

A Preventive Maintenance Regimen for ASE 15

Jeffrey Garbus

Soaring Eagle Consulting

Preventative Maintenance Regimen

In order to make sure your Sybase Adaptive Server Enterprise (ASE) 15 server keeps its enormously successful uptime success rate, an Adaptive Server must be properly maintained to stay in good working order. This article discusses the common tasks need to be performed in order to ensure your uptime.

The different types of maintenance we describe will be organized based upon scope: at the server, database, or object level. Some maintenance affects the entire server; for example, memory is a shared resource among all the tasks running on an Adaptive Server, and sharing or configuring it may affect every session accessing the server. Other tasks take place at the database level. For example, the database is the server's unit of backup and recovery, so we perform database backups (making a copy of the database) and other types of care and maintenance one database at a time. Finally, at the table level (the table is what contains your data), you do things to help ensure database performance; that is, fast access to the data (for example, updating statistics or running an index reorg).

Server Level Maintenance

Part of Systems Administration is disaster preparedness. Under some circumstances, you may need to reinstall your entire server. Or, more cheerfully, you may want to duplicate a server (for example, during a hardware upgrade, which can be good practice for a physical server meltdown). Given this, it becomes necessary to include scripts that can be used to rebuild the entire server as part of the server level maintenance routine. This need not be a daily task, but there are plenty of shops that extract this information from system tables on a nightly basis (via a cron job) against just such a potential calamity.

Activity Monitoring

Adaptive Server comes with various tools to monitor the activity on the server. Regular monitoring of the server identifies situations where the system starts to perform badly because of misconfiguration, and can be used to identify any bottlenecks in the server. The act of

monitoring the server's performance on a repeated basis throughout a day can also be used to identify performance peak times for predictive analysis.

sp_sysmon

The sp_sysmon stored procedure produces low-level system utilization information. This data is often more useful for identifying bottlenecks or periods of intensive server use.

The output of sp_sysmon is extensive, and learning to read and interpret all of its results requires a great deal of practice. Administration can be as much art as science; there may be many sets of correct values for the server configuration parameters. Precise meanings of the various sections and subsections is described in the *Performance and Tuning Guide* of the ASE documentation set, and is a great place to start if you suspect server performance issues.

Use of sp_sysmon contributes some overhead, but it is difficult to measure. The procedure should be run before and after changing system parameters, adding or removing caches, disks, or engines within the server. This will allow the administrator to gather a before and after picture of any changes to the server and their effect on performance. Running the procedure before and after the installation of new applications to the server can help identify the impact the new application has on the overall server. And of course, this should be run periodically during peak times to see what kind of load is on your systems.

Note: You can pick any monitoring interval. It can be useful to run (for example) 5 minutes on, 5 minutes off (many shops run this as a matter of course). If you have a lot of processing you want to understand, and the processing runs 30 minutes, go ahead and run sp_sysmon for 30 minutes.

There are currently many monitoring tools on the market which will pick this information up & put it into a data warehouse for analysis.

MDA Tables

ASE has optional, configurable memory-only tables that can be used to monitor performance, users, and a variety of other tasks. This monitors some of the system information available in sp_sysmon, but also monitors application information (for example, what queries are running, how often, and how long).

This is something you should consider installing if you are planning on tracking system and application performance long-term, but note that enabling all of the MDA tables on a very busy system might have a performance impact at peak.

What to Monitor

When using the sp_sysmon output, keep an archive of the outputs and try to at least gather information about Engine Busy Utilization (in the Kernel Utilization section), and Disk I/O

Management. Try to gauge maximum, minimum and average values for these readings during peak and off-peak times. This will provide a frame of reference, for what is "normal" on the server, and make abnormalities easier to identify. Also try to gather information on User/Process connections (Worker Process Management and Task Management sections). Metrics like the total number of connections at a particular time of day, average connections per user, and the average length of time blocked by other users (some of which you will have to calculate yourself) can help gauge the normal work load attributed by users to the server.

As an alternative, Adaptive Server also comes with a Monitor Server that collects real-time data on the servers and can hold store that data to assist in discovering bottlenecks. You may write your own client tools to display and interpret these data, or you may use the monitor clients of Sybase Central.

Monitor & manage the System Errorlog

By default, ASE will place an errorlog file in the directory in which you started the server (usually the install directory). In addition to boot time messages identifying which resources came up, their success, and the amount of elapsed time, any major errors or anomalies from the server will be recorded in the errorlog; this can include anything from abnormal disconnections from the server, to major system halting errors will be present in the errorlog. Some third-party tools will notify you of additions to the errorlog. Many shops write tasks that monitoring the errorlog on a continual (say, once per minute) basis. Note that SQL error messages will list a value of *severity* that will indicate the relative importance of the error (the higher, the more severe).

Make sure that you both monitor and maintain (i.e. periodically truncate) the Errorlog.

We recommend continuous monitoring of the log (i.e. write or borrow a script that does this frequently).

Resource Verification

Another good practice for administrators is to verify that all resources in the server are allocated properly and that all configurable parameters are optimally configured. Adaptive Server allocates memory for most resources and parameters at startup, and most of the server options require some amount of memory to maintain. If any resources are over-allocated then memory that might be better used in cache by the Adaptive Server is essentially wasted. If any options or parameters are under-allocated then Adaptive Server may not perform as well. Check resources allocated vs. in use; are any configured parameters drastically out of range? For example:

- Concurrent connections at 100, with 25 the most used at any one time?
- Device connections at 50, with only 5 devices defined?

- Remote access defined but not in use?
- Are all of your resources on line?
- Did all of the disks start up?
- Are all of the engines on line?

This is a good task to perform weekly to start, and then move to monthly or quarterly based upon growth patterns you see.

sp_monitorconfig

Here's one that a lot of folks never learned about: You can use the sp_monitorconfig stored procedure to keep track of the utilization of many of the system parameters that you've tuned.

This stored procedure gives you aggregate information on many configuration parameters, letting you know if you are over- or under-configured.

Run it periodically (at least monthly) in order to validate your system configuration.

Software Maintenance

Sybase maintains its software with releases and point releases, (called "EBFs" on older releases of the server) "SWR" (Software RollUp). These software updates are available to supported users via download.

A common approach to dealing with SWRs is:

1. Wait a couple of weeks or months before installing the upgrade. On very rare occasions, a fix will come out immediately after the SWR release.
2. Install it on the development server and test it there for a few weeks. This is often common sense in most shops. Before anything is implemented in production, the changes should be implemented in a development environment and rigorously tested to make sure the new upgrade fixes the identified problems and also to make sure the changes do not expose any new problems.
3. Finally, after testing on your development server, install it on your production server. Be prepared to back off your upgrade at a moment's notice. It is always advisable to hope for the best and prepare for the worst when dealing with software upgrades. Should the installation of a software update go badly, the worst case scenario is the database server may need to be rebuilt. Be prepared to fall back to the last known working version of the server should the upgrade process fail.

The above approach is considered appropriately conservative. Many shops generally adhere to balancing the following two axioms regarding software implementation.

1. If it ain't broke, don't fix it. The various SWR releases usually come with a list of problems that have been repaired. Often, the majority of the list may not apply to the applications running on your servers. Introducing the latest SWR may cause the applications to behave differently enough that major modifications may be called for.
2. If you put off upgrades long enough, eventually Sybase will not support you. While patience is considered a virtue, waiting too long may invalidate your support agreements. The goal is to not wait too long.

Recording Run-Time Data

sp_configure

This stored procedure will list all system configuration variables that are easily set by the administrators. These configuration settings are also stored in a file under the \$SYBASE directory called <servername>.cfg; if multiple servers are running there will be multiple configuration files. Adaptive Server also archives the file when changes are made; these backup files are named <servername>.bak (current version) and a series of numbered older files <servername>### (historical configurations). Always keep offsite copies (hard and soft) of the config file. The copy can be used to start a restored server without having to manually reconfigure the server a parameter at a time. It is also a good idea to archive a good copy of the results from a run of sp_configure. If there are any problems with the configuration file, this allows the appropriate settings to be entered manually.

Disaster Recovery

It is also important to prepare a disaster recovery plan, possibly more than one. When dealing with any disaster recovery plan, always ask how each plan could go wrong. Include, as part of the plan, the physical database dumps as well as any configuration data or scripts that would be used to re-construct the server (or database). Test these backup plans extensively. Ask the practical joker in your group to review your plan and pick holes in it.

Database-Level Maintenance

An ASE database is the server's unit of backup & recovery. It is portable and self contained, in that it has its own data dictionary,

Adaptive Server has a series of routines called *dbcc* or *Database Consistency Checker*. These were originally unpublished routines used to verify the supporting structures for the data stored in the database. These structures, if corrupted, can prevent the database from functioning, and if left unrepaired can potentially lead to a fatal corruption of the database. If the corruption is copied into a database backup, the backup and future restorations from that backup will also be corrupt.

DBCC CHECKSTORAGE

We use dbcc checkstorage, the evolution of the older dbcc commands, to validate database consistency on a periodic basis (preferably nightly). This gives you the opportunity to stop a tiny error from snowballing into a larger one.

Soft and Hard dbcc Faults

dbcc categorizes faults discovered under two categories: Soft and Hard faults.

Soft Faults

A soft fault is an inconsistency in Adaptive Server that has not been determined to be persistent. Most soft faults result from temporary inconsistencies in the target database caused by users updating the target database during the dbcc checkstorage operation or by dbcc checkstorage encountering Data Definition Language (DDL) commands. These faults are not repeated when you run the command a second time. Soft faults can be reclassified by comparing the results of the two executions of dbcc checkstorage or by running dbcc tablealloc and dbcc checktable after dbcc checkstorage finds soft faults. If the same soft faults occur in successive executions of dbcc checkstorage, they can be considered "persistent" soft faults. Persistent soft faults may indicate a corruption, or a "hard" fault. If dbcc checkstorage is executed in single-user mode, the soft faults reported are "persistent" soft faults. These faults can be resolved by using sp_dbcc_differentialreport or by running dbcc tablealloc and dbcc checktable. If you use the latter two commands, you need to check only the tables or indexes that exhibited the soft faults.

Generally soft faults that remain after checkstorage is run are not significant and no further action need be taken.

Hard Faults

A hard fault is a persistent corruption of Adaptive Server. Not all hard faults are equally severe. For example, all the following situations cause a hard fault, but the results are different:

- A page that is allocated to a nonexistent table minimally reduces the available disk storage.
- A table with some rows that are unreachable by a scan might return the wrong results.
- A table that is linked to another table causes the query to stop.

Some hard faults can be corrected by simple actions such as truncating the affected table, or running a different dbcc command with the fix option. Others can be corrected only by restoring the database from a backup.

Database Dumps

Regularly dump each database, including the master, sybsystemprocs, and model databases. Since all the *tempdb* databases are cleared whenever the server is started it is unnecessary to take dumps of this database. Aside from making the backups, it is also important to get a copy of the backups in a secure off-site location. Additionally, Keep track and maintain a record of the dumps.

Log Management

Transaction logs are the repository of changes made to the databases. But since they are limited in size, they must be watched to prevent them from filling up. A full log stops all modification to the database, and the only way to reliably "prune" the logs (remove completed, checkpointed transactions) is to dump them. Make sure that the logs are dumped on a regular basis and use threshold management to avoid "log segment full" errors.

To verify how large the logs have grown, use the *sp_spaceused* stored procedure or *sp_helpsegment logsegment* to monitor the log and its remaining pages.

Space Management

Another task that is left to System Administrators is to ensure that the databases themselves are not completely full, or that the space in a database is not being misused. A database will grow over time as data is added to it. Eventually, it can grow to the point where it uses all its allocated space. To prevent this from happening, use the *sp_spaceused* stored procedure to verify the unused space in the database. When the database approaches full, more space can be added or data can be purged from the database.

It is also the administrator's job to ensure that the space is not misused. Try to make sure that the databases are adequately sized with room to grow.

Note that you can now configure databases for automatic expansion. Some shops do this so that their production environments don't have sudden problems; others do not so that they know when they've miscalculated, and can address the issues that made the databases grow unexpectedly.

This is another good task to perform weekly to start, and then move to monthly or quarterly based upon growth patterns you see.

Script Maintenance

As we mentioned before, from time to time, it may become necessary to attempt a complete rebuild of an Adaptive Server. In order to recreate the server it is essential to have scripts ready that can perform the installation, configuration, and population of the databases. Always keep

up-to-date scripts with all database definition information, from *disk inits* through object creation. Many of these can be created retroactively with third-party reverse engineering tools, or the *defncopy* utility. Keep the scripts on a different machine than the production database server. This way, if the production database server is ever corrupted beyond repair, the server can be duplicated on another machine.

Verifying Dumps

"Almost any administrator can perform a backup. A useful administrator can perform a restoration."

The above comment is not directed towards the administrators themselves but at the backups they create. When performing backups to tapes, the tapes can become worn or damaged preventing restoration from the bad tapes. It is also possible that the backups have not been performed properly, which may also prevent restoration.

If tape striping is used, beware of any bad tapes in the stripe set; a single bad tape will invalidate an entire stripe set. The same is true of long chains of transaction log dumps — a single bad transaction log dump will invalidate all subsequent dump files.

Try to get a separate server set aside for periodic testing of the database and transaction log dumps so you can verify whether or not the backups can actually work (this is a great task for junior DBAs to do approximately quarterly). This also provides an excellent rehearsal for a real disaster.

Table-level Maintenance

Update Statistics

When a query is processed by Adaptive Server, the optimizer makes a decision as to the most effective method for retrieving the data. The data used to make these decisions is known as the optimizer statistics. ASE uses two system tables called *systabstats* and *sysstatistics* to store a variety of measures on each table in the database. Statistics dealing with the table as a whole (*systabstats*) are dynamically maintained, but those on individual columns (*sysstatistics*) are not, and must be updated on a regular basis. Otherwise, the data distribution of the table may change and the Adaptive Server may not know it. Updating statistics may be a very intrusive task and should be performed during off-peak times. For 24x7 applications, choose a non-peak time, but do run this periodically.

Note: In truly high-volume environments, you can run this on a restored database, and use then import those statistics into the production server using the *optdiag* utility, but that's a lot more work.

Statistics (including time since the last update) may be observed by querying the *systabstats* and *sysstatistics* table, or by using the *optdiag* utility from the OS prompt.

Here are a few tips:

- 1) If your tables are large (and whose aren't these days?), you can update the statistics based upon a sampling, rather than an entire table; this speeds things up dramatically. Testing has identified a 10% sampling to be as consistent as 100% sampling, so I usually select 20-25% for safety's sake.
- 2) I usually update *index* statistics rather than all statistics; it's also much faster, only updating statistics for columns that are in indexes.
- 3) Update your statistics as often as you can afford to; barring that, update it as the data on which you index skews.
- 4) For maximum speed, you can update your stats in parallel (if parallelism is enabled).

Indexes

Over time, indexes can become unbalanced, and can become fragmented. It's a good idea to periodically drop & recreate clustered indexes, which will automatically cause nonclustered indexes to be dropped & recreated. Note that for non-APL tables, if you are running reorg rebuild (or similar), you are going to rebalance the indexes.

This is a good task to perform nightly or weekly to start, depending upon your downtime windows and database sizes; these may need to be staggered.

Summary

Especially in production systems, system health is vital to keeping the servers operational and at peak performance. To verify the status of the system, it is important to run basic maintenance routines to ensure that no problems have begun to develop. These tasks, depending on the size of the environment can vary from just a few steps to complex, involved routines. They can take potentially a few minutes each day to taking over the time of an administrator. Because of this and the criticality of the tasks, it is also recommended that they be automated to whatever extent is possible. Develop scripts to perform some of these tasks and page or notify administrators when a problem appears, or invest in some third party tools that automate and perform some of these tasks.

It is up to you to create a checklist, validate that preventive maintenance is running, validate the output, and act accordingly.