Configuring Sybase ASE 15.0 For Thomson Reuters Kondor+ Suite 3.2 & 3.3

By Thomson Reuters and Sybase, Inc.
TABLE OF CONTENTS

1 About This Document
   1 Purpose
   1 Limit And Scope
1 Server Configuration
   2 Trace Flags
   2 Meta-Data Caches
   3 Disk I/O
   3 Network Communication
   3 Query Tuning
   4 Use of Parallel Query Execution
   4 Physical Memory
   6 Processors
   6 User Environment
   7 Monitoring
7 Database Configuration
   On Raw Partitions vs. File System Devices
   7 On Database Options
8 Upgrade Process Tuning
8 References

EXECUTIVE SUMMARY

Configuring ASE 15.0 for Kondor+ 3.2 & 3.3 is straightforward. The most time consuming steps are the integral parts for the general sizing exercise such as estimating the number of cores needed for the application and the DBMS, the amount of IOPS and storage space plus the amount of RAM needed.

This document details configuration parameters which have been identified as having an important role in running Kondor+ on ASE+.

Thomson Reuters and Sybase have jointly authored this document to ensure a crisp documentation as well as direct and productive recommendations.
1 ABOUT THIS DOCUMENT

1.1 Purpose

The purpose of this document is to define and elaborate on the configuration recommendations from Thomson Reuters and Sybase, Inc. for running the Kondor+ Suite 3.2 & 3.3 on Sybase ASE 15.0.

1.2 Limit and Scope

This document does not apply to ASE 12.5, however, certain recommendations remain similar.

This document does not describe all parameters affecting the operations of running Kondor+ on ASE, but focuses on the central aspects of set-up and configuration.

This document is based on testing with Kondor+ 3.2 & 3.3 and Sybase 15.0 ESD 4 + nOFF patch, in Q1 2011. However, the recommended setting should remain similar for the minor versions of both products.

Sybase 15.0 ESD 4 nOFF patch version:

- EBF 18902 : Sun Solaris Sparc 64 bits
- EBF 18903 : Sun Solaris Sparc 32 bits
- EBF 18763 : Sun Solaris x86 64 bits
- EBF 18901 : Linux 64 bits
- EBF 18900 : Aix 64 bits

2 SERVER CONFIGURATION

Below follows a list of recommended configuration options for production dataserver. This list is followed by sections detailing the configuration option by area. The sections in this chapter will include discussions on options not explicitly stated in the list.

- optimization goal: allrows_mix
- optimizer level:
  - ase_current: K+Suite 3.2 sp3 & 3.3
  - ase_default: K+Suite 3.2 sp2
- enable compatibility mode: 0
- statement cache size: <integer>
- enable literal autoparam: 1
- enable cis: 1
- number of open databases: 15
- number of open objects: 15000
- number of open indexes: 10000
- number of open partitions: 5000
- number of lock: 1000000
- number of sort buffers : 30000
- procedure cache size : 30000
- max memory: <integer>
- allocate max shared memory: 1
- stack size: 188416
- Trace flag: 1716, 4084, 15316, 15359, (15385 only on AIX)
- ASE page size : 2k
2.1 Trace Flags

Trace flags, or SET DEBUG, are used to enable specific behaviors and/or to view particular information. These are not to be taken lightly and should never be enabled without sufficient motivation. Nor should trace flags be kept in a new installation unless it is identical, serves the same purpose and is of the identical version. Issues with incorrectly enabled trace flags can cause severe issues that can be both hard and lengthy to identify and resolve.

Thomson Reuters recommend setting trace flag 1716, which in the versions concerned turns off the fix made in Change Request(CR) 590800 to raise an error message when select from a text is done from inserted or deleted table. Setting trace flag 4084 will disable console logging, setting console to nonblocking fails. Setting trace flag 15316 provide more precise ISNULL nullability. Trace flag 15359 must also be set to ensure outer join optimization are correctly applied (see Sybase cases 11560358, 11561845). Trace flag 15385 avoid a regression on max operator on AIX platform only. The max operator bug is not reproductible on other platforms. Kondor+ Suite 3.2 & 3.3 are tested and certified on ASE 15.0 with those trace flags set.

For enabling trace flags see reference [1].

2.2 Meta-Data Caches

These are sheer resources. Configuring these resources with too low a value will cause performance degradations and configuring resources with too high a value will cause a small amount of wasted memory.

Check the number of objects with the system stored procedure `sp_countmetadata`, but do note that the procedure does not include volatile objects such as cached statements, temporary tables or worktables.

Increase “number of sort buffers” to 5,000 – 10,000 to improve performance for sorts, e.g. ORDER BY-clauses, some merge joins and create index. However, a large number of sort buffers can be detrimental to UPDATE INDEX STATISTICS performance. Thus employ the following procedure for optimal performance for updating statistics:

```sql
EXEC sp_configure ‘number of sort buffers’, 0, ‘default’
go
UPDATE INDEX STATISTICS <table name> [options]
go
```

-- Reset to original value. This example assumes that value is 10,000
EXEC sp_configure ‘number of sort buffers’, 10000

Monitor resource consumption with `sp_monitorconfig “all”`.

There is a dependency between this parameter and the size of the procedure cache. See section below on the sizing of this cache.
2.3 Disk I/O

The configuration parameter "disk i/o structures" should be set high enough that lack of structures does not cause delayed IO, a good starting value is 4096. The parameters "max async i/o per engine" and "max async i/o per server" should be left at their respective defaults.

Two other parameters that should be left at their respective default values are "disable disk mirroring" and "allow sql server async i/o". Both are static and modifying them usually causes performance degradations.

Monitor the above with the system stored procedure sp_sysmon and if the same IO is delayed more than 50 times, then ASE prints a message of this to the errorlog, i.e. messages concerning this in the errorlog should be attended to.

2.4 Network Communication

The suggested network packet size is 2048, which is the default packet size in ASE 15.0.3. The following three steps are required to configure a larger packet size:

1. Allow larger packet sizes.
2. Configure memory for larger packet sizes.
3. The client application needs to request larger packet sizes.

Allow larger network packets by setting "max network packet size" to the desired value.

To configure memory for larger network packets, set "default network packet size" to the desired value, e.g. 4096. In the case that the user population is very large and the subset of users benefitting from large packets small, then use "additional network memory" as documented in reference [3].

See Kondor+ Suite documentation for configuring the application to use larger packets.

2.5 Query Tuning

Kondor+ 3.2 & 3.3 are tested and certified WITHOUT compatibility mode enabled (e.g. with the full Sybase 15 optimizer). For more information regarding compatibility mode, including benefits, side effects and restrictions see reference [4].

Kondor+ 3.2 & 3.3 are tested and certified with the default optimization goal allrows_mix. For more information on ASE 15.0 query processing see, for example, reference [5].

Kondor+ 3.2 & 3.3 are tested and certified with the ASE 15.0.3 ESD4 optimizer. To activate this last version you should set "ase_current" to the optimizer level. The default mode use ESD1 optimization only.

EXEC sp_configure 'optimizer level', 0, 'ase_current'
go

The ASE feature "statement cache" improves Kondor+ performance, and the application suite is tested and certified with this feature enabled. A reasonable initial size for statement cache is 100MB or 51200 pages. Make doubly sure that "enable literal autoparam" is enabled!
2.6 Use of Parallel Query Execution

Kondor+ 3.2 & 3.3 suite are not tested nor certified for use with parallel query execution and it should not be used with this setting. As such, the recommended setting is: \texttt{max parallel degree}: 1

2.7 Physical Memory

Physical memory, a.k.a. RAM, is usually a cost-effective way to improve system performance. It also improves performance predictability, i.e. an environment that behaves more gracefully when workloads fluctuate. The main guideline is to ensure that the host is not paging, i.e. total consumed memory fits reasonably well within physical memory. So the total amount of memory used by all processes running on the host must be less than RAM, with some headroom deducted. Allocate memory to ASE by setting \texttt{"max memory"} to the desired amount of memory, for example:

\begin{verbatim}
EXEC sp_configure 'max memory', 0, '24GB'
go
\end{verbatim}

Also make sure that ASE allocates all of \texttt{"max memory"} during start-up by setting \texttt{"allocate max shared memory"}:

\begin{verbatim}
EXEC sp_configure 'allocate max shared memory', 1
go
\end{verbatim}

The main benefit from larger amounts of memory allocated to ASE is that the data caches can be larger, holding more of the database in memory and thus avoiding costly disk I/O.

Prior to configuring data caches, attend to the meta-data caching as discussed in [Meta-Data Caches], network configuration as detailed in [Network Communication] and configuration of user and stack size, as discussed in [User Environment].

The last item on the list is \texttt{"procedure cache size"}. The procedure cache holds, among other things:

- Stored procedures.
- Triggers.
- Views.
- Defaults and rules.
- References constraints.
- Plans for all SQL executed on ASE.
- Optimizer memory for optimization.
- Cached statements

Starving the procedure cache is worse than starving the data cache since a cache miss in procedure cache involves not only the disk I/O but also re-compiling the missing SQL. Configure procedure cache by setting \texttt{"procedure cache size"} to the desired size with \texttt{sp_configure}:

\begin{verbatim}
1 http://search.sybase.com/kbx/solvedcases?id_number=11529966
\end{verbatim}
EXEC sp_configure 'procedure cache size', # of 2k pages
  go

Sybase documentation provides different formulas to estimate the size of the procedure cache. Formula based on number of concurrent users and reference call id: 12

Procedure cache size for stored procedures = Max number of concurrent users * (4 + size of largest plan) * 1.25 = 32MB
Procedure cache size for sorting = 104 * (10000 buffers) * (2048 bytes / buffer) / (4 bytes / key) = 530MB
Procedure cache size = Procedure cache size for stored procedures + Procedure cache size for sorting = 600MB

The recommended starting point for "procedure cache size" is 600MB.

Note that this is a starting point and that the configuration for other workloads may be a factor larger.

Monitor resource consumption with sp_monitorconfig "procedure cache size" and adjust if needed.

Now that procedure cache is configured it is possible to check available memory with:

EXEC sp_configure 'memory', go

This will generate an error, but as part of the error message ASE includes the available memory, so the output will include:

Msg 17411, Level 16, State 1:
Server 'rel1503esd2', Procedure 'sp_configure', Line 322:
Configuration option is not unique. Parameter Name Default ...
----------------------------- ------- ...
additional network memory ...
  .
  .
  .
(1 row affected)
An additional 163102 K bytes of memory is available for reconfiguration.
This is the difference between 'max memory' and 'total logical memory'.
(return status = 1)

This ASE only has around 160MB to use for data caches. To configure more memory increase "max memory".

"global cache partition number" should be at least equal to 2 * nb cores (rounded to next power of 2) to avoid spinlock contention in memory data cache. Example for 6 cores:

EXEC sp_configure 'global cache partition number', 16
  go

To use large I/O configure at least 20% of 16KB data cache. Example configure 4GB:

EXEC sp_poolconfig 'default data cache', '4G', '16K'
  go
2.8 Processors
The maximum number of cores an ASE can use is determined by the following parameters:

- max online engines
- number of engines at startup

Each ASE engine is an OS process named "dataserver".

Use as many engines as peak workload requires, plus some headroom. CPU Busy during peak workload should be around 70-80%.

Monitor with \texttt{sp\_sysmon}, which will display, among other things:

<table>
<thead>
<tr>
<th>Engine</th>
<th>Busy Utilization</th>
<th>CPU Busy</th>
<th>I/O Busy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engine 0</td>
<td>41.0 %</td>
<td>0.8 %</td>
<td></td>
</tr>
<tr>
<td>Engine 1</td>
<td>60.0 %</td>
<td>0.3 %</td>
<td></td>
</tr>
<tr>
<td>Engine 2</td>
<td>44.1 %</td>
<td>1.2 %</td>
<td></td>
</tr>
<tr>
<td>Engine 3</td>
<td>77.8 %</td>
<td>0.7 %</td>
<td></td>
</tr>
<tr>
<td>Engine 4</td>
<td>52.0 %</td>
<td>0.2 %</td>
<td></td>
</tr>
<tr>
<td>Engine 5</td>
<td>48.9 %</td>
<td>0.7 %</td>
<td></td>
</tr>
</tbody>
</table>

\begin{tabular}{lrr}
\hline
Summary & Total & 323.8 % & 3.8 % \\
Average & & 54.0 % & 0.6 % \\
\hline
\end{tabular}

In this case, the average CPU Busy is 54%, but this sample was not taken at peak workload.

Over-configuring the number of engines can be seen as both harmless as well as a prudent precaution. From a technical perspective too many engines cause overhead in ASE as well as for the OS.

See reference [615] for more information on monitoring ASE with \texttt{sp\_sysmon}.

2.9 User Environment
The maximum number of sessions or user connection in ASE is determined by one parameter: "number of user connections". Set this to an appropriately high value since shortage will cause connection failure. The recommended starting point is 1000 (about 5 connections by connected application/server). Also increase the sessions' stack size to the recommended value. Kondor+ has been tested and certified with ASE 15.0 with this value:

\begin{verbatim}
EXEC sp\_configure 'stack size', 188416
GO
\end{verbatim}
2.10 Monitoring

Continuously monitoring a system’s resource consumption and workload is an integral part of the DBA profession. Sybase provides many options to choose from. The Monitoring Tables are widely used and much appreciated by administrators world-wide.

However, monitoring a system adds workload. Therefore, be prudent in which monitoring options are enabled. Certain aspects, and thus monitoring tables and options, are only needed during specific circumstances such as root cause analysis during problem resolution.

See reference [7] for more information on monitoring ASE with Monitoring Tables. Another tool that provides performance data is sp_sysmon. It also includes suggestions for interpreting its output and provides recommendation on possible configuration changes.

3 DATABASE CONFIGURATION

This section holds recommended good practices to employ when setting up and configuring databases, as well as discussions providing backgrounds for these recommendations: Use raw partition devices and not file system. Always separate data and log. Separate data and log for tempdb as well!

Separating data and log improves performance as well as manageability. There is no argument for mixing data and log.

3.1.1 On raw partitions vs. file system devices

File system devices are extremely common and raw devices often seen as legacy; an approach taken only in the olden days for demanding systems. This statement does not take cost of setup, management and monitoring into account, nor does the equation include performance considerations.

In no case are file system devices as fast for safe IO as raw devices. And rarely is configuration of file system devices and virtual memory included in the cost of managing storage of mission critical data shared by many users and applications.

File system devices require testing and configuration to ensure they are properly configured for database management systems. This includes journaling, block sizes and other applicable options:

• Journaling is not required for other than FS internal physical consistency as the DBMS does its own logging.
• Mismatch between FS and DBMS block sizes can cause OS overhead and performance degradations.

There are also CPU efficiency and consumption to take into account as the extra processing of asynchronous IO through file system require more CPU. Therefore hardware resources needs to be reserved, i.e. not used by Sybase ASE, for IO on file system devices.
Recommendations:

• Use raw devices for all Sybase ASE devices. Quick IO on Veritas FS should be considered as raw in this context.
• In the event that raw devices are not an option, potentially due to OS or storage system limitations then use UFS with Direct IO.
• Do not use UFS without Direct IO. This includes ZFS, which is thus not a viable option.
• Make sure dsync is always set to “off” and direction is set to on all filesystem devices (use sp_helpdevice to see current setting of all devices)
• tempdb is a specific case. We do not need durable write. This allows the configuration of tempdb devices on ramdisk or pure file system devices (use system caching). Configure a specific tempdb ‘sa’ for all admin activities.
• For RAID disk storage, use RAID -1 + 0 capability. It offers both performance and protection from disk failures. Sybase log device should be on separate physical and logical devices.

3.1.2 On database options

Kondor+ recommends the database option "select into/bulkcopy/pllsort" to be set only on the tempdb databases. This ensures that the incremental transaction log backup strategy is not compromised. If there are some external code which requires it, it is possible to enable this parameter on all databases, but this removes a safety net.

Notes: On the contrary, it is recommended to set the database option "select into/bulkcopy/pllsort" on all databases during K+ upgrade when running KplusRecompute after upgrading the databases to improve execution performance.

An option that is commonly set on development systems is "trunc log on chkpt". However, this should never be set on production systems since it causes excessive disk IO. To make sure that the log does not get full, use scheduled log backups (or just truncate) complemented with thresholds to handle workload variations.

4 UPGRADE PROCESS TUNING

The upgrade process execute large IO (Massive update, alter, create index) on the database. To increase upgrade performance we recommend:

- To increase Log IO size to use 16K
  EXEC sp_poolconfig 'default data cache', '10G', '16K'
  Go
- To increase the ULC to 20480
  EXEC sp_configure 'user log cache size', 20480
  go
  Don’t forget to revert to the original production dataserver values.
5 REFERENCES
