

Sybase Adaptive Server® Enterprise 15 Underused features for the DBA

BY JEFFREY GARBUS, SOARING EAGLE CONSULTING

INTRODUCTION

Sybase Adaptive Server Enterprise (ASE) 15 is a fully-featured database management system (DBMS) for the DBA. In addition to new performance features for ASE 15 (semantic partitioning, for example), and later enhancements to 12.5 (the job scheduler, MDA tables), there are a variety of tools available to the DBA which can be used to manage the performance and administration of your production system.

Your author has had the pleasure of having worked at many dozen, perhaps hundreds of ASE shops over the years, and has noticed that there are a variety of great features which are consistently underutilized.

There's no reason, for the most part, not to use these (though I will suggest guidelines as we discuss each feature), though a few you will want to use in moderation.

We'll offer up a few guidelines as we go.

RESOURCE GOVERNOR

It's pretty common to hear complaints along these lines (these are all from real "issues"):

- "The system was slowed down last night because a super-user was downloading 30 million rows of data to his Access database and it caused a bit of blocking & network bottlenecking"
- "We have a mixed-use environment with no restrictions on when various departments can run jobs. Some of these jobs can grab significant resources over the course of a few minutes and completely stop up the rest of the processing"
- "The marketing guys will frequently send coupons to a couple of thousand folks, but every once in a while will accidentally try to send a few million, which gums up the system in many ways"

The Resource Governor provides the ASE administrator with a mechanism to limit the resources a given process may request or acquire. It can limit the runaway queries at the application or login level. It can limit:

- I/O cost, estimated and/or actual
- Elapsed time, estimated and/or actual
- Number of rows to be returned, estimated and/or actual

Stated differently, you can choose to enforce your resource limits:

- Prior to execution
- During execution
- Or both

You can also choose a scope for the action. If you have a complicated batch, you may want the scope to be the batch rather than an individual SQL Statement. Your choices there are:

- Query
- Batch
- Transaction
- Batch and transaction

The Systems Administrator can instruct the server to perform any of the following actions when a resource limitation is exceeded:

- A warning can be issued
- The batch can be aborted
- The transaction can be aborted
- The login can be terminated (session ended)

You can also set different resource limits for different times of day. For example, you may choose to let the “batch” login have all the resources it wants from midnight to 6am, but limit the resources the rest of the time (of course, you want to make sure your developers know that they need to put restart logic in their scripts!)

Resource limits are enabled at the server level. This allows the server to allocate memory for time ranges, internal alarms, and to set up storage structures to apply applicable ranges & limits to login sessions.

Note that this is where we get into a point of contention with some DBAs, who claim that enabling resource limits slows the server down. In a sense they are right; it does take some server resources to maintain the resource governor; but it’s not that much. If you’re running at 90% of capacity, you don’t want to add ANY overhead to your box; if you’re running at 30% of capacity, it doesn’t matter a lot. In between, there’s a gray line some place... the Resource Governor should take up less than 5% of resources, though these numbers can be difficult to impossible to measure.

LOGICAL PROCESS MANAGER

Particularly for folks who work with multiple RDBMS, the mechanism with which ASE deals with multiple processors can be a mystery. And, in today’s multiple-core environments, it’s unusual to have only one processor.

So, the simple answers are:

1. ASE creates an OS-level process for each engine you define (sp_configure “engine”, N).
2. These processes have no affinity whatsoever to the physical processors (or virtual processors, as the case may be)
3. There is a “dbcc tune” command to create an affinity, but it’s rare to need it (and you shouldn’t do this unless Sybase Technical Support gives you a good reason to!)

That said, within the scope of tuning your system, you may have a need to give, say, 25 of your 32 processors to your OLTP users, and only 7 of your processors to batch or DSS or... whatever you decide.

You can accomplish this using the logical process manager (LPM).

In addition, you can change the run-queue priorities of tasks or logins, so that whichever set of processors they’re assigned to, they get either first crack or a lower-priority crack at resources.

There are three predefined execution classes within ASE:

EC1	High Priority
EC2	Medium Priority
EC3	Low Priority

From a practical standpoint, this may be all you need, unless you want named priorities to which to bind processes.

Each of the execution classes are assigned three attributes:

- Base priority, the initial priority of any task in this class. Priorities can be increased by Adaptive Server as a task is running
- Engine affinity, the numbers of the engines upon which a task can run. By default a task can run on all engines
- Time slice, the amount of CPU time a task can run before it gets swapped out of the CPU

You can assign work to a processor group at three levels:

- Application
- Login
- Stored Procedure

So in other words, you can specify that certain logins may run at a lower priority, or that a particular stored procedure runs at a higher priority regardless of who runs it.

PARALLEL QUERY PROCESSING

Parallel Query Processing (PQP) became available with ASE version 11.5.

The idea behind parallel processing is to allow multiple processors resolve a query for a single process. In other words, one query may use multiple processors.

This is not the default ASE behavior, as the following learn-from-my-mistake may highlight.

When PQP first came out, we became excited (we're P&T geeks). We had a system running at about 30% of capacity for about 800 simultaneous users (not a huge system, but not a tiny one either). Looking forward to all of the "Hey! Our system got much faster" calls, we immediately enabled parallel query processing, and set up for 1000 additional parallel processes. We figured, "Well, 800 processes are running at 30%, worst case 1800 run at about 70%." We were wrong. The box (an 8-way, which we had configured for a degree-of-parallelism of 7) was instantly slammed at 100% of CPU.

Needless to say this was not a hit, and calls from the users were not telling us how much faster the server seemed to be running. We suspect this is why many folks are not using PQP, out of a bad experience they had at some time in the past.

PQP is a resource hog, in that it will consume all of the resources you give it; so, don't give it all of the resources.

Assuming you have some idle CPU & memory resources, start it slow, and then titrate up the number of worker processes as well as the degree of parallelism. Keep the hash parallel degree below the number of processors. Watch your CPU, as well as sp_sysmon output in the parallelism category. Bump it up slowly & wait for the happy phone calls.

There are a lot of different types of query that will benefit from PQP. Plus, if you are implementing semantic partitioning, you get significant performance advantages as the processors divide up the work amongst partitions.

SERVER-LEVEL ROLES

In the early days of SQL Server, maintaining users in multiple databases was a nontrivial exercise.

Once a login was added to the server, you needed to determine in which database(s) to place the login. If an application spanned multiple databases, you had to add that login to each of those databases, and within those databases add the login to whatever group is appropriate. Of course, within each database, a login (user) could only be added to one group (in addition to the "public" group).

Between the need to map logins across databases, and the 1:1 limitation of users to groups, security in a complex application could be... complex.

With the advent of around ASE 11, Sybase introduced the concept of server-level roles, for the sa, sso, and the operator. These were needed so that all administrators weren't logging in with the same id/password, not to mention those folks dealing with server-level security and backup/restore operations. This led to the question, "If they can do it, why can't we?"

Some of us more adventurous types modified the system tables to create & use our own groups. Then Sybase made it easy for everybody by adding this functionality around 11.9.2.

Now, you can create roles at the server level. These (server-level) roles can be added to individual databases, and permissions granted at the database level. And here are the cool things:

1. If you grant the login the server-level roles, the login will have access to any and all databases to which you have added the roles
2. The logins will inherit, in each database, the permissions granted to those roles
3. There is a 1:many (not a 1:1) of logins to roles, so you may grant logins multiple roles within a database, even though you cannot place a user into multiple groups

UPDATING STATISTICS (STEPS & SAMPLING)

One consistent performance ... opportunity ... which comes to our attention is along these lines: "This query runs great about 98% of the time, but 2% of the time the performance is simply off the wall. We've tried using it in a stored procedure with recompile, but even that isn't working."

Could be, the problem is in your histogram. For the most part, the default histograms are great for the optimizer to use for determining optimal query plans and join orders for resolving your database request. Sometimes, though, the default histogram isn't enough.

At a lot of shops where we do performance and tuning engagements, we have ended up changing their default "update index statistics" parameters (you do run this on a regular basis, do you not?).

When we hear of erratic query performance, one of our early preventive changes was to increase the number of distribution steps to 200, which coincidentally is the new default in ASE 15 (if it's not, set it, you can do so at the server level). If your performance is still occasionally erratic, consider setting the steps to something higher. This especially applies to large tables with unbalanced or temporally erratic distributions.

Also, especially for those of you with large databases / tables, the sampling technology is useful. A friend whose opinion we trust (and who coauthored our performance and tuning book) ran significant performance tests, and informed us that a sampling rate as low as 10% gave him the same results (correct optimizer selections) as a 100% sampling rate. In our DBA-inspired paranoia, we've always selected 25% ourselves.

Between the larger steps for performance, and the smaller sampling rate for speed of maintenance, you should be able to help the optimizer out by a bit.

JOB SCHEDULER

Sybase released the job scheduler, an answer to the question, "How do I plan my maintenance schedules?"

In our Systems Administration books, we always recommend a preventive maintenance regimen (run dbcc, backup your databases, update your statistics, to name a few). What we don't do is talk about how to queue it up and verify that it ran.

Most shops have been running UNIX, and using the UNIX crontab for scheduling. I've seen some pretty intense scripts written in a variety of shells that do everything from count databases to running table level maintenance (update statistics) for the dynamic table list.

Today, the job schedule that ships with the server is a java-based scheduling tool that lets you schedule everything from your preventive maintenance tasks to your own batch jobs. You also get templates for most of the standard preventive maintenance tasks...

MDA TABLES

With the advent of the middle of the 12.5 release, Sybase introduced the monitoring & diagnostic tables (see our other article, Identifying Slow-Running Queries in ASE 15 for this & other tips on setting up & using the MDA tables).

Many folks don't use the MDA tables because of a perception that the performance hit is too high.

Pursuant our earlier statement, if you're already running at 95% of capacity, don't add anything to your system; unless you're trying to find out why you're running at 95% of capacity.

We think that part of the reason there's a perception about the performance hit of the MDA tables is that when you enable absolutely everything, it does hit the system over 5% of CPU (I've heard but never observed 10%). (Rob Vershoor's web site has a nice script to get this all set up)

The catch is, you don't need to enable everything. Take those components you're trying to track down, enable them, and leave the rest alone.

As an alternative, enable them when you're trying to collect performance data, and then disable them. At the very least, this can help you pick up baseline performance data, and at the very best, can help you isolate and identify system bottlenecks to correct.

SUMMARY

None of these tools are a mystery. They're simply not in use in most of the shops we see. They run the gamut from performance enhancers to mechanisms used to control logins or processes that have run out of control.

Push the limits of the server, it is there for your convenience.

Jeff Garbus has 20 years of expertise in architecture, tuning and administration of Sybase ASE, Oracle, and Microsoft SQL Server databases with an emphasis on assisting clients in migrating from existing systems to pilot and enterprise projects. He has co-authored 15 books and has published dozens of articles on the subject. Mr. Garbus is the CEO of Soaring Eagle Consulting, an organization that specializes in assisting businesses maximize database performance www.soaringeagle.biz.