lexana \ net

March 2007

Using WSRM to Track SQL Server's Resource Usage

By Drew Hamre

email

Microsoft's Windows System Resource Manager (WSRM) is a workload management tool included with Windows Server 2003 Enterprise or Datacenter[1]. Administrators can use WSRM to control how CPU and memory are shared among competing processes. WSRM is typically used to normalize consolidated workloads (reducing the risk that a misbehaving application will interfere with others on the system), or to ensure predictable response times for remote terminal users.

Less well known is WSRM's built-in accounting capability, which tracks the resources consumed by system and application processes. When running consolidated SQL Server instances, this information has advantages as a source for allocating system costs. In this paper we'll review WSRM and pay special attention its role in chargeback accounting for SQL Server.

An overview of WSRM

WSRM allows system administrators to set resource consumption policies for CPU and memory. Administrators select the processes to be managed, and then set resource consumption targets. WSRM will apply these policies based on a date/time schedule, and manage the system according to these active policies. It can generate alerts for policy events, and can track resource consumption (discussed below).

WSRM is written in C++ and implemented as a Windows service. This service is installed on each system where WSRM is required, and runs with LocalSystem authority. A separate administrative UI (written in C#) may be installed only on a single system, from which the other WSRM installations can be managed remotely.

Machine resources are allocated according to policies for CPU utilization (percent CPU), process working set size (physical resident pages), and committed memory (page table and page file usage). WSRM controls CPU allocations by adjusting process priorities. When a process exceeds its administrator-defined share, WSRM

---

[1] WSRM is included with Microsoft's Windows Server 2003 Enterprise/Datacenter media (though it requires a separate install), or it can be downloaded from Microsoft's site. The installation kit has been updated for SP1/R2 and includes versions for 32-bit, x64, and Itanium CPUs. For Windows, WSRM essentially replaces a third-party tool (a version of Aurema's ARMTech) that was bundled with Windows 2000 Datacenter.

lowers its priority. Unused resources are reallocated to other processes according to administrator-defined policies.

WSRM can adjust two aspects of memory utilization: a process's *working set* size and its *committed memory* size. WSRM can prevent a process's working set (those virtual pages that reside in physical memory) from exceeding a specified limit by calling kernel functions (i.e., *Get/SetProcessWorkingSetSizeEx*).

WSRM can also enforce limits for committed memory (the physical memory for which space has been reserved on the paging file). Increasing consumption of committed memory may be symptomatic of a memory leak. As committed memory cannot be taken back by the operating system, WSRM can either terminate the program or write a diagnostic error to the event log.

Limitations of WSRM resource management

While WSRM helps manage CPU and memory, it does not manage I/O, either storage or network. This may be an important limitation, especially if grappling with workloads that are I/O bound.

Another limitation is that WSRM manages *application processes* only. System processes should be excluded, and WSRM contains such a default exclusion list. (Note there's also an optional 'application exclusion' list that may be used, though it's empty to start.) Administrators shouldn't exclude applications that launch other processes (for example, mmc.exe), lest the launched programs (now unmanaged) consume available resources.

WSRM memory limits do not apply to *Address Windowing Extensions* memory, or large page or kernel memory. WSRM can't be used with other resource managers, or with applications that set their own priorities. Further, WSRM works independently of application parameters that control resource sharing, and should sometimes defer to these settings. In managing SQL Server, for example, WSRM's memory and CPU-affinity policies should not be set, deferring instead to SQL Server's native switches.

'Managing' versus 'profiling' policies

Note there are two policy types within WSRM: a *managing policy* that controls resource allocation for the server; and a *profiling policy* that simply logs information about a policy's effects, without controlling allocations. Profiling policies allow you to use WSRM only to capture accounting information, not to manage resources. We'll use *profiling policies* when configuring WSRM for accounting (below).

Using WSRM accounting to track resource usage

WSRM 'Resource Accounting' accumulates histories of process-level system usage. The accounting option is enabled during installation (Figure 1), or from within WSRM's administrative console. Once enabled, WSRM will periodically sample and record resource counters for all OS processes (both application and system).

This information is summarized every 10-minutes (though this interval can be modified) and persisted in a dedicated JET Blue ('ESE') database. WSRM's console includes a retrieval UI, allowing this information to be aggregated, sorted and filtered. Even more helpful to production systems, the data can be exported in tab-delimited, SMF, or CSV formats (below). Exported data can then feed into downstream processes to be used for chargeback accounting, to assess system performance, and to help project future infrastructure needs.

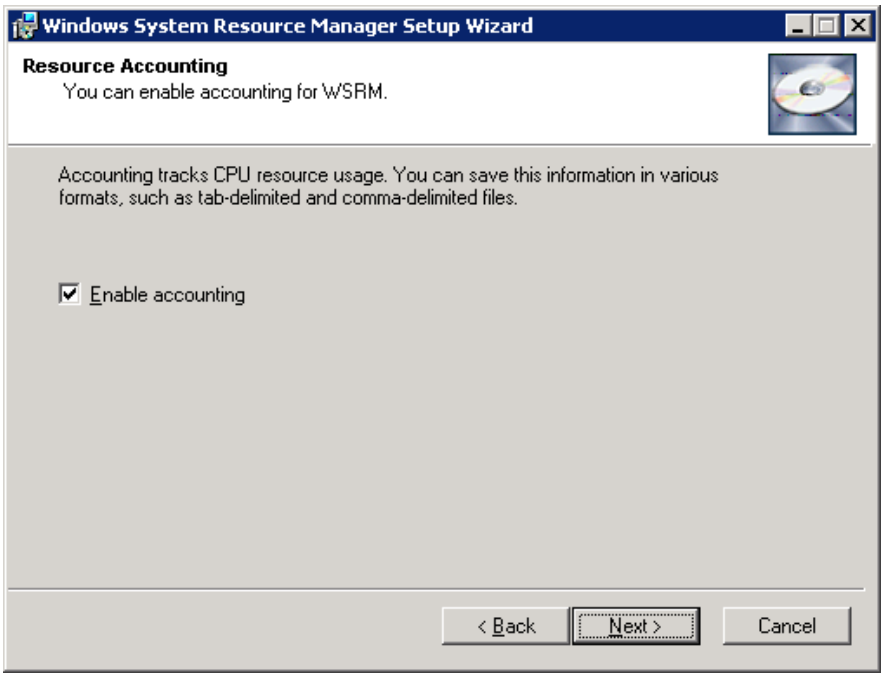

Figure 1: Accounting enabled during WSRM installation. Accounting can also be enabled/disabled from WSRM's administrative UI

Which metrics does WSRM record?

WSRM's accounting is broader in scope than its resource management:

- Accounting tracks all processes, both application *and system*
- Accounting includes metrics for all points of contention, *including I/O*

For each process on the system, WSRM periodically records metrics such as PID, process start time, kernel time, total CPU time, handles, thread count, I/O bytes, authentication context, initial command line, EXE file/path, and so on. The list in Table 1 (below) is from WSRM's help files, with descriptions edited to save space.

Accounting data	Description
Allocation Name	Resource-allocation policy associated with the accounting data
Creation Time	Date/time the accounting record was created
Domain	Domain or workgroup of the user who created the process
Program Path	Directory path to the process's executable file
Kernel Mode Time (MS)	Time in kernel mode
Other Operation Count	Number of I/O operations that are neither reads nor writes (for example, a control function). This counter counts all I/O activity generated by the process

Other Transfer Count (KB)	KBs transferred in input and output operations excluding reads and writes (e.g., by control functions).
Page Fault Count	The number of page faults
Page File Usage (KB)	Amount of virtual memory reserved in paging files
Peak Page File Usage	The maximum amount of virtual memory that this process has reserved for use in the paging files
Peak Virtual Size (KB)	The maximum virtual address space the process has used at any one time.
Peak Working Set Size (KB)	The maximum size of the working set of this process at any point in time
Private Page Count (KB)	Current number of pages allocated that are accessible only to this process.
Process Name	The name of the process associated with the accounting record
Quota Non-Paged Pool Usage	Current non-paged pool usage for the process, in kilobytes
Quota Paged Pool Usage	Current paged pool usage for the process, in kilobytes
Quota Peak Non-Paged Pool Usage	Peak non-paged pool usage for the process, in kilobytes
Quota Peak Paged Pool Usage	Peak paged pool usage for the process, in kilobytes
Read Operation Count	The number of read I/O operations generated by the process, including file, network and device I/Os
Read Transfer Count (KB)	The number of kilobytes read in I/O operations, including file, network and device I/Os
Session ID	The Terminal Services session identifier that owns the process
Thread Count	The number of threads currently active in this process.
Total CPU Time (MS)	The total time, in milliseconds, that the threads of the process used the processor to execute instructions.
User	The name of the user who started the process.
User Mode Time (MS)	The elapsed time the processor spends in user mode
Virtual Size (KB)	The current size of the process’s virtual address space
Working Set Size	The current size of the working set (the set of memory pages

(KB)	recently touched by the threads in the process)
Write Operation Count	The number of write I/O operations generated by the process
Write Transfer Count (KB)	The number of kilobytes written in write I/O operations

Table 1: Metrics tracked by WSRM

WSRM and SQL Server consolidation

SQL Server database consolidation is a reaction against SQL Server sprawl. Rather than accepting *de facto* corporate deployments (large numbers of lightly-loaded systems, each running stand-alone copies of SQL Server dedicated to a single application), businesses are combining database applications and running them on fewer servers.

In planning SQL Server consolidations, a fundamental decision is whether to combine application databases into a *single SQL Server instance*, or whether to host *multiple SQL Server instances*, each containing a single consolidated application. WSRM accounting is well suited to the latter approach (multiple SQL Server instances) because the instances are each exposed as separate OS processes. Therefore, WSRM can track the resources used by each individual DBMS instance, which in this case is linked to a particular consolidated application.

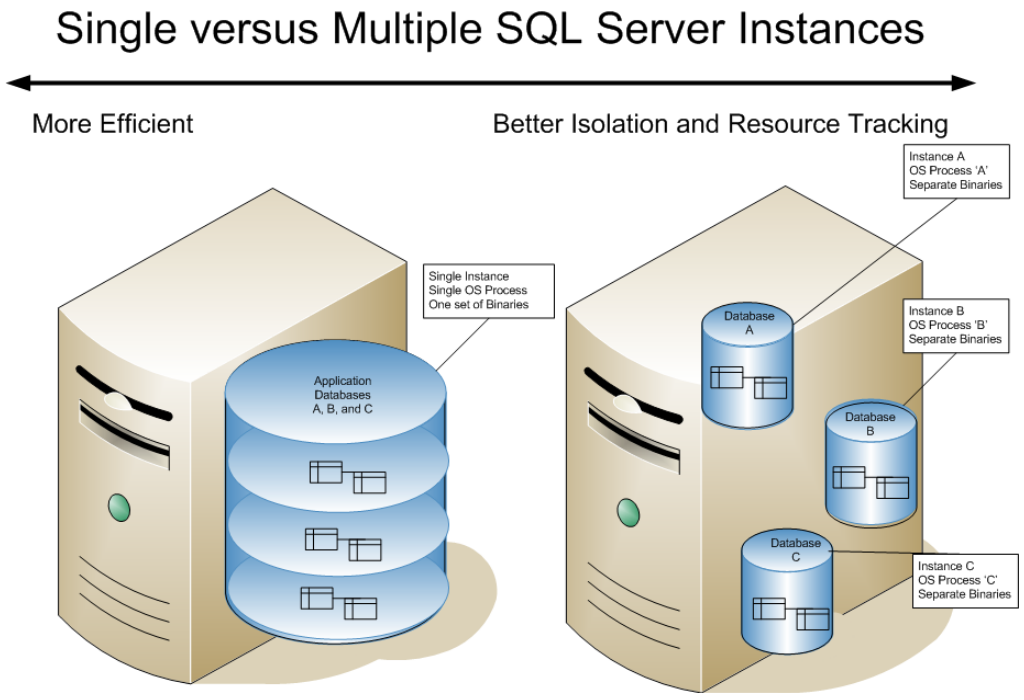

Figure 2: When SQL Server databases are consolidated using multiple instances, each runs as an operating system process that's visible to WSRM

Of course, multiple-instance architectures apply only to those pieces of SQL Server that are instance aware. In SQL Server 2005, this includes most significant processing components including the database engine and SQL Agent, analysis

services, and reporting services. Each instance of these components will run as a separate process, and can be tracked individually by WSRM accounting.

Note that SQL Server 2005's ETL sub-system, Integration Services (SSIS), is *not* instance aware. All resources used by SSIS on a system will occur within a single operating system process, and WSRM won't be able to discern which consolidated application is associated with the ETL activity. In this case, it may be simplest to treat SSIS as a shared resource with costs split among stake-holders.

Identifying specific SQL Server instances in WSRM accounting data

One of WSRM's advantages as a source for chargeback data is that it's easy to link a particular SQL Server instance with the process resources it consumes. WSRM data includes a field ('Command Line') with this information, captured with the start-up command for the service. In the data in Table 2 below, records associated with SQL Server instance "A0000301" (highlighted) can be identified by this parameter.

Time Stamp	Process Id	Command Line
1/20/2007 19:43	8684	C:\...\Microsoft SQL Server\MSSQL.2\MSSQL\Binn\sqlservr.exe -sA0000301

Table 2: With WSRM, SQL Server operating system processes and the resources they use are identified by the corresponding SQL Server instance name

Because WSRM consumption data is linked to the corresponding SQL Server instance, it's far easier to use WSRM data for chargeback than PerfMon data. PerfMon identifies process-level consumption data only by PID (process ID, which varies each service startup) or by PerfMon-generated labels based on start-up sequence (e.g., "sqlservr", "sqlservr#1", etc.). PerfMon data may be customized so that process-level data is identifiably linked to a particular instance[2], but the technique is cumbersome and more intrusive than simply using WSRM.

Although this paper focuses on using WSRM accounting with SQL Server, we should also note that WSRM simplifies tracking Terminal Server sessions. Terminal Server processes are spawned with a user's credentials, and can be tracked accordingly. The length of a user's terminal server session (connected time) can be found by tracking *explorer.exe*, as this shell starts when the user logs on and terminates when the user logs off.

---

[2] If PerfMon data must be used, one solution is to define custom PerfMon counters for each SQL Server instance, and then load these counters with a numeric constant that identifies the instance, cross-referenced with corresponding OS PID (retrieved via a startup query). Refer to the sample below.

```
-- PerfMon sample showing how to load SQL Server "instance markers" at service startup.
-- -- 1) Load a custom counter with a numeric constant identifying the instance
exec sp user counter1 5

-- -- 2) Load another custom counter with the instance's PID (varies)
declare @ProcessID int
SET @ProcessID = CAST((SELECT SERVERPROPERTY ('processid')) AS int)
exec sp_user_counter2 @ProcessID
```

Tips and tricks for using WSRM accounting

While WSRM has significant advantages as a source of chargeback data, developers should be cognizant of the following when building such a system:

Accounting data must be regularly exported

Accounting information accumulates in WSRM's database, which expands accordingly. To keep its size manageable, historical data must be periodically archived and deleted. Additionally, the ESE information isn't easily accessible, and periodic exports will make the information more accessible to downstream software.

```
@Echo OFF
TITLE DumpWSRM
:DumpAcct
REM Periodically export WSRM accounting data
REM ------------------------------------------------------------
Set CURRDATE=%TEMP%\CURRDATE.TMP
Set CURRTIME=%TEMP%\CURRTIME.TMP
DATE /T > %CURRDATE%
TIME /T > %CURRTIME%
REM Swap %%k%%j for different MMDD order
Set PARSEARG="eol=; tokens=1,2,3,4* delims=/, "
For /F %PARSEARG% %%i in (%CURRDATE%) Do SET YYYYMMDD=%%l%%j%%k
Set PARSEARG="eol=; tokens=1,2,3* delims=:, "
For /F %PARSEARG% %%i in (%CURRTIME%) Do Set HHMM=%%i%%j%%k
REM Echo %YYYYMMDD%%HHMM%
wsrmc /get:acc /archive:1 /o:c:\WSRMLogs\WSRM-%YYYYMMDD%%HHMM%.txt
/sd:1/01/06 /ed:1/01/08 /del
REM Wait timeout seconds
timeout /t 600 /nobreak
goto DumpAcct
:END
```

The above command file periodically exports WSRM accounting data to a uniquely-named file (date/time stamped). The highlighted command uses WSRM's command-line interface ("*wsrmc*") to export accounting information in CSV format (*"/archive:1"*), and then remove the exported records ("*/del*").

Total CPU is cumulative, and must be normalized for multiple CPUs/cores

Table 3 below shows a fragment of exported WSRM accounting data. This fragment has been filtered to show only a single process, and includes only three measures, plus a calculation. The calculation (final column) shows the Proportion of CPU used by this process during each measurement interval.

Time Stamp	Total CPU Time (ms)	Elapsed Time (ms)	Proportion CPU, calculated as: (Interval_CPU / Interval_Elapsed)/ CPU_Core_Scaling
1/20/2007 19:51	1538891	7503634	N/A
1/20/2007 19:53	1567157	7623729	0.029

1/20/2007 19:55	1593922	7743776	0.028
1/20/2007 19:57	1620906	7863808	0.028
1/20/2007 19:59	1648516	7983840	0.029
1/20/2007 20:01	1669047	8103888	0.021
1/20/2007 20:03	1686516	8223920	0.018
1/20/2007 20:05	1761344	8343983	0.078
1/20/2007 20:07	2076813	8464046	0.328
1/20/2007 20:09	2468031	8584125	0.407
1/20/2007 20:11	2877063	8704204	0.426
1/20/2007 20:13	3293125	8824268	0.433

Table 3: Subset of WSRM data with "Proportion CPU" calculation.

To calculate the overall proportion of CPU used during each interval, please note:

- "Total CPU Time" accumulates continually, and therefore the CPU time used in any single interval is calculated as that interval's (accumulated) total, less the previous interval's total.
- CPU times are process totals across all threads, summed across all CPUs and cores. In the above example (DL585 with four dual-core Opteron CPUs), the raw aggregate CPU must therefore be divided by 8 (the number of effective CPUs) to convert this value to a proportion of overall system CPU resources.

WSRM accounting data must be cleaned

Raw WSRM accounting data may contain spurious information that will need to be filtered and cleaned as part of any accounting or chargeback process. Data profiling and outlier identification should be part of any dataflow using this information.

WSRM and clusters

Two aspects of WSRM's behavior on clusters should be noted:

- WSRM policies are server-specific, not cluster-wide. A WSRM-managed application which fails-over from Node-1 to Node-2 will then be managed by the policies defined for Node-2 (or will be unmanaged if policies for that application don't exist).
- Cluster automation software, including *Matrix Server* from PolyServe, may stop/start services to automate failover and fallback. In this situation (as well as more generally), remember that WSRM tracks *active* processes, not installed processes. For example, a SQL Service that's stopped will disappear from WSRM's accounting snapshots. Also note that re-started services are assigned different PIDs (Process IDs) by the operating system.

Cluster interactions are sufficiently complex to warrant a whitepaper by Microsoft.

Additional Resources

Microsoft Windows System Resource Manager Overview (KB1413) by Andrzej Switka
http://cgscomm3.inet.cpqcorp.net/Technology/Documents/Knowledge%20Briefs/Q3FY04/KB1413.doc

WSRM Command-Line Interface (Microsoft whitepaper)
http://download.microsoft.com/download/a/7/a/a7a06462-1d80-4386-9505-91cca1e61940/WSRM%20Command-Line%20Interface.doc

Using WSRM and Scripting to Manage Clusters (Microsoft whitepaper)
http://www.microsoft.com/downloads/details.aspx?FamilyID=ba2559e6-dd23-41a6-9efb-1d90f8f1fc17&displaylang=en

WSRM Accounting (Microsoft whitepaper)
http://download.microsoft.com/download/d/2/5/d2524d17-b893-46f9-bebe-b1f7b927e144/Windows%20System%20Resource%20Manager%20Accounting.doc

Acknowledgement

Thanks to reviewers for their time and suggestions; any inaccuracies remain mine.

Summary

WSRM is typically used to allocate CPU and memory resources among competing applications. Less well known is WSRM's accounting capability, which tracks applications' resource consumption. During SQL Server consolidations, WSRM's accounting data may ease the development of chargeback software, allowing system costs to be apportioned according to metered use.

Drew Hamre is a principal of lexana\net and lives in Golden Valley, Minnesota.