Tech Tip: Understanding Server Memory Counters

Written by Bill Bach, President of Goldstar Software Inc.

This tech tip is the second in a series of tips designed to help you understand the way that your Pervasive PSQL Summit v10 database engine utilizes memory on the database server, and thus provide you with insights on optimizing that memory usage to get the best performance out of your system. In the last Tech Tip, we investigated the database cache counters to see how well we are doing now. In this tip, we provide some background information on your server's memory counters.

Defining Memory Counters

There are a few primary memory counters that you need to understand in order to tune any server, so let's start with definitions so that we are all on the same page:

Installed Physical RAM: The amount of physical RAM is the amount of memory currently installed in your computer. This is the amount of memory which your computer BIOS reports at boot time and reflects the amount of memory chips installed.

Available Physical RAM: The amount of RAM which is actually made available by the operating system. Some operating systems will report this as the same as the installed physical RAM, but not always. For example, Windows 2003 Standard will make only 4GB available, regardless of the amount of physical memory. As a result, installing Windows 2003 Standard onto a machine with 8GB of memory actually ends up wasting half of its resources.

System Memory In Use: The amount of memory that is currently allocated by the operating system. Note that the amount of memory in use can be higher than the available physical RAM due to the use of the Windows swapfile, which can use disk space as an extension of memory (albeit VERY slowly).

Process Memory Usage: The amount of available physical RAM that is currently allocated by the operating system to a specific process.

Process Virtual Memory Size: The amount of memory that a process has requested for its own use. This number may be (and usually is) larger than the process memory usage, because the operating system may swap out portions of the process code and data to the swap file, and if these are not needed, they stay on disk instead of in RAM.

Process Maximum Addressing Space: This number indicates the maximum amount of memory that a process can possibly access. It includes all code, data, stack space, and any other related data structures.

Process Addressing Space In Use: This number indicates the number of bytes already reserved in the addressing space by a given process. This is the sum of all bytes needed to store all code and data structures in memory. Additionally, it includes all other OS overhead for the process, including a 1MB stack space for each thread allocated by the operating system.

Locating Windows Server Memory Counters

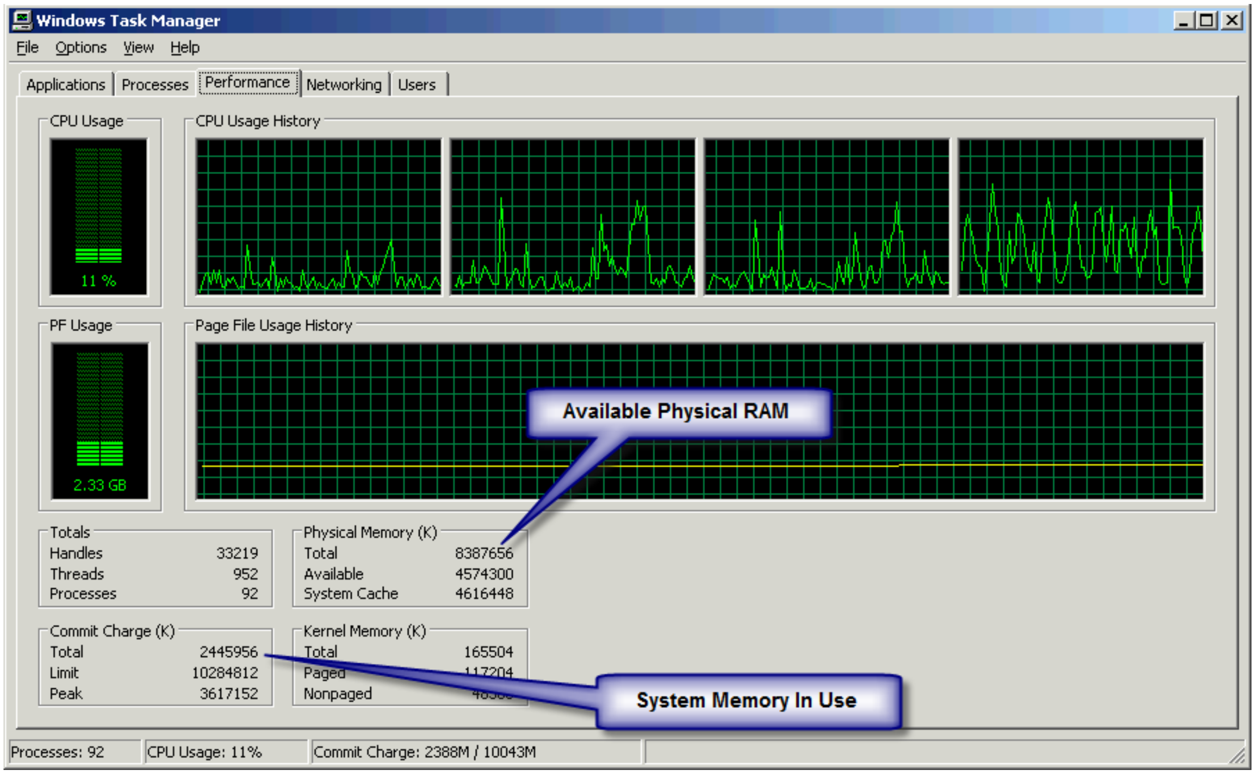
Now that we've got the definitions out of the way, let's do some research on our computer to locate these values.

Locating Installed Physical RAM

As previously indicated, the installed physical RAM can only be seen by rebooting the server and entering setup, or by watching the BIOS verify physical RAM at startup.

Locating Available Physical RAM and System Memory In Use

The next few items can be found by reviewing a Task Manager "Performance" tab screen. Let's take a look at a sample from a Windows 2003 Server:



Totals
Handles 33219
Threads 952
Processes 92

Physical Memory (K)
Total 8387656
Available 4574300
System Cache 4616448

Commit Charge (K)
Total 2445956
Limit 10284812
Peak 3617152

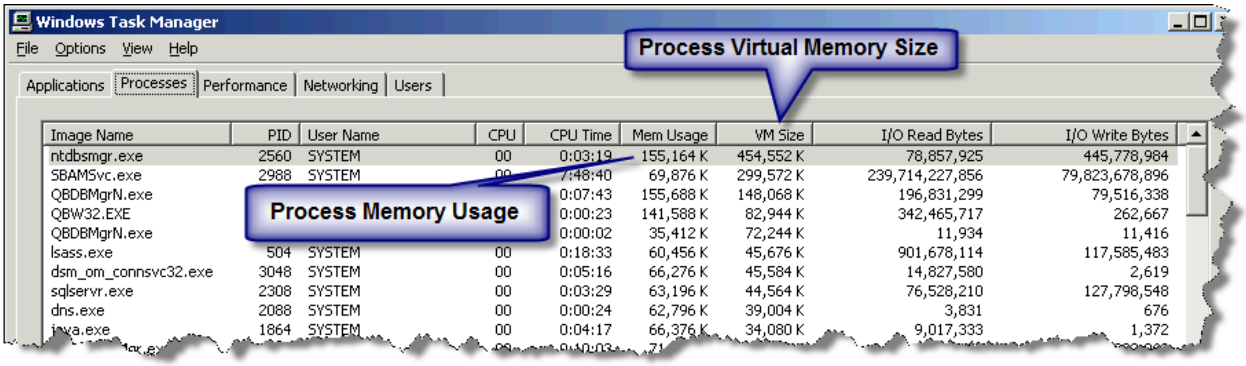
Kernel Memory (K)
Total 165504
Paged 117204
Nonpaged 48300

Processes: 92 | CPU Usage: 11% | Commit Charge: 2388M / 10043M

We see in this picture that the **Available Physical RAM** is 8GB, and the **System Memory In Use** is currently 2.4GB. It is also interesting to note the other two numbers in the lower left corner. The **Commit Charge Limit** is the sum total of the swap file size, which defaults to 1.5 times the amount of system memory, and this is the maximum amount of virtual memory supported by the system. The **Commit Charge Peak** indicates the highest level the **Commit Charge Total** has achieved since the system was last started.

Locating Process Memory Usage and Process Virtual Memory Size

Now, switch to the Processes tab in Task Manager. Select the View/Select Columns... menu option and select the columns **CPU**, **CPU Time**, **Mem Usage**, **VM Size, I/O Read Bytes**, and **I/O Write Bytes**. Then, click on the "VM Size" header to sort in descending order by this value, and your database engine should pop up to the top. Here's an example:



Process Virtual Memory Size

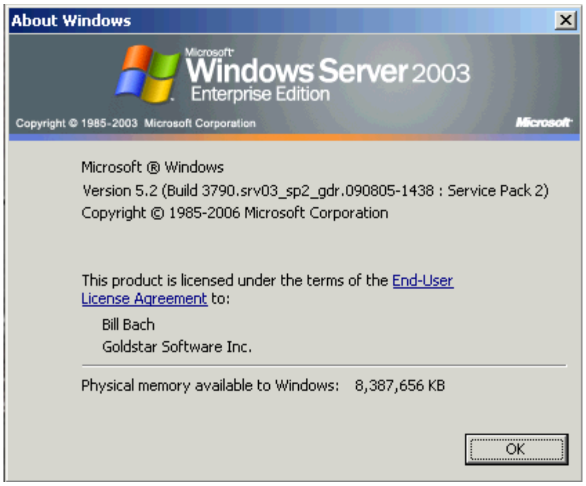
Process Memory Usage

Image Name	PID	User Name	CPU	CPU Time	Mem Usage	VM Size	I/O Read Bytes	I/O Write Bytes
ntdbsmgr.exe	2560	SYSTEM	00	0:03:19	155,164 K	454,552 K	78,857,925	445,778,984
SBAMSvc.exe	2988	SYSTEM	00	7:48:40	69,876 K	299,572 K	239,714,227,856	79,823,678,896
QBDBMgrN.exe				0:07:43	155,688 K	148,068 K	196,831,299	79,516,338
QBW32.EXE				0:00:23	141,588 K	82,944 K	342,465,717	262,667
QBDBMgrN.exe				0:00:02	35,412 K	72,244 K	11,934	11,416
lsass.exe	504	SYSTEM	00	0:18:33	60,456 K	45,676 K	901,678,114	117,585,483
dsm_om_connsvc32.exe	3048	SYSTEM	00	0:05:16	66,276 K	45,584 K	14,827,580	2,619
sqlservr.exe	2308	SYSTEM	00	0:03:29	63,196 K	44,564 K	76,528,210	127,798,548
dns.exe	2088	SYSTEM	00	0:00:24	62,796 K	39,004 K	3,831	676
java.exe	1864	SYSTEM	00	0:04:17	66,376 K	34,080 K	9,017,333	1,372

We see in this example that the NTDBSMGR.EXE process, the Pervasive PSQL Summit v10 32-bit database engine, has reserved approximately 450MB of virtual memory space, but it is only using 155MB of "real" memory at this time.

Locating Process Maximum Addressing Space

Unfortunately, the **Process Maximum Addressing Space** value cannot be found in a direct counter, so we have to do a bit of research. First, find out what operating system version (Windows 2003, 2008, Vista, etc.) and edition (Standard, Enterprise, etc.) you are running. Then, find out if it is a 32-bit or 64-bit version of that operating system. The "WINVER" command can help us with this information:



Microsoft ® Windows
Version 5.2 (Build 3790.srv03_sp2_gdr.090805-1438 : Service Pack 2)
Copyright © 1985-2006 Microsoft Corporation

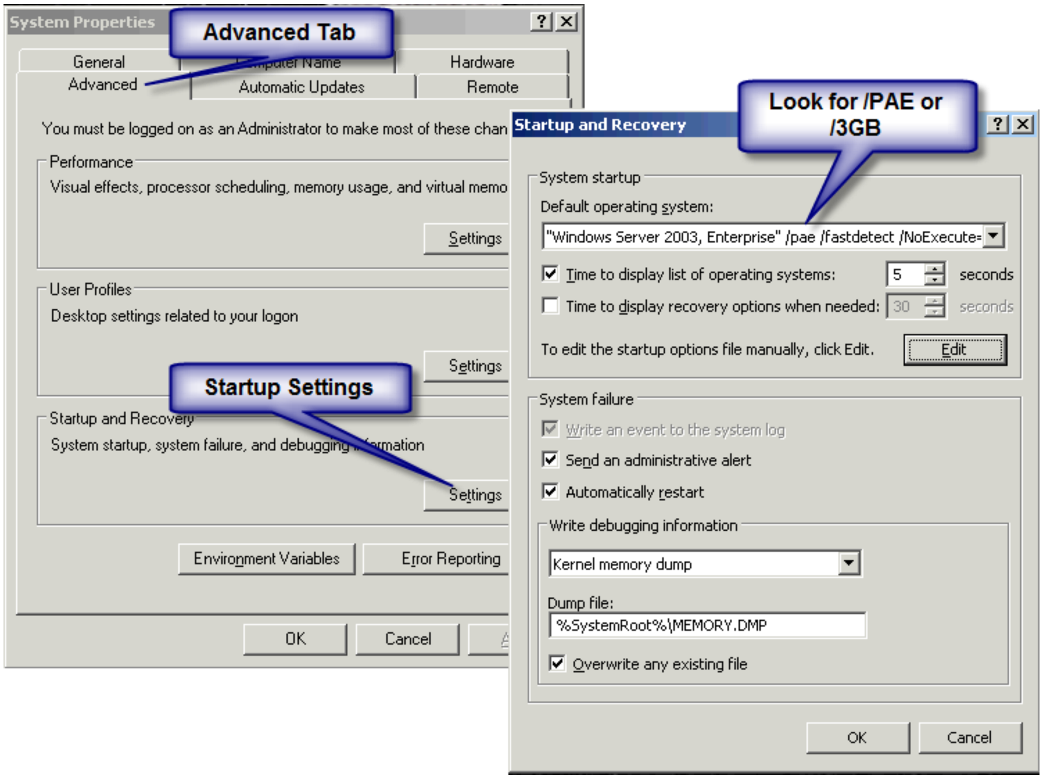
This product is licensed under the terms of the End-User License Agreement to:

Bill Bach
Goldstar Software Inc.

Physical memory available to Windows: 8,387,656 KB

Here, we see that we have a Windows Server 2003 Enterprise operating system with Service Pack 2 installed. We also see the physical memory available to Windows, which echoes what we saw in Task Manager above.

If you have a 32-bit system, you need to next find out if you've enabled the /3GB or /PAE switches in the operating system startup settings. We do this from the System Properties dialog (right-click *My Computer* and select *Properties*). Here, we see the /PAE switch, but no /3GB switch.



System Properties

General | Computer Name | Hardware | Advanced | Automatic Updates | Remote

You must be logged on as an Administrator to make most of these chan

Performance
Visual effects, processor scheduling, memory usage, and virtual memo
Settings

User Profiles
Desktop settings related to your logon
Settings

Startup and Recovery
System startup, system failure, and debugging information
Settings

Environment Variables | Error Reporting

OK | Cancel

Startup and Recovery

System startup
Default operating system:
"Windows Server 2003, Enterprise" /pae /fastdetect /NoExecute=
☑ Time to display list of operating systems: 5 seconds
□ Time to display recovery options when needed: 30 seconds
To edit the startup options file manually, click Edit. Edit

System failure
☑ Write an event to the system log
☑ Send an administrative alert
☑ Automatically restart

Write debugging information
Kernel memory dump
Dump file:
%SystemRoot%\MEMORY.DMP
☑ Overwrite any existing file

OK | Cancel

Finally, determine your Pervasive engine bit level, either 32-bit or 64-bit. (PSQLv10 is the first engine available in an x64 release, so if you are running an older one, you can safely assume 32-bit.) In this case, we know that we are running the 32-bit version of Pervasive PSQL Summit v10 database engine, because this is the only one that CAN run on a 32-bit server.

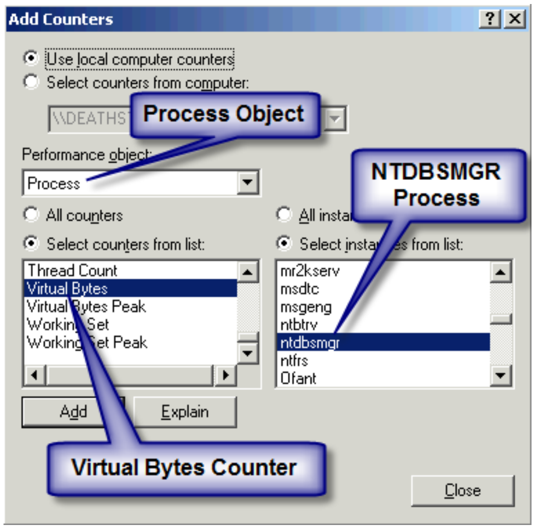
Once you have all of that information, go to this web site and find your configuration in the chart:

http://msdn.microsoft.com/en-us/library/aa366778%28VS.85%29.aspx

Finding the appropriate segment in the chart, we see that our 32-bit PSQLv10 engine on our 32-bit server (without the /3GB) switch supports a **Process Maximum Addressing Space** of 2GB. Further on down this web page, we can also see the maximum supported physical memory for our Windows 2003 Server Enterprise 32-bit operating system is 64GB, which means that we COULD theoretically add more memory, if the hardware (and budget) allowed.

Locating Process Addressing Space In Use

The last setting, **Process Addressing Space In Use**, cannot be easily divined from the tools that we've already seen, so we have to go in search of a new one. Let's use the Performance Monitor utility, otherwise known as PerfMon. After you've started PerfMon, click on the (+) icon on the tool bar to add a new counter with the following parameters:



Add Counters

● Use local computer counters

○ Select counters from computer:

\\DEATHS

Performance object:

Process

○ All counters

● Select counters from list:

Thread Count
Virtual Bytes
Virtual Bytes Peak
Working Set
Working Set Peak

○ All instances

● Select instances from list:

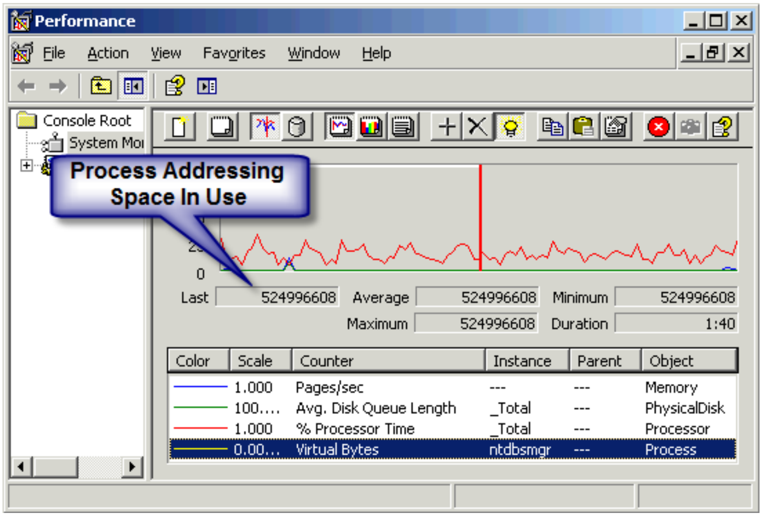
mr2kserv
msdtc
msgeng
ntbtrv
ntdbsmgr
ntfrs
Ofant

Process Object

NTDBSMGR Process

Virtual Bytes Counter

Click OK to add the counter to the PerfMon screen, and then click on that counter to highlight it.



Process Addressing Space In Use

Last 524996608 Average 524996608 Minimum 524996608

Maximum 524996608 Duration 1:40

Color	Scale	Counter	Instance	Parent	Object
─	1.000	Pages/sec	---	---	Memory
─	100....	Avg. Disk Queue Length	_Total	---	PhysicalDisk
─	1.000	% Processor Time	_Total	---	Processor
─	0.00...	Virtual Bytes	ntdbsmgr	---	Process

You'll notice that the highlighted line on the graph is WAY off the scale, and there's no way to adjust the scale to make it visible, so we'll just ignore the chart portion of the window. (This issue has been addressed in Windows Server 2008.) Luckily, we just need the number anyway, and we see here that we have approximately 524MB of Virtual Bytes in use by the NTDBSMGR process. In a nutshell, we are well under the stated maximum of 2GB and perfectly safe for the time being.

Summary

Hunting down all of these memory counters and maximums is really only the first step in understanding your server, but it is a critical step indeed. In our next Tech Tip, we'll delve deeper into the meanings of these values and see how to interpret these counters to further analyze and tune your Pervasive database server environment.

Author Information:

Bill Bach is the Founder and President of Goldstar Software Inc., a Pervasive reseller in the Chicago area that specializes in providing Pervasive products, services, and training to its customers in North America and abroad. Bill has written numerous tools and utilities to help system administrators and database developers work with their Pervasive database environments, and his training classes for Pervasive PSQL and DataExchange are the most comprehensive classes available. Get more information from http://www.goldstarsoftware.com.