Memory configuration, performance profiling, setting your fill factor, and more

Edited by Nancy Michell

Configuring memory

Q I'm trying to figure out the best memory configuration for my SQL Server boxes. The admin who preceded me set Boot.ini for each machine with 12GB of RAM on a SQL Server 2000 two-node cluster as follows: Yes /PAE NO /3GB (No AWE set for SQL Server). With 12GB of RAM available, should I remove the /3GB switch in Boot.ini, turn on AWE and give SQL Server about 10GB of the 12GB? Nothing else is on the machines running SQL Server, so no other apps need memory.

A Yes, you should turn on address windowing extensions (AWE) and preallocate an upper limit of RAM for SQL Server – 10GB sounds good on a 12GB dedicated SQL Server box. (Note that preallocating is only valid for SQL Server 2000. Starting with SQL Server 2005, using AWE is no longer static and can be changed on the fly.) There has always been a big debate as to whether to use both the /3GB and /PAE switches or just the required /PAE. Though you only truly need /PAE set and AWE enabled, I suggest you use both switches; however, there are some factors to consider.

Using the /3GB switch boils down to whether it's needed in your particular situation. Are you running out of MemToLeave areas of memory that must be in the first 2 or 3GB of virtual address space? If you do enable the switch, are you starving the operating system of memory? (See support.microsoft.com/kb/316739 for more information.) If you're working on a cluster, you could set /3GB on one node and not on the other. That way, if you're testing with /3GB and having problems, you can fail the instance over to the other node pretty quickly. Keep in mind that if you have more than 16GB RAM, the /3GB switch is not supported.

By using /3GB, you are increasing the virtual address space (VAS) by 50 percent, so applications that put memory pressure on VAS and not just the data cache benefit greatly. Fortunately, the 64-bit servers, both IA64 and x64, eliminate this misunderstood factor. Concerns regarding starving the OS are not applicable if the machine is dedicated to SQL Server. Leaving 2GB for the OS is a bit of overkill as well; if this server is dedicated to SQL Server with only the standard minimal OS services running, you'll find there will be about 1.3GB of free memory on the server, so you might as well let SQL Server use that extra 1GB. Start at 10GB, use PerfMon to monitor available memory over a long period to see how much sits idle, and then adjust accordingly. Remember that you will incur swapping if you over-commit on SQL Server 2000, as AWE is not as dynamic as it is with SQL Server 2005. The key to deciding whether to use the /3GB switch is testing it in your specific environment.

THANKS TO THE FOLLOWING MICROSOFT EXPERTS FOR ANSWERING THIS MONTH'S QUESTIONS: *Jacky Chiou, Michael Epprecht, Saleem Hakani, Rajnish Kaushik, André Prinsloo, Buck Woody, Cheng Xie and Jacob Yang.*

Instance names for replication

Q Can I use my server's IP address now in SQL Server 2005 Replication to indicate which instance to replicate? In SQL Server 2000, according to 'How to Replicate between Computers Running SQL Server in Non-Trusted Domains or across the Internet' (support.microsoft.com/kb/321822), doing so will cause errors, but I don't know if this is still true.

A When specifying server instances to participate in replication, you must supply the SQL Server registered instance name. For example, you must use the SQL Server instance name when specifying Publisher or Subscriber parameters to replication stored procedures or to the replication agent connection settings on the command line. If the network name for the SQL Server instance differs from the registered instance name, replication connections by agents will fail.

If the network name of the instance and the SQL Server instance name differ, consider adding the SQL Server instance name as a valid network name. One way to set an alternative network name is to add it to the local hosts file. The local hosts file is located by default at WINDOWS\system32\drivers\etc or WINNT\system32\drivers\etc. For example, if the computer name is comp1, the computer has an IP address of 10.193.17.129, and the instance name is inst1/instname, add the following entry to the hosts file:

```
10.193.17.129 inst1
```

SQL Server Integration Services

Q I am installing an active/active cluster for SQL Server 2005 (64-bit Enterprise Edition with two servers) and I'll have a total of four instances of SQL Server 2005. SQL Server Integration Services (SSIS) will be required for all the instances. What can you tell me about clustering SSIS and the effect on maintenance plans?

A While you can cluster the SSIS service, it is not necessary to do so and you could run into a variety of problems, including a lack of support for delegation (see msdn2.microsoft.com/aa337083) and that it's not multi-instanced – you can only have one instance running on a node at a time.

Tip: Easier profiling

Did you know you can now correlate PerfMon with SQL Server 2005 Profiler?

Perhaps you've noticed CPU usage spikes, excessive memory consumption, or overall slow performance in Performance Monitor on your SQL Server machine and wondered what caused these performance anomalies. Before SQL Server 2005, you would have to use Profiler to capture a trace, then look at sysprocesses in Enterprise Manager, and finally, capture your Performance Monitor logs – which, of course, required you to fire up PerfMon. And after all this work using all these tools, you'd still need to manually reconcile events between the tools to figure out why performance was suffering. And that would mean slogging one-by-one through each of the logs. It wasn't fun, but you had to do it if you wanted to get to the bottom of your performance problems.

With SQL Server 2005, you still need to capture a trace and examine your PerfMon logs, but Profiler now lets you attach them. You can scroll through your T-SQL statements and Profiler will automatically show you, graphically, what happened. If you click in the Performance Monitor user interface in Profiler, you will jump to the statement that correlates to that timestamp. This saves lots of time in troubleshooting your SQL Server environment.

Let's see how to attach PerfMon logs to Profiler:

1. Start Performance Monitor and begin capturing information from the database server.
2. Create a new counter log under Performance Logs and Alerts and enter a name for your new log.
3. Add new counters, such as % Processor Time. You'll also want to set your logging to start either manually or on a scheduled basis using the Schedule tab.
4. Click OK and, if you elected the manual option, be sure that you start logging.
5. Set up a trace on your SQL Server through Profiler. You can do this by clicking New Trace on the File menu. Make sure to include the StartTime and EndTime in your trace, then name the trace and set it to save to a file. Finally, you must blast your server to simulate some transaction activity, then stop capturing both Performance Monitor and Profiler data.
6. In Profiler, select Import Performance Data from the File menu. Next, choose the location where you stored your PerfMon log and select File | Open | Trace. Finally, select the location where you stored your Profiler trace.

When you're done, you'll see how much easier it is to figure out what effect particular SQL statements have had on processing time.

Tip: Check your fill factor

Say you have a glass completely filled with water, and you try to put more water in that glass. What happens? The water overflows.

It's like that with SQL Server. Whenever a new row is added to a full index page, SQL Server moves about half the rows to a new page to make room for the new one. This is known as a page split. Page splits make room for new records, but they also take time and are very resource intensive. And they can cause fragmentation, which may adversely affect I/O operations. So how can you avoid them?

To prevent such situations, you must proactively determine the fill-factor value. When an index is created or rebuilt, the fill-factor value determines the percentage of space on each leaf-level page to be filled with data, reserving the remainder for future growth. For example, configuring a fill-factor value of 60 means that 40 percent of each leaf-level page will be left empty to provide for index expansion as data is added to the underlying table.

The default fill-factor value is always 0, which is OK for the majority of situations. Basically, a fill factor of 0 means that the leaf level is filled almost to capacity, but some space is left for at least one additional index row. (Note that a fill factor of 0 and 100 are similar.)

You can set the fill-factor value for individual indexes during a CREATE INDEX or ALTER INDEX statement, or you can configure this value directly at the server level so that any new indexes created will use the default.

The following example sets the fill-factor value at the server level to 70 percent, meaning you will have 30 percent free space for future expansion. Of course, you must carefully test this option before implementing it in a production environment.

```
USE Master;
GO
SP_Configure 'show advanced options',1;
GO
SP_Configure 'Fill Factor', 70;
GO
--You must restart SQL Server Engine for changes to take effect.
```

What if you want to configure the fill factor at the individual index level? Assume you are building the following table and you would like to create a unique index on the column called Col_A with a fill-factor value of 70. The command would look like this:

```
--Create an Item table
USE Item_DB;
GO
CREATE TABLE ITEM (Col_A Varchar(100),Col_b Varchar(200));
GO;

--Create a unique index on colum Col_A of Item table with a Fill Factor value of 70
CREATE UNIQUE INDEX AK_Index ON Item (Col_A)
WITH (FillFactor = 70);
GO
```

How do you identify the fill factor for each index? You can query sys.Indexes to get the fill-factor value for all the indexes in a database, like so:

```
USE Item_DB;
GO
SELECT Fill_Factor FROM Sys.Indexes WHERE Object_id=object_id('item') AND name IS NOT NULL;
GO
```

Previously, SSIS had to be installed – not running, just installed—for the Maintenance Plan Wizard to run. However, this is no longer the case in SQL Server 2005 SP1. If SSIS is not running, the maintenance plans can be executed by SQL Server Agent.

Rather than clustering SSIS, you may consider keeping it running as a standalone service and editing the MsDtsSrvr.ini.xml to point to any and all running instances. This lets you manage your packages from any nodes and provide the high availability most customers are looking for without any of the problems associated with clustering the service.

For more information on failed maintenance plan creation, be sure to see the Knowledge Base article at support.microsoft.com/kb/909036.

Strange execution times

Q During my load test on my SQL Server 2005 SP1 box, SQL Server Profiler recorded many negative values of stored procedure (SP) execution time and in some cases the SP execution time, did not match the result of subtracting start time from end time.

A There are a number of things that can affect the reporting of SP execution time and other performance times in SQL Server Profiler. Remember, SQL Server 2005 uses milliseconds to count execution time, and if you are using any technologies that change the unit of measurement, you will get inconsistent reporting and execution times that don't add up.

For example, if you're using other power schemes, CPU stepping or AMD Cool 'n Quiet technology, you are changing CPU frequencies, which then do not match what SQL Server Profiler is expecting when calculating execution time.

There's a Knowledge Base article at support.microsoft.com/kb/931279 that explains the symptoms, a variety of causes and some remedies. ■