Utilizing Surveillance for Sybase ASE 15.0 Migration

Sybase ASE 15 is a significant upgrade for Sybase customers, and customers are expecting to spend more time testing their applications to ensure getting your environment up and running as quickly as possible. Bradmark recommends installing Bradmark's Surveillance for Sybase at the beginning to benchmark the current environment to see the cause of any outage or slow performance before and after the migration process. By monitoring critical system metrics before migration, Surveillance will help customers prepare and speed up the upgrade process. The following is a review of tasks that can be performed by using Surveillance. This document is based on a Sybase published whitepaper entitled, "Planning Your ASE 15.0 Migration: Tips, Tricks, Gotcha's & FAQ" that provides recommended tasks to be performed for a migration to ASE 15.

Introduction & Pre-Upgrade Planning

Sybase Recommendation

Reduced 32-bit Platform Availability:

- Sybase has reduced the number of platforms with 32-bit binaries so many customers will be moving to 64-bit versions of ASE. The 64-bit platforms allow more memory to be used.
- If utilizing more memory with ASE, Sybase recommends monitoring monOpenObjectActivity for higher logical IO and CPU utilization. This can result from previous table scans that required physical IOs but now fit into data cache with all logical IOs. ASE does not need to yield the CPU to wait for the physical IOs, so processes involved in logical memory table scans will use their full time slice, yield the CPU and then immediately jump on the run-able queue. This could result in an immediate and dramatic increase in CPU utilization and a resulting drop in application performance.
- Sybase recommends running this query:

```
select *
   from master..monOpenObjectActivity
   where DBID not in (1, 2, 3) -- add additional tempdb dbid's if using multiple tempdbs
     and UsedCount > 0
     and IndexID = 0
order by LogicalReads desc, UsedCount desc
```

Surveillance Solution

Surveillance monitors the monOpenObjectActivity in the "Open Objects" real-time window. User and temp objects are shown on separate tabs or can be viewed together on the "All Objects" tab. Data from the windows can be saved to csv or xls files for later use. You could also save the MDA_OPENOBJ collection in the local repository and forward it to a centralized repository. This query could then be run from the repository as:

```
select *
   from <repository database>..SYBASE_MDA_OPENOBJ
   where DBID not in (1, 2, 3) -- add additional tempdb dbid's if using multiple tempdbs
     and C_USEDCOUNT > 0
     and C_INDEXID = 0
order by C_LOGICALREADS desc, C_USEDCOUNT desc
```

Surveillance stores more columns in SYBASE_MDA_OPENOBJ than the original monOpenObjectActivity for calculations of rates and deltas, and columns to hold the host and entity names. To get the same output as the Sybase query, substitute the select * to:

```
select c_DBID, c_OBJECTID, c_INDEXID, c_DBNAME, c_OBJECT_NAME, c_LOGICALREAD,
c_PHYSICALREADS, c_APFREADS, c_PAGESREAD, c_PHYSICALWRITES, c_PAGESWRITTEN,
c_ROWSINSERTED, c_ROWSDELETED, c_ROWSUPDATED, c_OPERATIONS,
c_LOCKREQUESTS, c_LOCKWAITS, c_OPTSELECTCOUNT, c_LASTOPTSELECTDATE,
c_USEDCONT, c_LASTUSEDDATE
   from …
   where …
   and C_ENTITY_NAME = "<Surveillance ASE name>"
order by …
```

The central repository can hold the data for many ASE servers. The example shows how to restrict the query to a single ASE instance.

Preparing for the Upgrade

Sybase Recommendation

Application I/O Profile

One item that may help is to create a quick IO profile of the application in 12.5.x and 15.0.x. This can be done by collecting MDA monOpenObjectActivity and monSysWaits data for a representative period of time. For monOpenObjectActivity data, you may wish to collect two sets of data - one for application databases and another for temp databases. If you compare the monOpenObjectActivity data with ASE 15.0, you may notice the following:

- o A huge increase in table scans on one or more tables (see earlier discussion on 32 to 64 bit for logic). This could be due to an optimization issue or just a move to a merge-join instead of a nested loop join using an index.
- o A large decrease in IOs in tempdb - particularly when joins between two or more temp tables are involved. This likely is due to merge join.
- o A significant drop in the UsedCount column for particular indexes (when IndexID > 1). Likely this is the result of missing statistics if the index contains more than one column.

While this may not point out the exact queries affected (discussed later), it can help reduce the effort to find the queries to just those involving specific tables. Keep in mind that the observation periods should be during the same relative loading to ensure that the results are comparable.

Surveillance Solution

Surveillance monitors the monOpenObjectActivity in the "Open Objects" real-time window. User and temp objects are shown on separate tabs or can be viewed together on the "All Objects" tab. Surveillance monitors the monSysWaits in the "ASE Server Health" real-time window's "Wait Statistics" tab. The Wait Class and Top Wait Events are displayed for the current wait time and the overall wait time.

Surveillance has a collection called MDA_OPENOBJ for monOpenObjects and two collections called MDA_SYSTIMEDEVENTS and MDA_SYSWAIT_CLASS for monSysWaits. These collections can be saved in the local repository and forwarded to a centralized repository. Queries could be run on the centralized repository to compare the difference between 12.5 and 15.0 to look for the conditions mentioned above.

Sybase Recommendation

Check Database Space:

- The pre-upgrade will check the available space in each database, but verify beforehand.
- For databases under 10 GB, try to have at least 25% free space.
- Master should have 50% free space, and sybsystemprocs needs room for new stored procedures.
- Check the other system databases too, for 2K pagesize, about 4MB is needed for catalog expansion.
- The system segment may need more space beyond system tables changes if using DBExpert to compare query plans or if you plan to use sysquerymetrics with ASE 15.0.

Backup and clear the transaction log as part of the upgrade procedure.

Surveillance Solution

Surveillance monitors overall database space, displayed in the "Databases" real-time window. Detailed segment and object information can be seen by right-clicking on a database name in the list, and selecting the Database Configuration, Database Space Utilization, or Database Audit options. Real-time windows for "Segment Usage" and "Device Space" are also available.

Sybase Recommendation

Increased HW Resource Requirements:

- ASE 15.0 will need additional memory due to new in-memory sorting and grouping algorithms.
- Additional memory required in the data cache with tempdb
- It may require additional procedure cache space for auxiliary scan buffers.
- In ASE 15.0.1, some new MDA tables will help monitor this from a procedure cache standpoint:
    - o monProcedureCacheModuleUsage
    - o monProcedureCacheMemoryUsage

Both tables contain a High Water Mark (HWM) column so that the max memory usage for particular activities can be observed on a procedure cache module basis.

- In addition, due to the increased number of first class objects implemented within the server to support encrypted columns, partitioned tables and other new features, you may need to increase the number of open objects/open indexes. This particularly is likely to be true if the ASE server contains a number of databases.

Sybase recommends that you monitor the metadata cache requirements including object and index descriptors.

Surveillance Solution

Surveillance monitors memory and cache. The "Memory Configuration" real-tim window shows the data cache and memory configuration values. The "Cache Activity" real-time window shows a summary of procedure cache and all data caches. Each cache has a detailed view of the objects in cache. A User-Defined Collection (UDC) can collect data on the HWM and statistics in monProcedureCacheModuleUsage and monProcedureCacheMemoryUsage.

Surveillance monitors and alerts on system resource usage for parameters such as open objects, open indexes, meta-data caches, and aux scan descriptors as recommended by Sybase.

Sybase Recommendation

Preparing for Post-Upgrade Monitoring for QP Changes:

- Especially post-upgrade, the MDA tables along with sysquerymetrics will be heavily used to find affected queries as well as changes in over system resource consumption.
- You should also familiarize yourself with the sysquerymetrics table, sp_metrics system procedure and the associated techniques to exploit this new feature to rapidly identify underperforming queries.

Surveillance Solution

Surveillance monitors sysquerymetrics in the "Query Metrics" real-time window, available when monitoring ASE 15.0 or higher. The QUERY_METRICS collection can be saved in the local repository for flashback (historical data viewing through the real-time interface) and can be forwarded to a central repository.

Query Processing Changes

Sybase Recommendation

Literal Parameterization and Statement Cache:

Since we are creating a dynamic procedure, this counts as an 'open object'. You will have to increase the configuration 'number of open objects' accordingly.

Surveillance Solution

Surveillance monitors 'number of open objects' and similar resources in the "Monitored Resources" real-time window. You can be alerted if these resources get close to their maximum value.

Sybase Recommendation

Monitor Index Usage:

Use monOpenObjectActivity to look for table scans as well as index efficiency/usage.

- ASE 12.5.3 didn't include the DBName and ObjectName fields. Regardless, this table has a wealth of information that can be useful during a migration. First, note that it is at the index level - consequently it is extremely useful to detect changes in query behavior with minimal impact on the system.
- You will need samples from a 12.5.x baseline system and the ASE 15.0 migration system under roughly the same query load.
- The "UsedCount" column is perhaps the most important for index usage. This counter keeps track of each time an index is used as a result of the final query optimization and execution.
  - Find table scans using the query:

```
select * from master..monOpenObjectActivity
  where DBID not in (1, 2, 3) -- add additional tempdb dbid's if using multiple tempdbs
    and UsedCount > 0
    and IndexID = 0
  order by LogicalReads desc, UsedCount desc
```

Note that not all table scans be avoided - the key is to look for significant increases in table scans.

Surveillance Solution

Surveillance monitors monOpenObjectActivity in the "Open Objects" real-time window. User and temp objects are shown on separate tabs or can be viewed together on one tab. Data from the windows can be saved to csv or xls files for later use. You could also save the MDA_OPENOBJ collection in the local repository and forward it to a centralized repository. This query could then be run from the repository as:

```
select * from <repository database>..SYBASE_MDA_OPENOBJ
  where DBID not in (1, 2, 3) -- add additional tempdb dbid's if using multiple tempdbs
    and USEDCOUNT > 0
    and INDEXID = 0
order by LOGICALREADS desc, USEDCOUNT desc
```

Surveillance stores more columns in SYBASE_MDA_OPENOBJ than the original monOpenObjectActivity for calculations of rates and deltas, and columns to hold the host and entity names. To get the same output as the Sybase query, substitute the select * to:

```
select c_DBID, c_OBJECTID, c_INDEXID, c_DBNAME, c_OBJECT_NAME, c_LOGICALREAD,
c_PHYSICALREADS, c_APFREADS, c_PAGESREAD, c_PHYSICALWRITES, c_PAGESWRITTEN,
c_ROWSINSERTED, c_ROWSDELETED, c_ROWSUPDATED, c_OPERATIONS, c_LOCKREQUESTS,
c_LOCKWAITS, c_OPTSELECTCOUNT, c_LASTOPTSELECTDATE, c_USEDCONT, c_LASTUSEDDATE
   from …
  where …
  and C_ENTITY_NAME = "<Surveillance ASE name>"
order by …
```

The central repository can hold the data for many ASE servers. The example shows how to restrict the query to a single ASE instance.

Sybase Recommendation

Monitor Statement Execution:

Use monSysSatement, monSysSQLText, and monSysPlanText to compare SQL statement execution times between 12.5.x and 15.0.x. The high level steps are:

1. Configure the statement pipe, sql text pipe, and statement plan text pipe as necessary for the ASE 12.5 server.
2. Create a temporary repository in tempdb
3. Repeat for monSysSQLtext and monSysPlanText as desired
4. Begin a monitoring process that once per minute inserts into the tables created above from the respective MDA tables
5. Execute one module of the application to be tested
6. Stop the application and halt the monitoring
7. bcp out the MDA collected data from tempdb
8. Repeat steps 1-6 for ASE 15.0
9. Create a second set of tables in the scratch database – one each for ASE 15.0 and 12.5. For example: mdaSysStmt_125 and mdaSysStmt_150
10. Load the tables from the collected information – either by bcp-ing back in or via insert/select
11. Run queries against the 12.5 and 15.0 statements to see which ones run faster or slower

The only gotcha with this technique is that it is only accurate within 100ms – the reason is that it is based on the CPU ticks length within ASE, which defaults to 100ms. Statements that execute less than 100ms will show up as 0.

Surveillance Solution

Surveillance monitors SQL statements in the "Current Executing Statements" and "Recently Executed Statements" real-time windows. The query plan is viewable for each statement if it's still available in memory. The MDA_SYSSTMT_PIPE, MDA_SQLTEXT_PIPE, and MDA_SQLPLAN_PIPE collections can be stored in the local repository and forwarded to a central repository. Surveillance can be the monitoring process mentioned in Step 4 above. The collected data can be stored in a repository eliminating the need for creating the table(s) in Step 2 and 3 and using bcp in Step 7, and the repeated steps for ASE 15.0. Queries on the statements can be performed on the repository.

Sybase Recommendation

Since the monitoring captures all statements from all users, the next step is to isolate out each of the specific user's queries and re-normalize using a new identity column. For example, the following query could be used to build the new table to be used to compare query execution:

```
select exec_row=identity(10), SPID, KPID, DBID, ProcedureID,
       ProcName, PlanID, BatchID, ContextID, LineNumber,
       CpuTime, WaitTime, MemUsageKB, PhysicalReads, LogicalReads,
       PagesModified, PacketsSent, PacketsReceived,
       NetworkPacketSize, PlansAltered, RowsAffected, ErrorStatus,
       StartTime, EndTime
  into monSysStmt_150
  from mdaSysStmt_150
  where SPID = 123
       and KPID = 123456789
  order by row_id
go
create unique index exec_row_idx
         on monSysStmt_150 (exec_row)
go
```

Surveillance Solution

To run from the Surveillance repository and copy to the new table, run the query as:

```
select exec_row=identity(10), c_SPID, c_KPID, c_DBID, c_PROCEDUREID,
       c_PROCNAME, c_PLANID, c_BATCHID, c_CONTEXTID, c_LINENUMBER,
       c_CPUTIME, c_WAITTIME, c_MEMUSAGEKB, c_PHYSICALREADS, c_LOGICALREADS,
       c_PAGESMODIFIED, c_PACKETSSENT, c_PACKETSRECEIVED,
       c_NETWORKPACKETSIZE, c_PLANSALTERED, c_ROWSAFFECTED, c_ERRORSTATUS,
       c_STARTTIME, c_ENDTIMe
  into monSysStmt_150
  from <repository database>..SBASE_MDA_SYSSTMT_PIPE
  where SPID = 123
       and KPID = 123456789
  order by row_id
go
create unique index exec_row_idx
         on monSysStmt_150 (exec_row)
go
```

Sybase Recommendation

Consequently, if the same exact sequence of test statements is issued against both servers, the exec_row columns should match. Consider the following query:

```
-- Get a list of SQL statements that executed slower in 15.0 compared to 12.5:
select f.exec_row, f.BatchID, f.ContextID, f.ProcName, f.LineNumber,
        CPU_15=f.CPUTime, CPU_125=t.CPUTime,
        Wait_15=f.WaitTime, Wait_125=t.WaitTime,
        Mem_15=f.MemUsageKB, Mem_125=t.MemUsageKB,
        PhysIO_15=f.PhysicalReads, PhysIO_125=t.PhysicalReads,
        LogicalIO_15=f.LogicalReads, LogicalIO_125=t.LogicalReads,
        Writes_15=f.PagesModified, Writes_125=t.PagesModified,
        ExecTime_15=datediff(ms,f.StartTime,f.EndTime)/1000.00,
        ExecTime_125=datediff(ms,t.StartTime,t.EndTime)/1000.00,
        DiffInMS= datediff(ms,f.StartTime,f.EndTime)-
        datediff(ms,t.StartTime,t.EndTime)
into #slow_qrys
from monSysStmt_150 f, monSysStmt_125
where f.exec_row = t.exec_row
   and (datediff(ms,f.StartTime,f.EndTime) > datediff(ms,t.StartTime,t.EndTime))
order by 20 desc, f.BatchID, f.ContextID, f.LineNumber
```

Surveillance Solution

To run from the Surveillance repository and copy to the new table, run the query as:

```
select f.exec_row, f.c_BATCHID, f.c_CONTEXTID, f.c_PROCNAME, f.c_LINENUMBER,
        CPU_15=f.c_CPUTIME, CPU_125=t.c_CPUTIME,
        Wait_15=f.c_WAITTIME, Wait_125=t.c_WAITTIME,
        Mem_15=f.c_MEMUSAGEKB, Mem_125=t.c_MEMUSAGEKB,
        PhysIO_15=f.c_PHYSICALREADS, PhysIO_125=t.c_PHYSICALREADS,
        LogicalIO_15=f.c_LOGICALREADS, LogicalIO_125=t.c_LOGICALREADS,
        Writes_15=f.c_PAGESMODIFIED, Writes_125=t.c_PAGESMODIFIED,
        ExecTime_15=datediff(ms,f.c_STARTTIME,f.c_ENDTIME)/1000.00,
        ExecTime_125=datediff(ms,t.c_STARTTIME,t.c_ENDTIME)/1000.00,
        DiffInMS= datediff(ms,f.c_STARTTIME,f.c_ENDTIME)-
        datediff(ms,t.c_STARTTIME,t.c_ENDTIME)
into #slow_qrys
from monSysStmt_150 f, monSysStmt_125
where f.exec_row = t.exec_row
   and (datediff(ms,f.c_STARTTIME,f.c_ENDTIME) > datediff(ms,t.c_STARTTIME,t.c_ENDTIME))
order by 20 desc, f.c_BATCHID, f.c_CONTEXTID, f.c_LINENUMBER
```

Sybase Recommendation

Of course it always nice to tell the boss how many queries were faster – which is quite easy to accomplish with the above – simply swap the last condition with a 'less than' to get:

```
-- Get a list of SQL statements that executed faster in 15.0 compared to 12.5:
select f.exec_row, f.BatchID, f.ContextID, f.ProcName, f.LineNumber,
        CPU_15=f.CPUTime, CPU_125=t.CPUTime,
        Wait_15=f.WaitTime, Wait_125=t.WaitTime,
        Mem_15=f.MemUsageKB, Mem_125=t.MemUsageKB,
        PhysIO_15=f.PhysicalReads, PhysIO_125=t.PhysicalReads,
        LogicalIO_15=f.LogicalReads, LogicalIO_125=t.LogicalReads,
        Writes_15=f.PagesModified, Writes_125=t.PagesModified,
        ExecTime_15=datediff(ms,f.StartTime,f.EndTime)/1000.00,
        ExecTime_125=datediff(ms,t.StartTime,t.EndTime)/1000.00,
        DiffInMS= datediff(ms,f.StartTime,f.EndTime)-
                datediff(ms,t.StartTime,t.EndTime)
into #fast_qrys
from monSysStmt_150 f, monSysStmt_125
where f.exec_row = t.exec_row
   and (datediff(ms,f.StartTime,f.EndTime) < datediff(ms,t.StartTime,t.EndTime))
order by 20 desc, f.BatchID, f.ContextID, f.LineNumber
```

Surveillance Solution

To run from the Surveillance repository and copy to the new table, run the query as:

```
select f.exec_row, f.c_BATCHID, f.c_CONTEXTID, f.c_PROCNAME, f.c_LINENUMBER,
        CPU_15=f.c_CPUTIME, CPU_125=t.c_CPUTIME,
        Wait_15=f.c_WAITTIME, Wait_125=t.c_WAITTIME,
        Mem_15=f.c_MEMUSAGEKB, Mem_125=t.c_MEMUSAGEKB,
        PhysIO_15=f.c_PHYSICALREADS, PhysIO_125=t.c_PHYSICALREADS,
        LogicalIO_15=f.c_LOGICALREADS, LogicalIO_125=t.c_LOGICALREADS,
        Writes_15=f.c_PAGESMODIFIED, Writes_125=t.c_PAGESMODIFIED,
        ExecTime_15=datediff(ms,f.c_STARTTIME,f.c_ENDTIME)/1000.00,
        ExecTime_125=datediff(ms,t.c_STARTTIME,t.c_ENDTIME)/1000.00,
        DiffInMS= datediff(ms,f.c_STARTTIME,f.c_ENDTIME)-
        datediff(ms,t.c_STARTTIME,t.c_ENDTIME)
into #fast_qrys
from monSysStmt_150 f, monSysStmt_125
where f.exec_row = t.exec_row
   and (datediff(ms,f.c_STARTTIME,f.c_ENDTIME) < datediff(ms,t.c_STARTTIME,t.c_ENDTIME))
order by 20 desc, f.c_BATCHID, f.c_CONTEXTID, f.c_LINENUMBER
```

Sybase Recommendation

Stored Procedure Profiling:

- The monSysStatement table reports the statement level statistics at a line-by-line basis - which in a sense is even handy as when a proc execution is slower, the exact line of the procedure where the problem occurred is easily spotted. Aggregating it to a procedure level for profiling is a bit trickier.
- Sybase provides a stored procedure on page 58 of the migration white paper.

Surveillance Solution

To run from the Surveillance repository

```
select exec_row=identity(10), c_SPID, c_KPID, c_DBID, c_ProcedureID, c_PlanID,
     c_BatchID, c_ContextID, c_LineNumber, c_CpuTime, c_WaitTime,
     c_MemUsageKB, c_PhysicalReads, c_LogicalReads, c_PagesModified,
     c_PacketsSent, c_PacketsReceived, c_NetworkPacketSize,
     c_PlansAltered, c_RowsAffected, c_ErrorStatus, c_StartTime, c_EndTime
into #t_exec_by_spid
from SYBASE_MDA_SYSSTMT_PIPE
where (((@startDate is null) and (@endDate is null))
        or ((c_StartTime >= @startDate) and (@endDate is null))
        or ((@startDate is null) and (c_EndTime <= @endDate))
        or ((c_StartTime >= @startDate) and (c_EndTime <= @endDate))
        )
order by c_SPID, c_KPID, c_BatchID, row_id
```

Sybase Recommendation	**Using Sysquerymetrics:** - The sysquerymetrics provides a number of performance metric columns including logical io's, cpu time, elapsed time, etc. - Note however, that this displays the query text and not the query plan. Consequently you first identify the slow queries in 15.0 and then compare the plans. Another useful technique is to use sysquerymetrics to support regression testing after changing the optimization goals, parallel resources/partitioning for a table or employing other ASE 15.0 features such as function-based indices.
Surveillance Solution	Surveillance monitors sysquerymetrics in the "Query Metrics" real-time window, available when monitoring ASE 15.0 or higher. The QUERY_METRICS collection can be saved in the local repository for flashback (historical data viewing through the real-time interface) and can be forwarded to a central repository. The data in the central repository can be queried to compare different executions.

Sybase Recommendation	**Procedure Cache Growth** Watch the procedure cache using the new MDA tables for this to observe if the increased proc cache requirements for sorting (due to merge sorts, etc.) impact the number of procs cached and increase as necessary.
Surveillance Solution	Surveillance monitors the procedure cache in the "Cache Activity" and "Procedures" real-time windows. The "Cache Activity" real-time window shows statistics such as requests, loads, writes, and stalls. The "Procedures" real-time window shows activity for stored procedures currently loaded in cache, including object type, memory, and object owner.

Phone: (800) 621-2808
Outside the U.S.: (713) 621-2808
Fax: (713) 621-1639
On the Web: www.bradmark.com