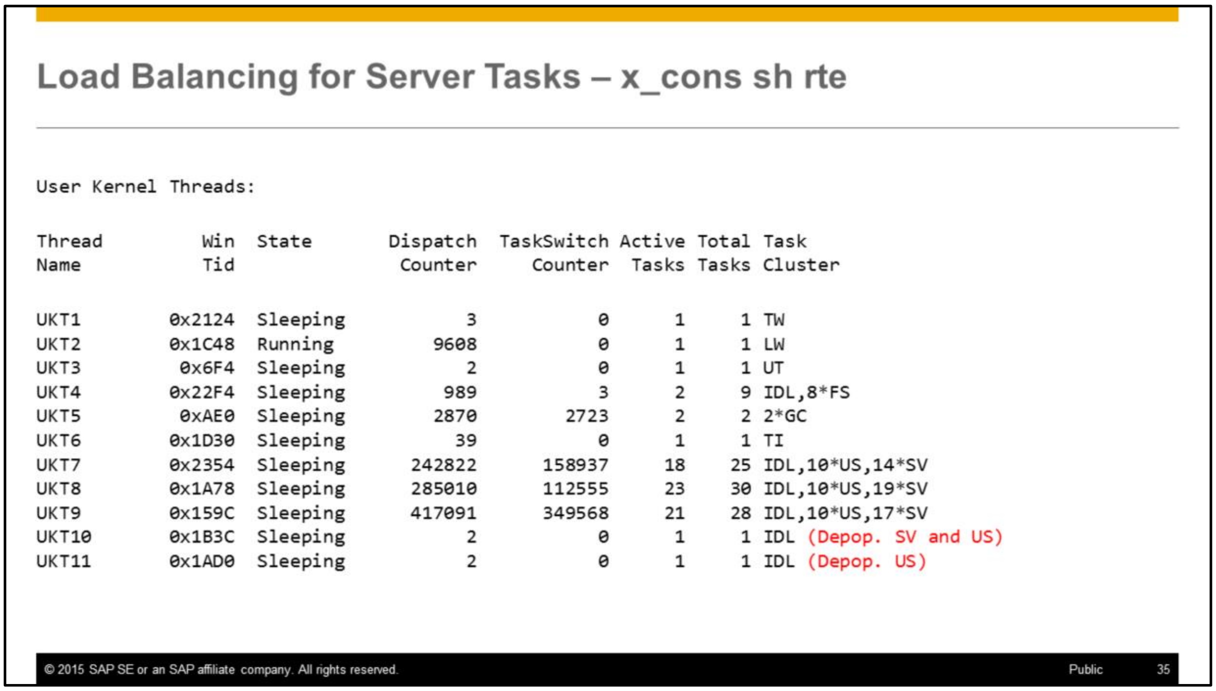
Public

New implementation of load balancing for server tasks (not released for customer systems yet.)

Different balancing groups have been introduced to make sure that the task types are not mixed up.

The new parameter **EnableServerTaskBalancing** allows to activate/deactivate the feature.

Additionally the parameter **UseableServerTaskUKTs** came to existence and allows (analogous to **UseableCPUs**) to restrict the number of threads with server tasks.



Load Balancing for Server Tasks – x_cons sh rte

```
User Kernel Threads:

Thread          Win  State       Dispatch  TaskSwitch Active Total Task
Name            Tid               Counter     Counter  Tasks Tasks Cluster

UKT1         0x2124  Sleeping           3           0      1     1 TW
UKT2         0x1C48  Running         9608           0      1     1 LW
UKT3          0x6F4  Sleeping           2           0      1     1 UT
UKT4         0x22F4  Sleeping         989           3      2     9 IDL,8*FS
UKT5          0xAE0  Sleeping        2870        2723      2     2 2*GC
UKT6         0x1D30  Sleeping          39           0      1     1 TI
UKT7         0x2354  Sleeping      242822      158937     18    25 IDL,10*US,14*SV
UKT8         0x1A78  Sleeping      285010      112555     23    30 IDL,10*US,19*SV
UKT9         0x159C  Sleeping      417091      349568     21    28 IDL,10*US,17*SV
UKT10        0x1B3C  Sleeping           2           0      1     1 IDL (Depop. SV and US)
UKT11        0x1AD0  Sleeping           2           0      1     1 IDL (Depop. US)
```

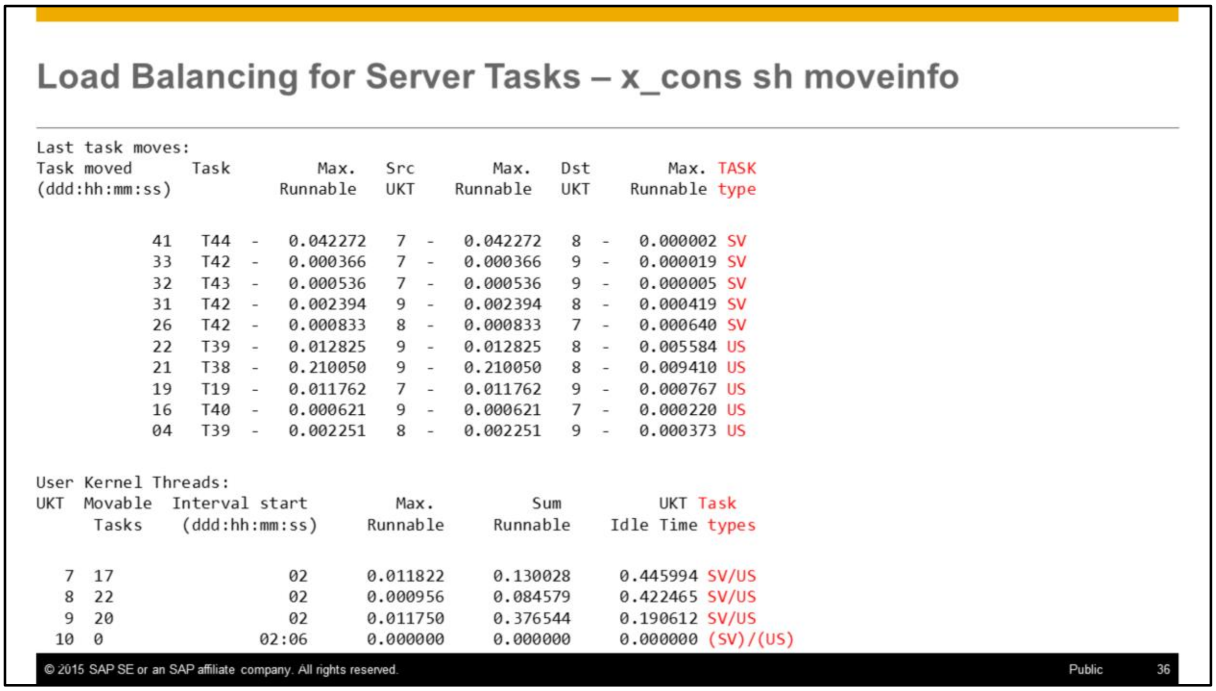
Public 35

The column „Task Cluster“ now contains information about „depopulation“.

If you see “(Depop. xxxxx)” the parameter **UseableCPUs** is smaller than **MaxCPUs** and/or the value of **UseableServerTaskUKTs** is smaller than **MaxServerTasksUKTs**.

UseableCPUs influences user tasks and **UseableServerTaskUKTs** is valid for server tasks.

In the marked UKTs(Task Scheduler) the tasks are balanced to other UKTs.



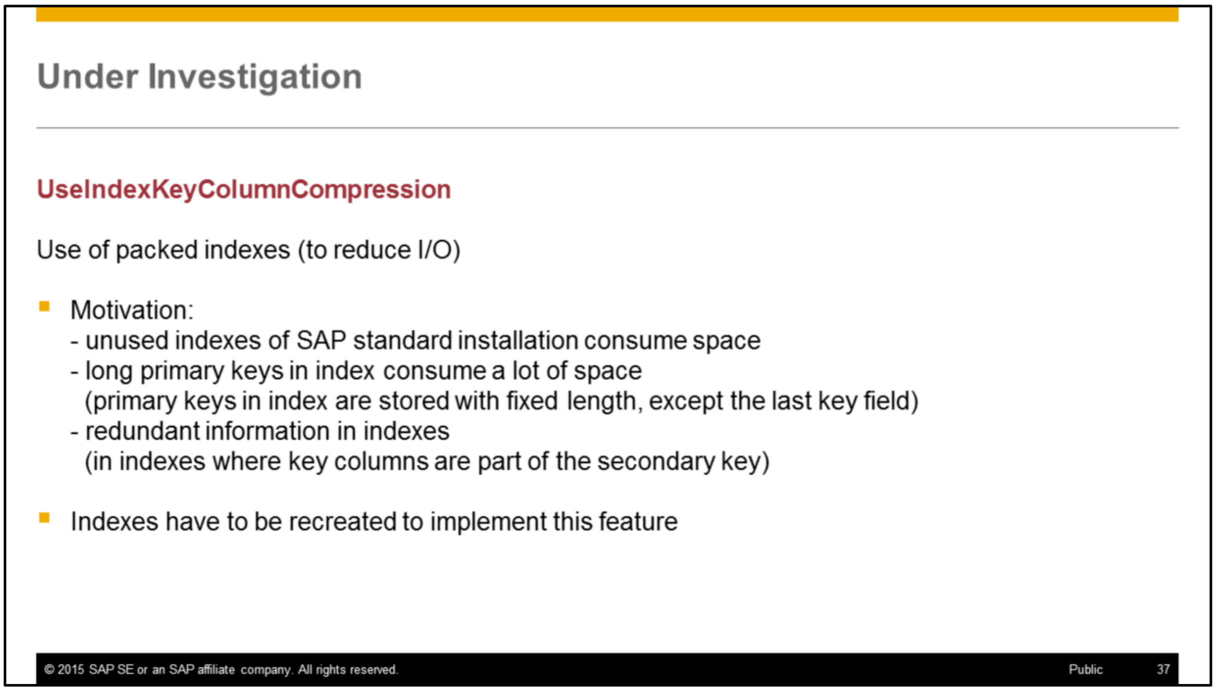
Load Balancing for Server Tasks – x_cons sh moveinfo

```
Last task moves:
Task moved      Task         Max.   Src        Max.   Dst        Max. TASK
(ddd:hh:mm:ss)            Runnable   UKT    Runnable   UKT    Runnable type

            41   T44  -   0.042272   7  -   0.042272   8  -   0.000002 SV
            33   T42  -   0.000366   7  -   0.000366   9  -   0.000019 SV
            32   T43  -   0.000536   7  -   0.000536   9  -   0.000005 SV
            31   T42  -   0.002394   9  -   0.002394   8  -   0.000419 SV
            26   T42  -   0.000833   8  -   0.000833   7  -   0.000640 SV
            22   T39  -   0.012825   9  -   0.012825   8  -   0.005584 US
            21   T38  -   0.210050   9  -   0.210050   8  -   0.009410 US
            19   T19  -   0.011762   7  -   0.011762   9  -   0.000767 US
            16   T40  -   0.000621   9  -   0.000621   7  -   0.000220 US
            04   T39  -   0.002251   8  -   0.002251   9  -   0.000373 US

User Kernel Threads:
UKT  Movable  Interval start         Max.         Sum          UKT Task
      Tasks    (ddd:hh:mm:ss)      Runnable     Runnable    Idle Time types

   7  17                 02       0.011822     0.130028     0.445994 SV/US
   8  22                 02       0.000956     0.084579     0.422465 SV/US
   9  20                 02       0.011750     0.376544     0.190612 SV/US
  10  0               02:06       0.000000     0.000000     0.000000 (SV)/(US)
```

Public 36

The command „show moveinfo“ now additionally shows the „TASK type“ of the balanced tasks.

The table „User Kernel Threads“ also has the additional column showing which task types have been balanced into or out of this UKT. Types in parentheses mean that tasks of this type are removed from this UKT (depopulated).

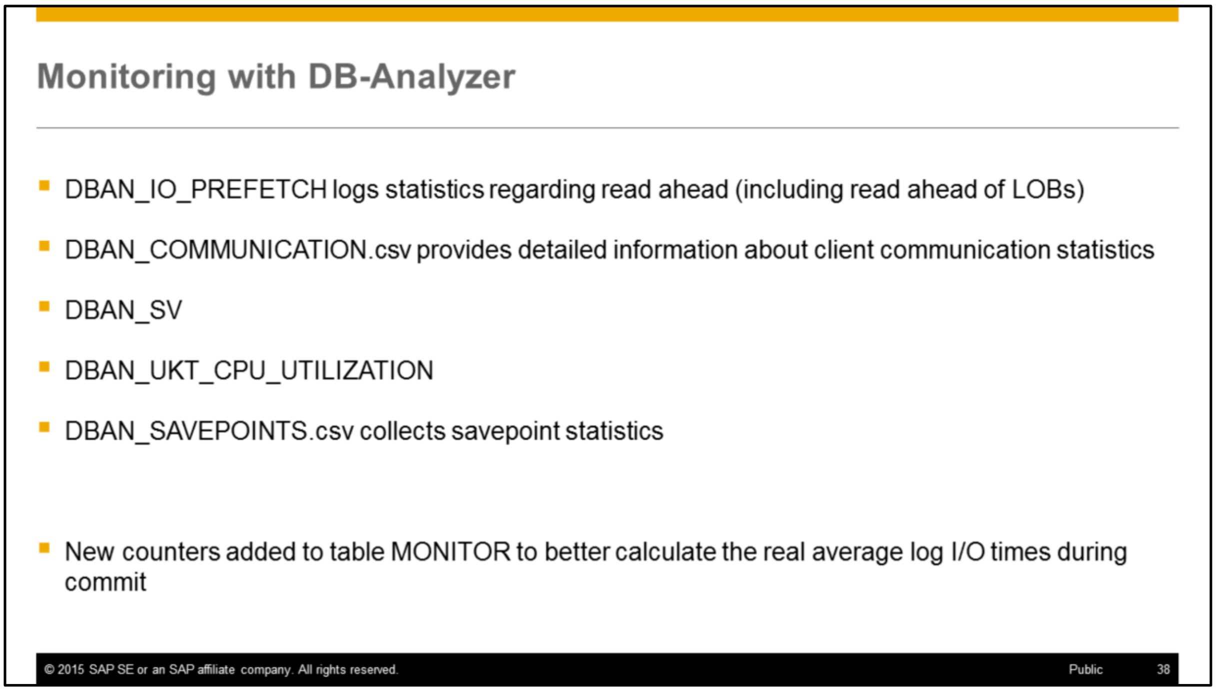


Under Investigation

UseIndexKeyColumnCompression

Use of packed indexes (to reduce I/O)

- Motivation:
  - unused indexes of SAP standard installation consume space
  - long primary keys in index consume a lot of space
    (primary keys in index are stored with fixed length, except the last key field)
  - redundant information in indexes
    (in indexes where key columns are part of the secondary key)
- Indexes have to be recreated to implement this feature



Monitoring with DB-Analyzer

- DBAN_IO_PREFETCH logs statistics regarding read ahead (including read ahead of LOBs)
- DBAN_COMMUNICATION.csv provides detailed information about client communication statistics
- DBAN_SV
- DBAN_UKT_CPU_UTILIZATION
- DBAN_SAVEPOINTS.csv collects savepoint statistics

- New counters added to table MONITOR to better calculate the real average log I/O times during commit

Public 38

DBAN_IO_PREFETCH

Check columns „Requests“ and „Ignored Requests“
If the percentage of read ahead requests ignored is >= 5% of all read ahead requests, this indicates that you could achieve a higher throughput for CHECK DATA by increasing the database parameter MaxDataAreaReadAheadRequests.

DBAN_SV

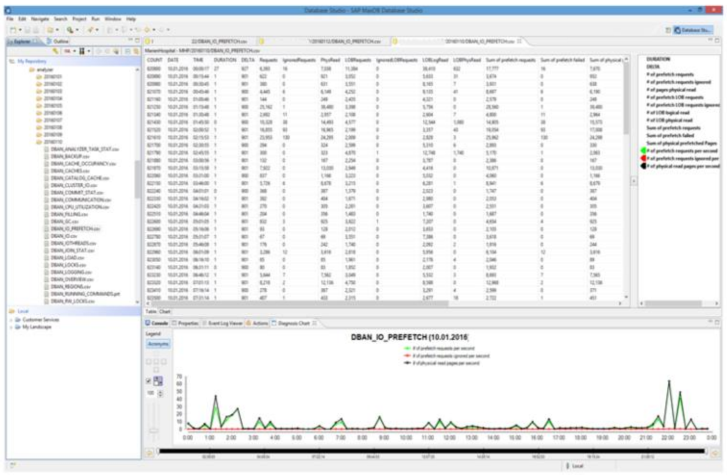
Check columns “ActiveServerTasks”, “IOWaitServerTasks”, “SV_PRThread”, “SV_PhysPRThreadPg”, “AvgAbsRTime_SV”, “AvgrelRTime_SV”, “RunnableSVTasks”

If “AvgrelRTime_SV “differs from “AvgAbsRTime_SV” in such a way that the value of the absolute time (AvgAbsRTime_SV) is a lot higher than the value of the relative time (AvgrelRTime_SV), this indicates a CPU bottleneck in the UKT in which the server tasks run. This CPU bottleneck (CPU bound) can be resolved by distributing the server tasks over several different UKTs.

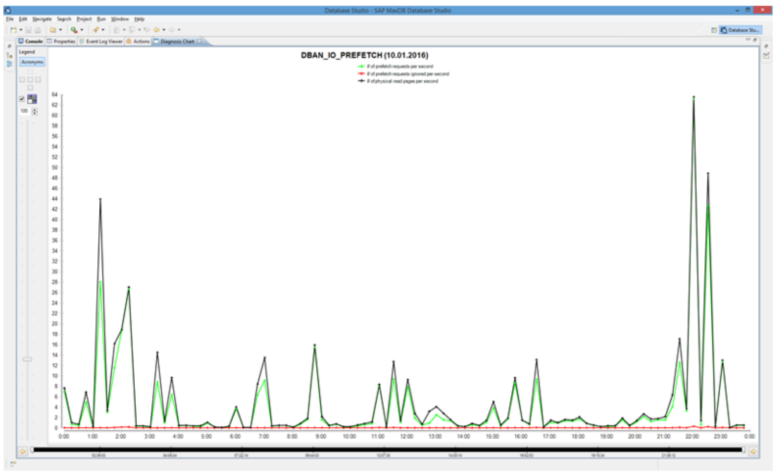
DBAN_UKT_CPU_UTILIZATION

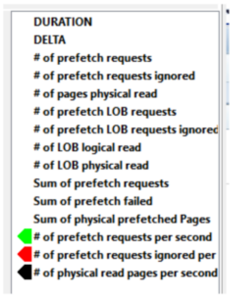
To confirm the CPU bottleneck if all server tasks are configured in one UKT, you can additionally use the file UKT_CPU_UTILIZATION.prt in the Database Analyzer expert analysis. This file provides an overview of the CPU load of all database threads. The server tasks are displayed in the column "Taskcluster" with the abbreviation SV.

Monitoring with DB-Analyzer (II) - Example

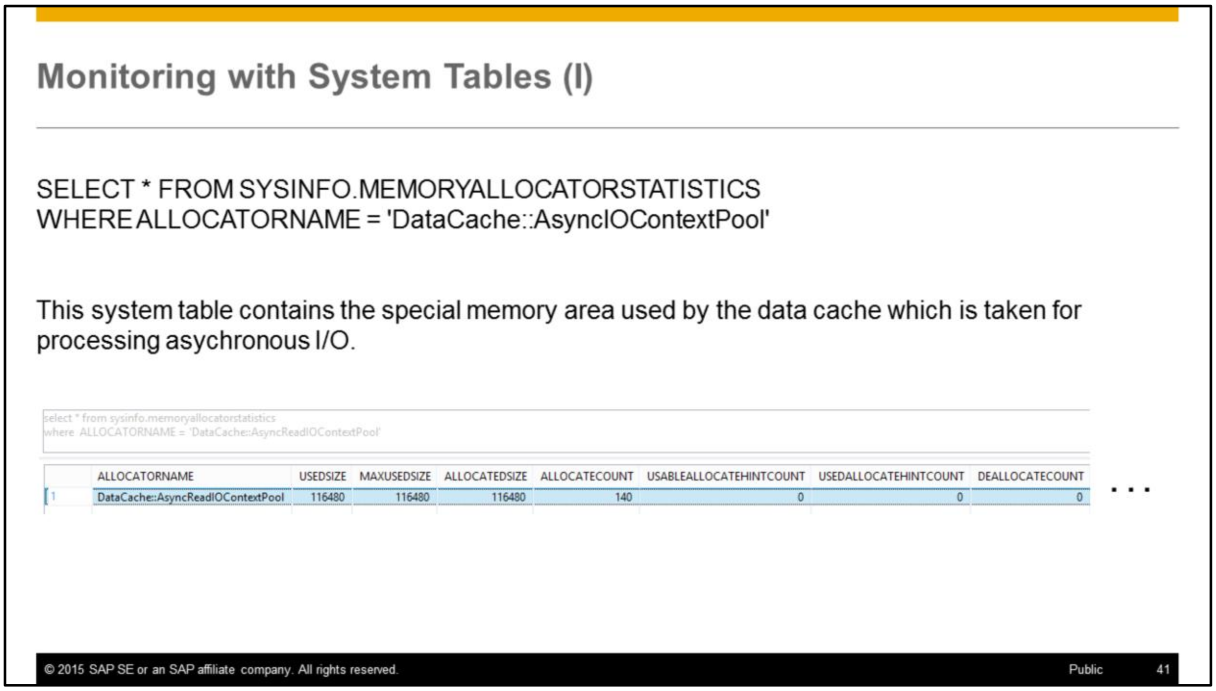

Database Studio - SAP MaxDB Database Studio
DBAN_IO_PREFETCH (10.01.2016)
# of prefetch requests per second
# of prefetch requests ignored per second
# of physical read pages per second

Monitoring with DB-Analyzer (III) - Example





DURATION
DELTA
# of prefetch requests
# of prefetch requests ignored
# of pages physical read
# of prefetch LOB requests
# of prefetch LOB requests ignored
# of LOB logical read
# of LOB physical read
Sum of prefetch requests
Sum of prefetch failed
Sum of physical prefetched Pages
🟩 # of prefetch requests per second
🟥 # of prefetch requests ignored per
⬛ # of physical read pages per second



Monitoring with System Tables (I)

SELECT * FROM SYSINFO.MEMORYALLOCATORSTATISTICS
WHERE ALLOCATORNAME = 'DataCache::AsyncIOContextPool'

This system table contains the special memory area used by the data cache which is taken for processing asychronous I/O.

```
select * from sysinfo.memoryallocatorstatistics
where ALLOCATORNAME = 'DataCache::AsyncReadIOContextPool'
```

	ALLOCATORNAME	USEDSIZE	MAXUSEDSIZE	ALLOCATEDSIZE	ALLOCATECOUNT	USABLEALLOCATEHINTCOUNT	USEDALLOCATEHINTCOUNT	DEALLOCATECOUNT
1	DataCache::AsyncReadIOContextPool	116480	116480	116480	140	0	0	0

. . .

You can watch the state of the server task reserved for asynchronous I/O with
x_cons <DB> sh tas

Example output:

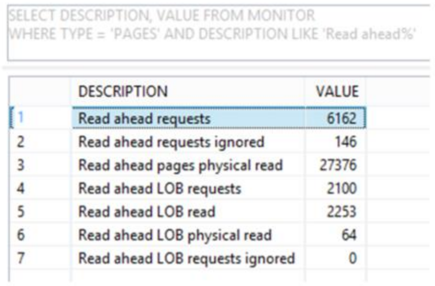
```
T60    4  0xD1C "IOCompl"        IOComPort Wait     255   0   150196(s)
```

Monitoring with System Tables (II)

SELECT DESCRIPTION, VALUE FROM MONITOR
WHERE TYPE = 'PAGES' AND DESCRIPTION LIKE 'Read ahead%'

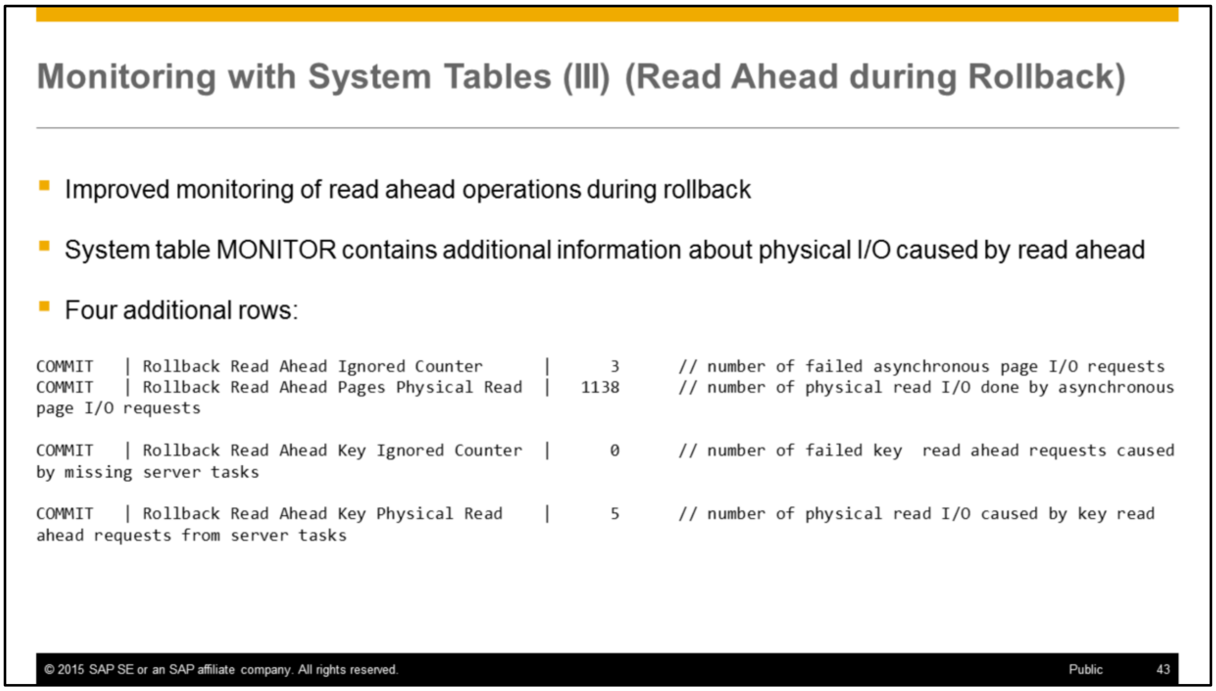
The system table MONITOR contains instance specific statistics and beyond that also information about read ahead:

- Read ahead requests – whole number of read ahead requests
- Read ahead requests ignored - number of requests which could not be processed with read ahead
- Read ahead pages physical read – number of pages which were read with read ahead from the disk



```
SELECT DESCRIPTION, VALUE FROM MONITOR
WHERE TYPE = 'PAGES' AND DESCRIPTION LIKE 'Read ahead%'
```

	DESCRIPTION	VALUE
1	Read ahead requests	6162
2	Read ahead requests ignored	146
3	Read ahead pages physical read	27376
4	Read ahead LOB requests	2100
5	Read ahead LOB read	2253
6	Read ahead LOB physical read	64
7	Read ahead LOB requests ignored	0

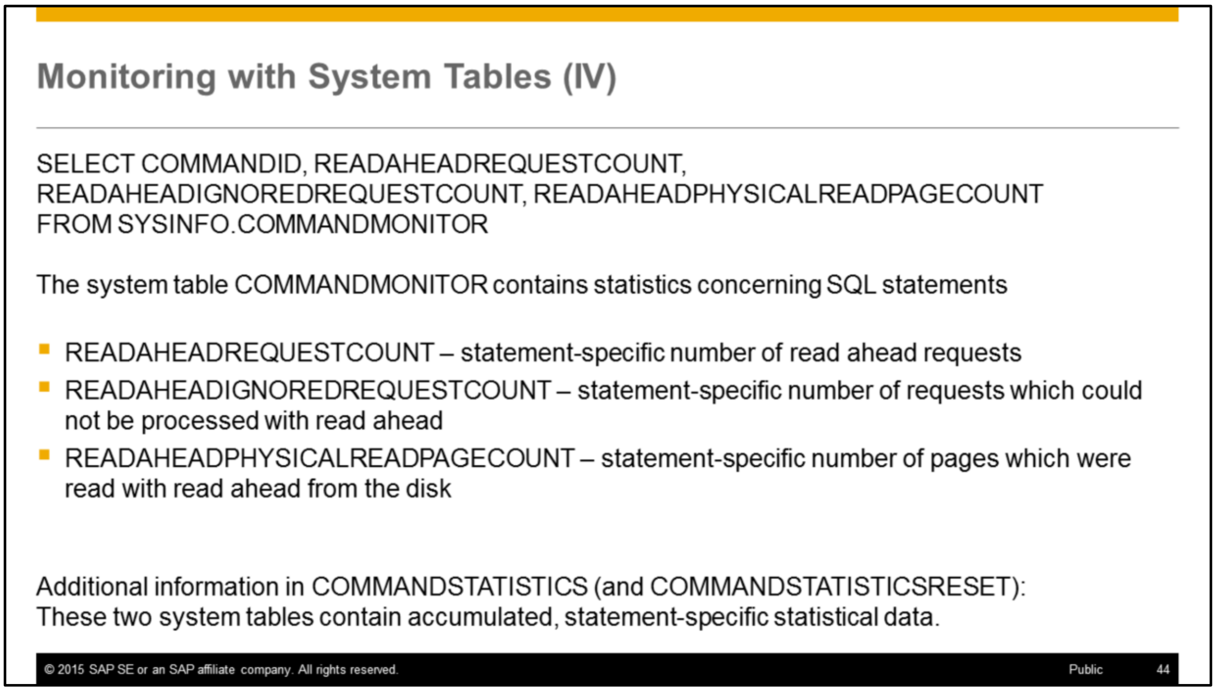


Monitoring with System Tables (III) (Read Ahead during Rollback)

- Improved monitoring of read ahead operations during rollback
- System table MONITOR contains additional information about physical I/O caused by read ahead
- Four additional rows:

```
COMMIT    | Rollback Read Ahead Ignored Counter      |     3      // number of failed asynchronous page I/O requests
COMMIT    | Rollback Read Ahead Pages Physical Read  |  1138      // number of physical read I/O done by asynchronous
page I/O requests

COMMIT    | Rollback Read Ahead Key Ignored Counter  |     0      // number of failed key  read ahead requests caused
by missing server tasks

COMMIT    | Rollback Read Ahead Key Physical Read    |     5      // number of physical read I/O caused by key read
ahead requests from server tasks
```

Public



Monitoring with System Tables (IV)

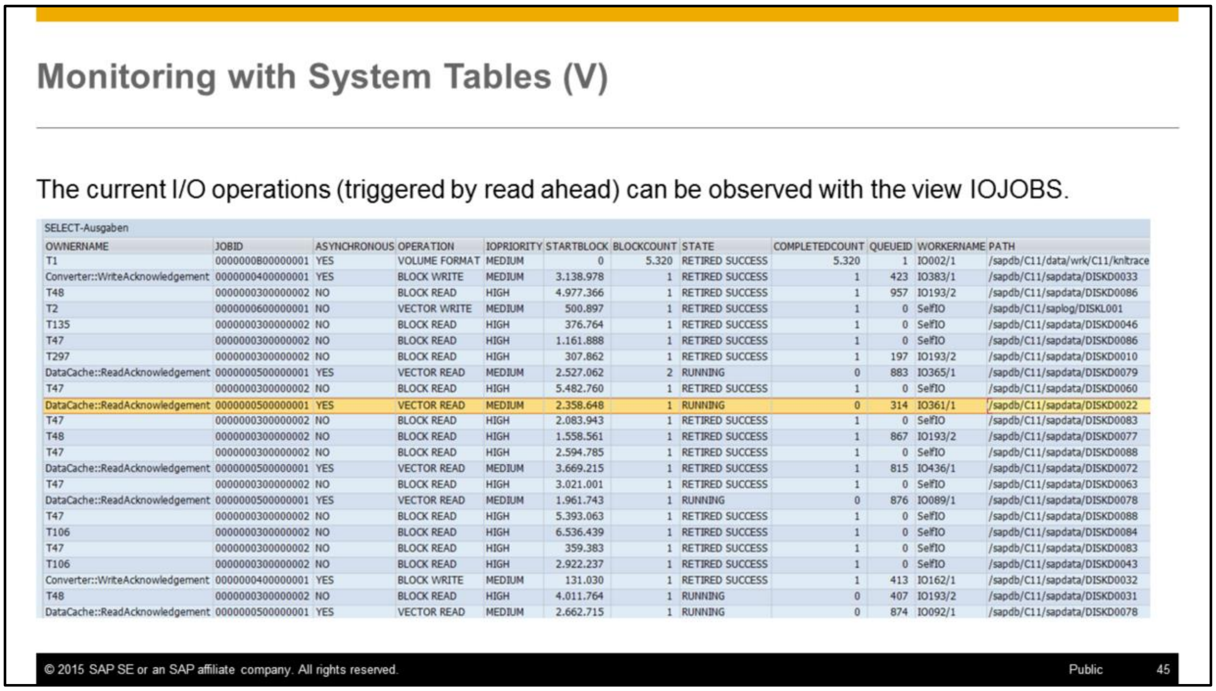
SELECT COMMANDID, READAHEADREQUESTCOUNT,
READAHEADIGNOREDREQUESTCOUNT, READAHEADPHYSICALREADPAGECOUNT
FROM SYSINFO.COMMANDMONITOR

The system table COMMANDMONITOR contains statistics concerning SQL statements

- READAHEADREQUESTCOUNT – statement-specific number of read ahead requests
- READAHEADIGNOREDREQUESTCOUNT – statement-specific number of requests which could not be processed with read ahead
- READAHEADPHYSICALREADPAGECOUNT – statement-specific number of pages which were read with read ahead from the disk

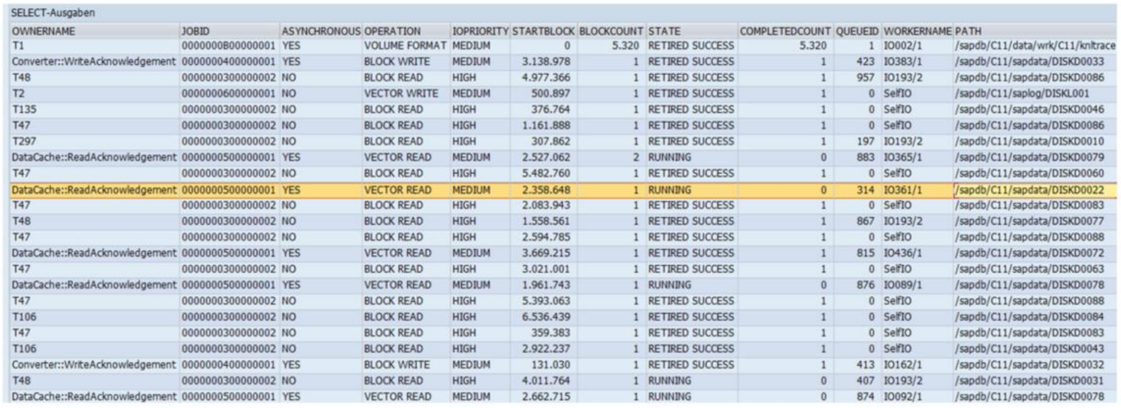
Additional information in COMMANDSTATISTICS (and COMMANDSTATISTICSRESET):
These two system tables contain accumulated, statement-specific statistical data.

Public 44



Monitoring with System Tables (V)

The current I/O operations (triggered by read ahead) can be observed with the view IOJOBS.



SELECT-Ausgaben

OWNERNAME	JOBID	ASYNCHRONOUS	OPERATION	IOPRIORITY	STARTBLOCK	BLOCKCOUNT	STATE	COMPLETEDCOUNT	QUEUEID	WORKERNAME	PATH
T1	0000000B00000001	YES	VOLUME FORMAT	MEDIUM	0	5.320	RETIRED SUCCESS	5.320	1	IO002/1	/sapdb/C11/data/wrk/C11/knltrace
Converter::WriteAcknowledgement	0000000400000001	YES	BLOCK WRITE	MEDIUM	3.138.978	1	RETIRED SUCCESS	1	423	IO383/1	/sapdb/C11/sapdata/DISKD0033
T48	0000000300000002	NO	BLOCK READ	HIGH	4.977.366	1	RETIRED SUCCESS	1	957	IO193/2	/sapdb/C11/sapdata/DISKD0086
T2	0000000600000001	NO	VECTOR WRITE	MEDIUM	500.897	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/saplog/DISKL001
T135	0000000300000002	NO	BLOCK READ	HIGH	376.764	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/sapdata/DISKD0046
T47	0000000300000002	NO	BLOCK READ	HIGH	1.161.888	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/sapdata/DISKD0086
T297	0000000300000002	NO	BLOCK READ	HIGH	307.862	1	RETIRED SUCCESS	1	197	IO193/2	/sapdb/C11/sapdata/DISKD0010
DataCache::ReadAcknowledgement	0000000500000001	YES	VECTOR READ	MEDIUM	2.527.062	2	RUNNING	0	883	IO365/1	/sapdb/C11/sapdata/DISKD0079
T47	0000000300000002	NO	BLOCK READ	HIGH	5.482.760	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/sapdata/DISKD0060
DataCache::ReadAcknowledgement	0000000500000001	YES	VECTOR READ	MEDIUM	2.358.648	1	RUNNING	0	314	IO361/1	/sapdb/C11/sapdata/DISKD0022
T47	0000000300000002	NO	BLOCK READ	HIGH	2.083.943	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/sapdata/DISKD0083
T48	0000000300000002	NO	BLOCK READ	HIGH	1.558.561	1	RETIRED SUCCESS	1	867	IO193/2	/sapdb/C11/sapdata/DISKD0077
T47	0000000300000002	NO	BLOCK READ	HIGH	2.594.785	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/sapdata/DISKD0088
DataCache::ReadAcknowledgement	0000000500000001	YES	VECTOR READ	MEDIUM	3.669.215	1	RETIRED SUCCESS	1	815	IO436/1	/sapdb/C11/sapdata/DISKD0072
T47	0000000300000002	NO	BLOCK READ	HIGH	3.021.001	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/sapdata/DISKD0063
DataCache::ReadAcknowledgement	0000000500000001	YES	VECTOR READ	MEDIUM	1.961.743	1	RUNNING	0	876	IO089/1	/sapdb/C11/sapdata/DISKD0078
T47	0000000300000002	NO	BLOCK READ	HIGH	5.393.063	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/sapdata/DISKD0088
T106	0000000300000002	NO	BLOCK READ	HIGH	6.536.439	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/sapdata/DISKD0084
T47	0000000300000002	NO	BLOCK READ	HIGH	359.383	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/sapdata/DISKD0083
T106	0000000300000002	NO	BLOCK READ	HIGH	2.922.237	1	RETIRED SUCCESS	1	0	SelfIO	/sapdb/C11/sapdata/DISKD0043
Converter::WriteAcknowledgement	0000000400000001	YES	BLOCK WRITE	MEDIUM	131.030	1	RETIRED SUCCESS	1	413	IO162/1	/sapdb/C11/sapdata/DISKD0032
T48	0000000300000002	NO	BLOCK READ	HIGH	4.011.764	1	RUNNING	0	407	IO193/2	/sapdb/C11/sapdata/DISKD0031
DataCache::ReadAcknowledgement	0000000500000001	YES	VECTOR READ	MEDIUM	2.662.715	1	RUNNING	0	874	IO092/1	/sapdb/C11/sapdata/DISKD0078

System view IOJOBS with the following columns:
- OWNERNAME – name of the owner
- JOBID CHARBYTE(8) - unique ID of the job, as the jobs may be displayed several times
- ASYNCHRONOUS CHAR(3) - YES/NO value indicating, whether the job is asynchronous
- OPERATION CHAR(16) - I/O operation name
- IOPRIORITY CHAR(6) - priority of the I/O (LOW, MEDIUM, HIGH)
- STARTBLOCK FIXED(10) - block where I/O starts on volume (only for volume I/O)
- BLOCKCOUNT FIXED(10) - count of consecutive blocks to read/write in this I/O
- STATE CHAR(7) - I/O state (PENDING, FAILED, SUCCESS, RETIRED FAILED, RETIRED SUCCESS)
- COMPLETEDCOUNT FIXED(10) - number of blocks transferred, in case of SUCCESS
- QUEUEID
- WORKERNAME
- PATH CHAR(256) - path to the volume or stream medium

Operation may contain:
- READ or WRITE for simple read/write on volume
- VECTOR READ or VECTOR WRITE for vector read/write on volume
- MULTI READ or MULTI WRITE for multi-vector read/write on volume
- STREAM TRANSFER for data transfer to/from stream
- OPEN STREAM, CLOSE STREAM, OPEN VOLUME or CLOSE VOLUME for open/close of volumes and streams
- INIT TAPE for initialization of the tape drive

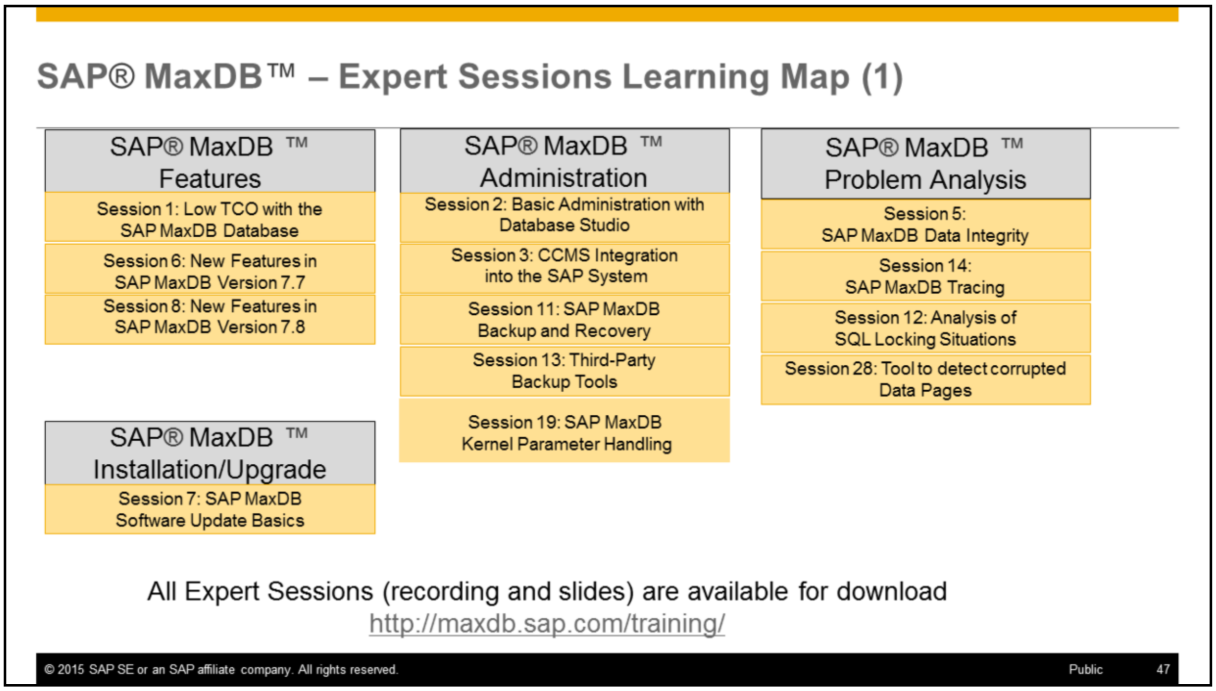
The table may contain some jobs several times, since the jobs are organized in various lists in the kernel:
- list of jobs on completion port: job will be displayed with empty task name and completion port name
- list of jobs on task: job will be displayed with task name and (if completion-port job) also with completion port name

In case several tasks wait on the same completion port, the jobs on this completion port will be displayed once without task name and once for each task with its task name. Task-specific jobs (i.e., those NOT using a completion port) are displayed only once with appropriate task name.

The nature of the information is very volatile, as I/O jobs are normally processed quickly (milliseconds). If a job hangs in the table for longer, or too many jobs are in the table, it's an indication of problem with I/O system or a potential deadlock situation in the kernel.
This system table is primarily intended for MaxDB developers.

Questions

SAP® MaxDB™ I/O Improvements 7.9 and Prefetch



SAP® MaxDB™ – Expert Sessions Learning Map (1)

SAP® MaxDB ™ Features
- Session 1: Low TCO with the SAP MaxDB Database
- Session 6: New Features in SAP MaxDB Version 7.7
- Session 8: New Features in SAP MaxDB Version 7.8

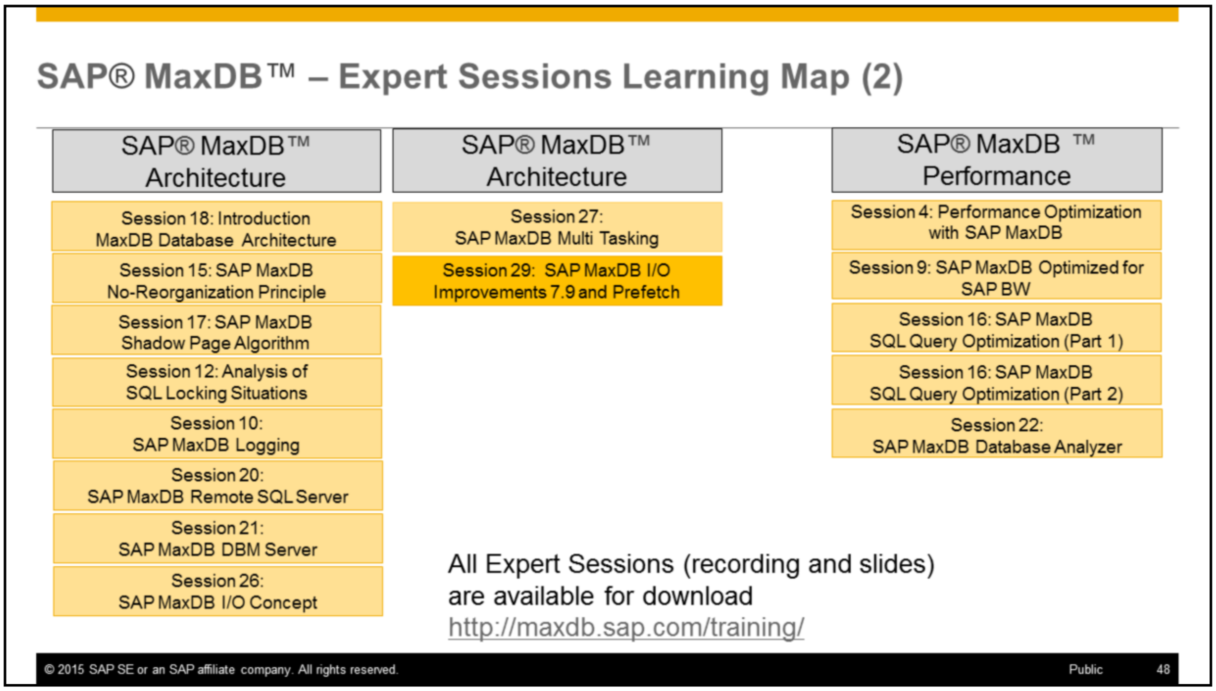
SAP® MaxDB ™ Installation/Upgrade
- Session 7: SAP MaxDB Software Update Basics

SAP® MaxDB ™ Administration
- Session 2: Basic Administration with Database Studio
- Session 3: CCMS Integration into the SAP System
- Session 11: SAP MaxDB Backup and Recovery
- Session 13: Third-Party Backup Tools
- Session 19: SAP MaxDB Kernel Parameter Handling

SAP® MaxDB ™ Problem Analysis
- Session 5: SAP MaxDB Data Integrity
- Session 14: SAP MaxDB Tracing
- Session 12: Analysis of SQL Locking Situations
- Session 28: Tool to detect corrupted Data Pages

All Expert Sessions (recording and slides) are available for download
http://maxdb.sap.com/training/

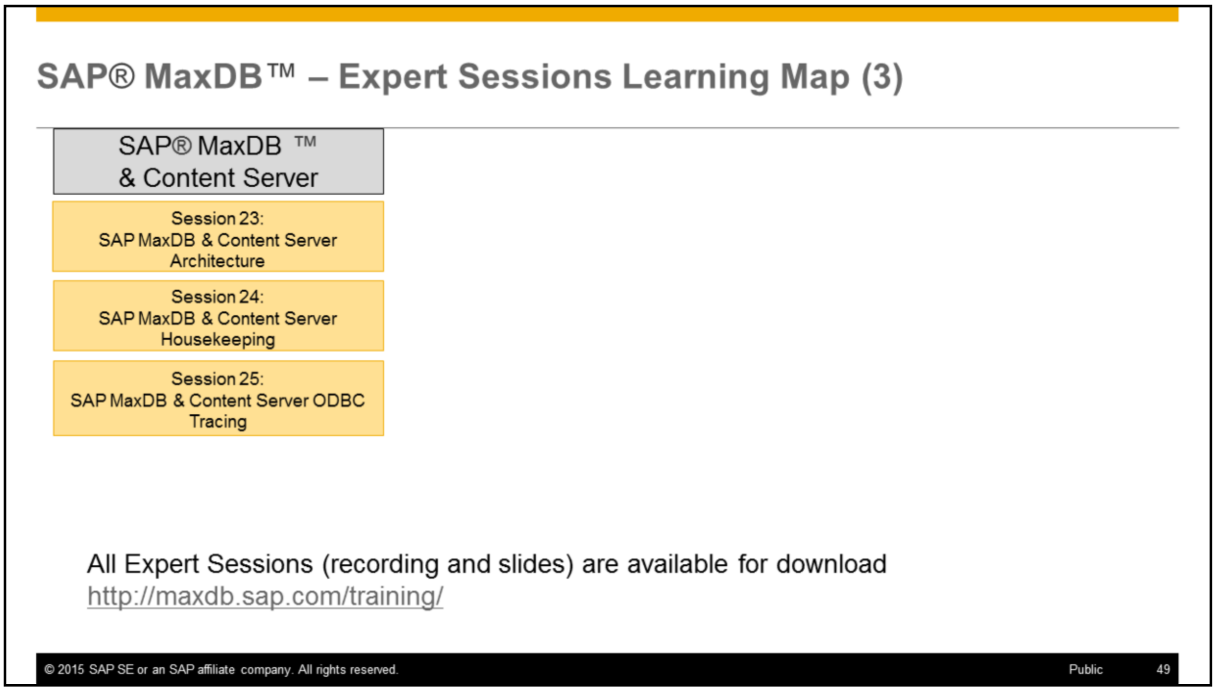
Public 47



SAP® MaxDB™ – Expert Sessions Learning Map (2)

SAP® MaxDB™ Architecture	SAP® MaxDB™ Architecture	SAP® MaxDB ™ Performance
Session 18: Introduction MaxDB Database Architecture	Session 27: SAP MaxDB Multi Tasking	Session 4: Performance Optimization with SAP MaxDB
Session 15: SAP MaxDB No-Reorganization Principle	Session 29: SAP MaxDB I/O Improvements 7.9 and Prefetch	Session 9: SAP MaxDB Optimized for SAP BW
Session 17: SAP MaxDB Shadow Page Algorithm		Session 16: SAP MaxDB SQL Query Optimization (Part 1)
Session 12: Analysis of SQL Locking Situations		Session 16: SAP MaxDB SQL Query Optimization (Part 2)
Session 10: SAP MaxDB Logging		Session 22: SAP MaxDB Database Analyzer
Session 20: SAP MaxDB Remote SQL Server		
Session 21: SAP MaxDB DBM Server		
Session 26: SAP MaxDB I/O Concept		

All Expert Sessions (recording and slides) are available for download
http://maxdb.sap.com/training/



SAP® MaxDB™ – Expert Sessions Learning Map (3)

SAP® MaxDB ™
& Content Server

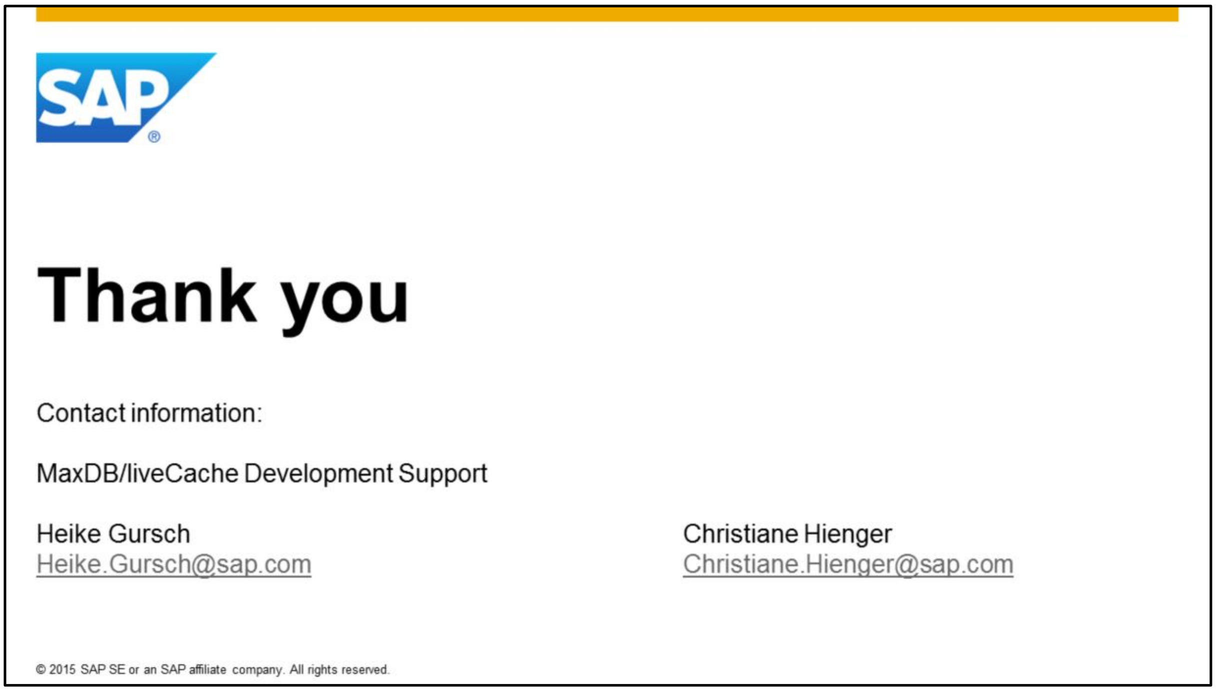
Session 23:
SAP MaxDB & Content Server
Architecture

Session 24:
SAP MaxDB & Content Server
Housekeeping

Session 25:
SAP MaxDB & Content Server ODBC
Tracing

All Expert Sessions (recording and slides) are available for download
http://maxdb.sap.com/training/

Public



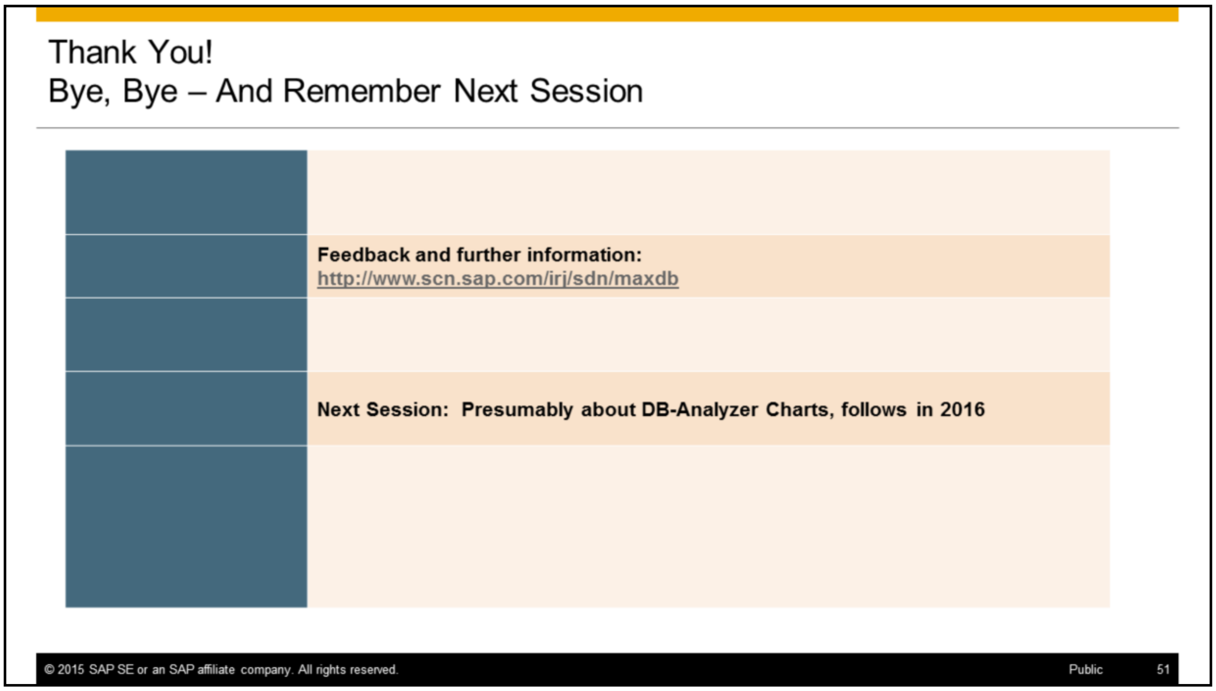
Thank you

Contact information:

MaxDB/liveCache Development Support

Heike Gursch
Heike.Gursch@sap.com

Christiane Hienger
Christiane.Hienger@sap.com

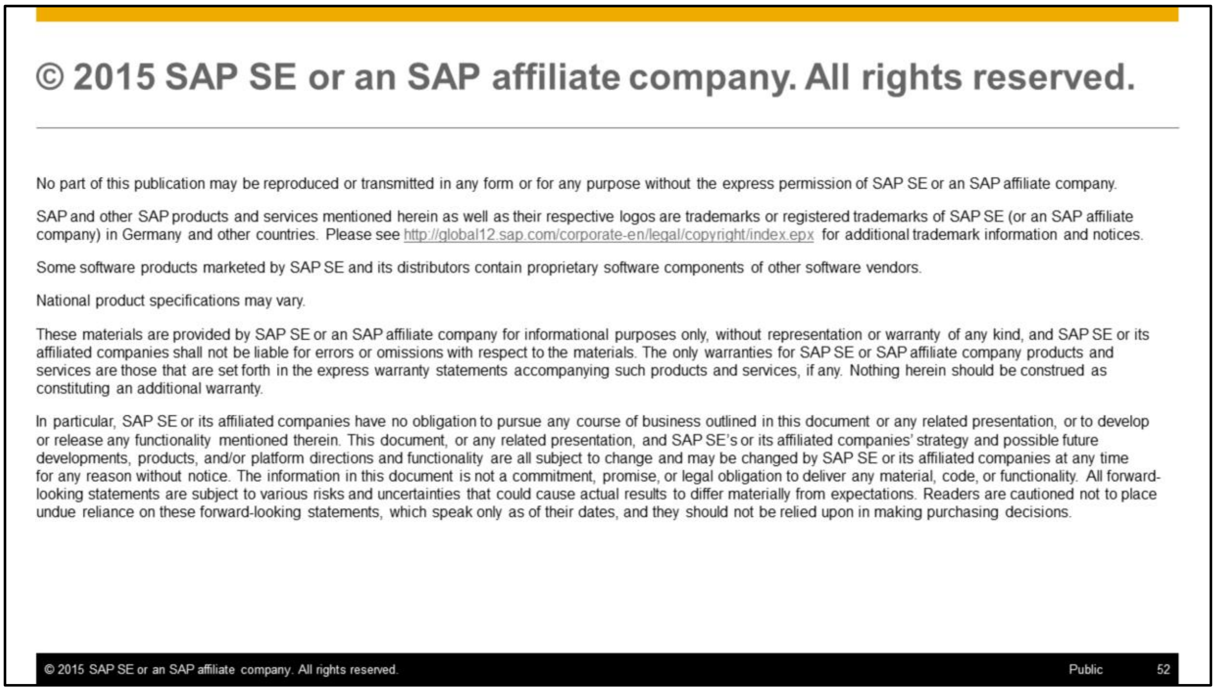


Thank You!
Bye, Bye – And Remember Next Session

Feedback and further information:
http://www.scn.sap.com/irj/sdn/maxdb

Next Session: Presumably about DB-Analyzer Charts, follows in 2016

Public 51



© 2015 SAP SE or an SAP affiliate company. All rights reserved.