SQL Sentry: Overhead Analysis

Aaron Bertrand
Senior Consultant
SQL Sentry, Inc.
July 2011

Table of Contents

Introduction

The purpose of this paper is to provide both qualitative and quantitative information about the overhead impact you should expect from our products, Event Manager and Performance Advisor for SQL Server. We have gone to great lengths to produce consistent and realistic load tests, and to analyze the stresses that our monitoring can place on CPU, memory, network and I/O while a typical load is running against a number of monitored SQL Server instances.

Why are we doing this? Many vendors of various tools will tell you that their solution is “low overhead” or “zero impact” – but very few prove it. We don’t want to try to sell you a tag line, but rather be very up front with you about the true effect our solutions will have on your systems. Aside from capturing metrics in your environment, the next best thing we can do is set up a test environment, run real-world workloads against those systems, and capture performance data. We encourage you to test our software yourself, but in case you don’t have the luxury, we wanted to provide you with realistic expectations. If you are able to perform performance testing in your environment, this will hopefully provide a sensible look into the methodology around such tests.

In the pages ahead, we will describe the environment where we are running the tests, the types of performance metrics we are interested in, and a thorough analysis of those metrics.

Environment

We set up an environment that would provide us with flexible and realistic testing; the goal being to simulate a typical scenario where there are multiple SQL Server instances being monitored, and to measure the various aspects of overhead caused by introducing our monitoring solution to that environment. While in many cases the SQL Sentry server service, repository and console can be run from the same machine, we created separate VMs for each component, as well as a separate monitor machine (from which we would run workloads and collect performance counters). Doing this would enable us to easily isolate the various performance metrics related to each component.

As a quick primer to the architecture of SQL Sentry software, the solution is made up of the following three primary components:

SQL Sentry Server Service

The server service is a Windows service that monitors instances of SQL Server, collects and analyzes performance data, generates notifications, and submits data to the repository. No agent is required to be installed on the monitored targets. Although there are no hard limits, each server service can monitor roughly 100 SQL Servers, and multiple server services can be used for larger environments. In this series of tests, a single server service was used to monitor up to 100 SQL Server instances.

SQL Sentry Repository

The repository is a SQL Server database that stores configuration data, as well as the performance data collected by the server service. The repository allows the console to configure monitoring and to review and analyze the performance and event data. There is a single repository database per SQL Sentry installation, and it can store data for hundreds of monitored servers.

SQL Sentry Console

The console is a .NET client application that can be used to visually monitor and analyze the performance of monitored servers as well as adjust configuration. The console connects to the repository, and does not automatically pull performance or event data from the monitored targets.

For more information on the general architecture of the SQL Sentry solution and the individual repository, server service and console components, see the following page:

http://bit.ly/SQLSentryArchitecture

The primary elements of our test environment setup are an Intel Modular Server (for more details, see http://intel.ly/SQLSentry-IMS), with four physical hosts (Intel Compute Modules, or ICMs) utilizing Hyper-V to host the SQL Sentry component VMs and up to 100 monitored target VMs. The four ICM hosts are each equipped with 2 quad-core, Intel Xeon X5550 2.66 GHz CPUs and 48GB of RAM, with three hosts participating in a cluster. The underlying disk is comprised of an Intel SAN (14 x 146GB 15K SAS) and a Promise VTRAK array (16 x 300GB 15K SAS). The Intel SAN hosts the CSVs (clustered shared volumes) for VM Store 1 and 2 (both of which are 1.36TB), used to host the VHDs for the monitored

target VMs. The VTRAK array hosts dedicated single-drive LUNs for the SQL Sentry component VMs (the monitoring server, the SQL Sentry server service, and the SQL Sentry console), as well as three dedicated LUNs for the SQL Sentry repository (one for the operating system, one for SQL Server data files, and one for SQL Server log files). The network connections are all 1GB, and there are three separate networks in effect: the domain network, the external virtual network (this represents communication between all test VMs and hosts in the cluster), and a private cluster network (which keeps all CSV and other cluster traffic from interfering with the other networks).

The ICM hosts (running Windows Server 2008 R2 and Hyper-V) are described here:

ICM01 – SQL VM host using VM Store 1 for VMs 68-83, and VM Store 2 for VMs 84-100
ICM02 – SQL VM host using VM Store 1 for VMs 1-34
ICM03 – SQL VM host using VM Store 2 for VMs 35-67

Each SQL guest runs Windows Server 2008 R2 Standard Edition, and has a single default instance of SQL Server 2008 Service Pack 1 (10.0.2531), with 10 sample databases and a copy of AdventureWorksLT2008. The SQL guest VMs are configured with 1 CPU, 1GB RAM, and a 40 GB fixed disk.

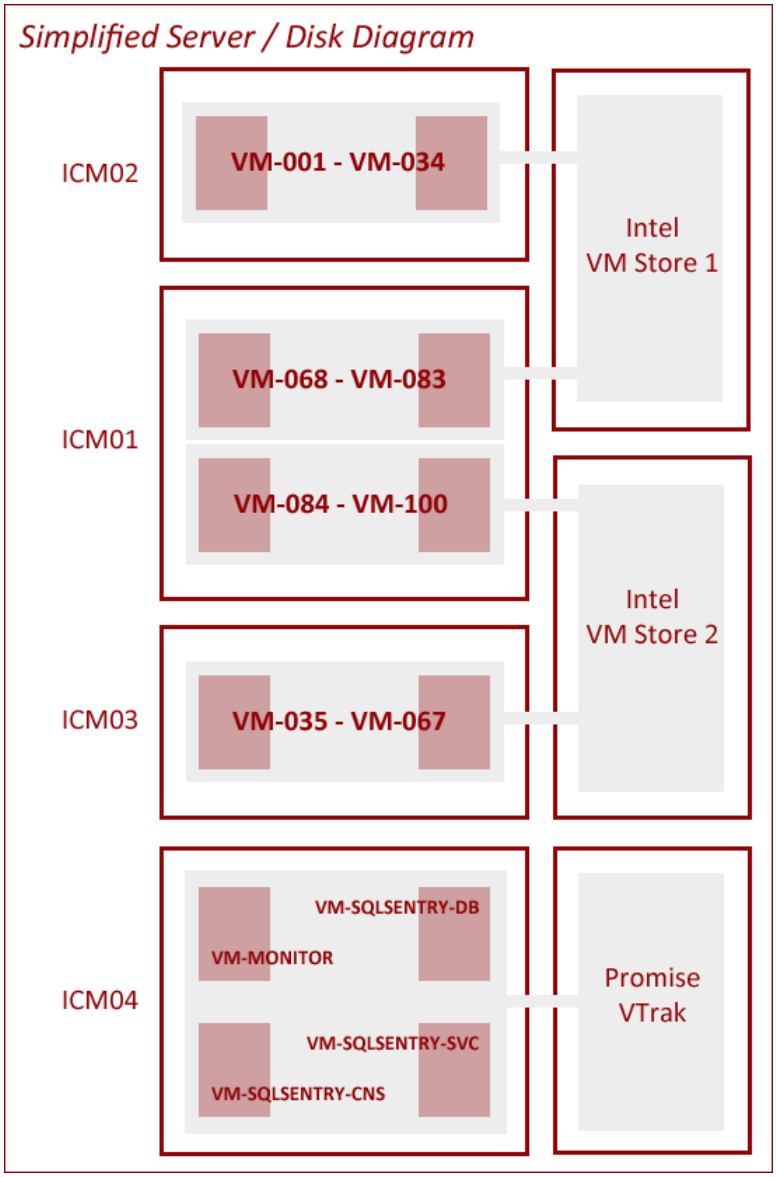
The split across the three physical hosts and the two disk stores is an effort to balance the number of VMs per host (33 or 34) and the number of VMs using each dedicated disk store (50).

ICM04 – standalone host with the following VMs, using SQL Sentry v6.1.33:

VM-SQLSENTRY-DB
Hosts SQL Sentry repository – 2 CPU, 8GB RAM
Fixed drives: 40GB (OS), 65GB (data), 20GB (log)
VM-SQLSENTRY-SVC
Runs SQL Sentry server service – 2 CPU, 2GB RAM
Fixed drive: 40GB
VM-SQLSENTRY-CNS
Runs SQL Sentry console – 1 CPU, 1GB RAM
Fixed drive: 40GB
VM-MONITOR
Runs workloads and performance monitoring – 1 CPU, 8GB RAM
Fixed drive: 40GB

To minimize any resulting performance skew, we were careful to ensure that over-committing of CPU did not occur on ICM04. The host has 8 cores, but only 6 cores and 19GB of memory would be allocated to the guest VMs. To isolate I/O as much as possible, each virtual machine within this host has a dedicated spindle for its disk(s).

Here is a simplified diagram of the servers listed above and how they utilize the disk pools:



For more details on the test environment and configuration, please refer to Appendix A, "Test Environment Details."

Test Methodology

We wanted to run workloads with minimal side effects – providing reasonably verbose and repeatable activity to monitor without overwhelming SQL Server, the virtual machines, or the underlying Hyper-V hosts, and with the ability to run the exact same tests over and over again.

There were two sets of metrics we wanted to collect: impact on the monitored SQL Server instances, and impact on the servers performing the monitoring. To facilitate the measurement of the impact on the servers performing the monitoring, we ran each SQL Sentry component (server service, repository, and console) on its own dedicated virtual machine. Having plenty of experience in performance monitoring, it was relatively easy to draw up the following table to layout which metrics we would observe during our tests:

Category	*Metric*	*Monitored Target*	*SQL Sentry Server Service*	*SQL Sentry Repository*	*SQL Sentry Console*
CPU	avg % processor time	◆	◆	◆	◆
	peak % processor time	◆	◆	◆	◆
	context switches/sec		◆		
Memory	buffer pool (MB)	◆		◆	
	plan cache (MB)	◆		◆	
	working set (MB)		◆		◆
	soft page faults/sec		◆		
Disk	database size (MB)			◆	
	physical read bytes/sec	◆		◆	
	physical write bytes/sec	◆		◆	
Network	Mbps received	◆	◆	◆	
	Mbps sent	◆	◆	◆	
	Packets received/sec	◆	◆	◆	
	Packets sent/sec	◆	◆	◆	
SQL Activity	batch requests/sec	◆		◆	
	transactions/sec	◆		◆	

Test Matrix

The tests would involve the workload (described below) running against a set of servers, and then measuring the performance counters in various scenarios: no monitoring, monitoring with Event Manager only, monitoring with Performance Advisor only, and monitoring with both Event Manager and Performance Advisor. In addition, the tests would measure the impact on the monitoring components when they were watching 1, 10, 25, 50 and 100 servers. The following table outlines the set of 21 tests that would be run:

Number of servers	Workload Only	Event Manager	Performance Advisor	Event Manager + Performance Advisor
0	◆			
1	◆	◆	◆	◆
10	◆	◆	◆	◆
25	◆	◆	◆	◆
50	◆	◆	◆	◆
100	◆	◆	◆	◆

The Workload

Based on trace analyses of multiple real-world customer systems, we developed a workload that was similar to the real environments in terms of batch size, plan cache variance, and number of events per minute that exceed Top SQL collection thresholds (this is 5 seconds by default). The simulated workload was designed to look as realistic as possible to any external monitoring processes, but at the same time, to incur minimal load on the monitored target. SQL Sentry doesn't know or care whether a query incurred actual load on the target; it only cares whether it exceeded the threshold, in which case enhanced collection kicks in. The important criteria are that it looks exactly like a real query – it will still incur the same network overhead to collect, processor time to normalize parameters, and database space to store.

The goals of all this careful planning were:

1. To avoid excessive load on the VM hosts from running 33+ VMs simultaneously, which could potentially skew results of the tests.

2. To reduce the signal-to-noise ratio on the monitored target, making it easier to deduce the actual overhead incurred by SQL Sentry.

3. To generate the same number of Top SQL events that we had observed in our customers' systems (one every five seconds, or 12 per minute).

Since Performance Advisor collects more extended data when a query runs longer than a defined threshold, we wanted to be sure that our simulated workloads generated a reasonable number of these events – but not too many, as we wanted the tests to represent the real-world workloads that we had observed. By varying the parameters to our stored procedures, we could intentionally force a certain number of queries per minute to exceed thresholds and spur the more extended data collection. By adding jobs that would cycle through various parameters, we could also trigger occasional blocking and deadlock events, as well as job runtimes exceeding default thresholds, using the sample databases. Finally, we made sure that we had enough plan variance through ad hoc queries and forced recompiles to represent a good cross-section of the queries we know are out there today – we could ensure plan cache variance both by forcing recompiles and by running queries from PowerShell using both ad hoc queries and procedure calls via RPC.

The following blog post describes how we run the workloads using remote PowerShell sessions to minimize direct impact on the SQL Server virtual machines:

http://bit.ly/AB-RemoteWorkloads

And this blog post describes our performance collection for these tests, which also uses PowerShell to collect counters remotely:

http://bit.ly/AB-PSCounters

Test Results

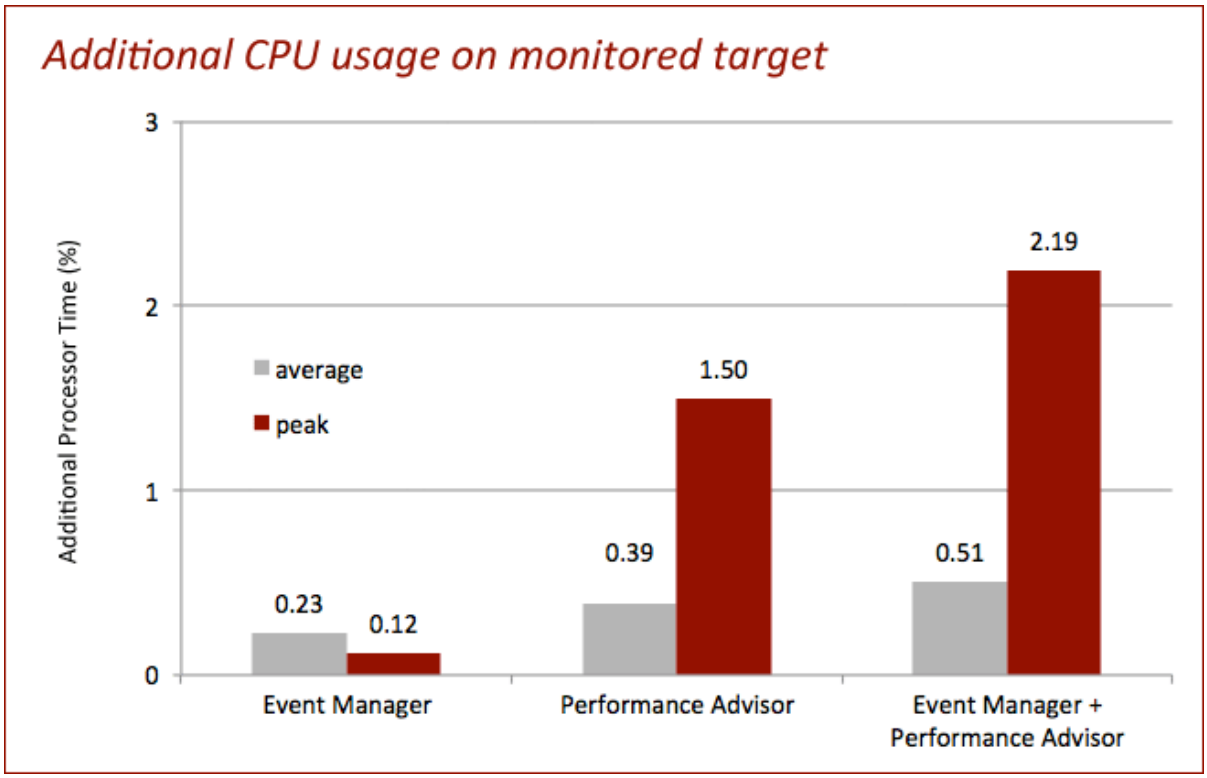
As described earlier, we ran a total of 21 60-minute tests. What follows are the details and analysis of each individual result. We did not find a lot of surprises here, though we were quite happy that some recent investments have really paid off in terms of resource usage. We'll walk through the additional resource utilization we observed on the monitored instance, and then the impact on the SQL Sentry server service, repository and console while monitoring 1, 10, 25, 50 and 100 servers. Finally, we will summarize the results to give you a rough idea of the overhead you should expect when you implement SQL Sentry in your environment.

Monitored Instance

To investigate the overhead that monitoring places on an instance of SQL Server, we took various performance counter metrics from one of the monitored targets. First we took baseline metrics when running a workload against that target for one hour, then repeated that same workload while monitoring the instance with Event Manager alone, Performance Advisor alone, and finally with both Event Manager and Performance Advisor together.

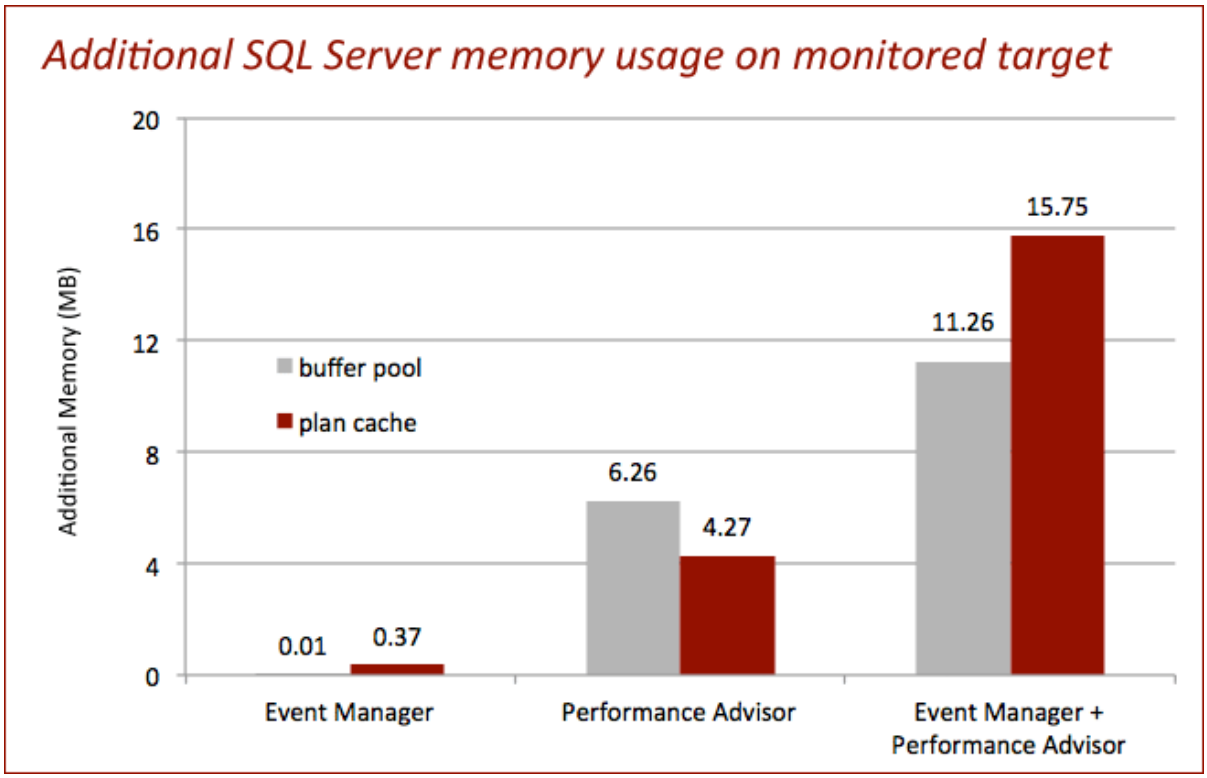
Average and peak CPU

You can see that on a monitored instance, the average CPU overhead is hardly noticeable. With monitoring, the additional average CPU burden on the monitored target ranged from 0.23% with Event Manager alone to 0.51% with both Event Manager and Performance Advisor. We suggest that if your servers do not have room to add an average of less than 1% CPU for monitoring overhead, there are probably much more important problems to solve.



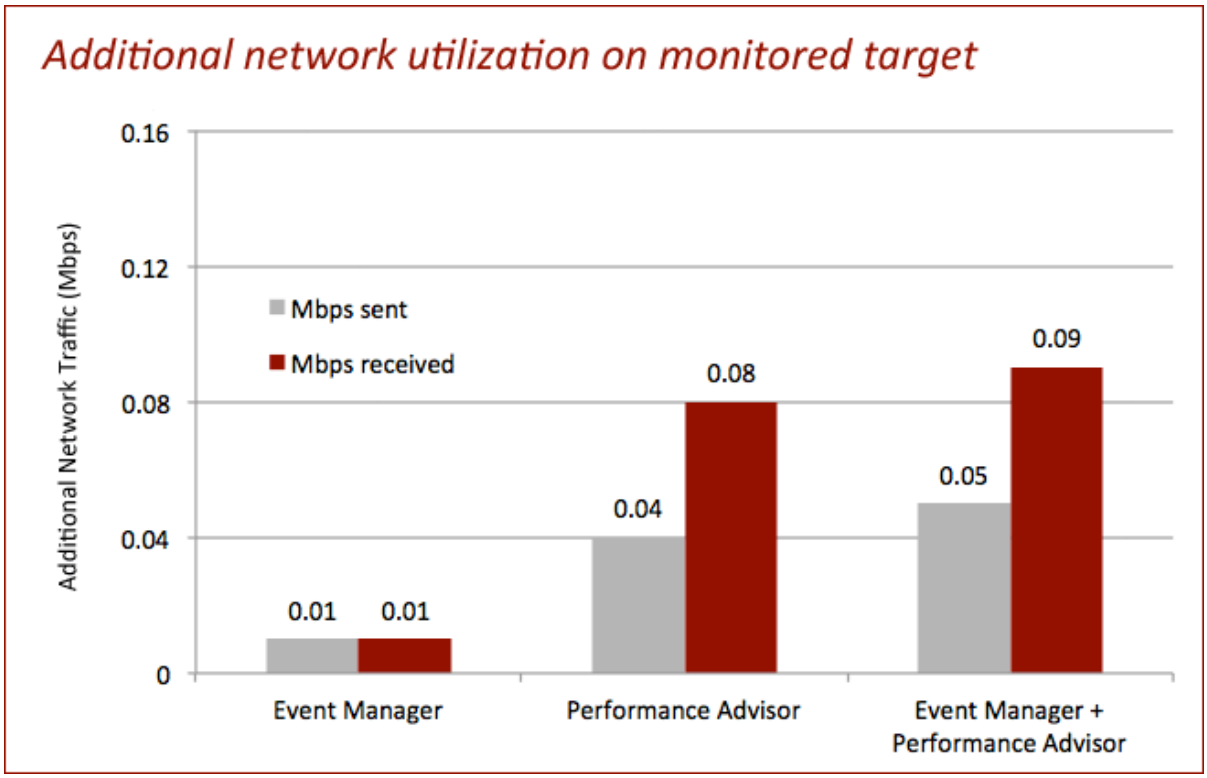
SQL Server Memory (plan cache and buffer pool)

Like CPU utilization, the memory overhead for SQL Server when being monitored with Event Manager, Performance Advisor, or both was minimal. The additional memory required to monitor an instance with Event Manager alone worked out to about 40 KB, almost exclusively in plan cache. For Performance Advisor, less than 11 additional MB were required, and when both monitoring features were combined, the memory usage compared to the workload with no monitoring increased by about 27 MB. About 16 MB of this delta was in plan cache, demonstrating the effort that has been put into SQL Sentry to minimize plan cache footprint.



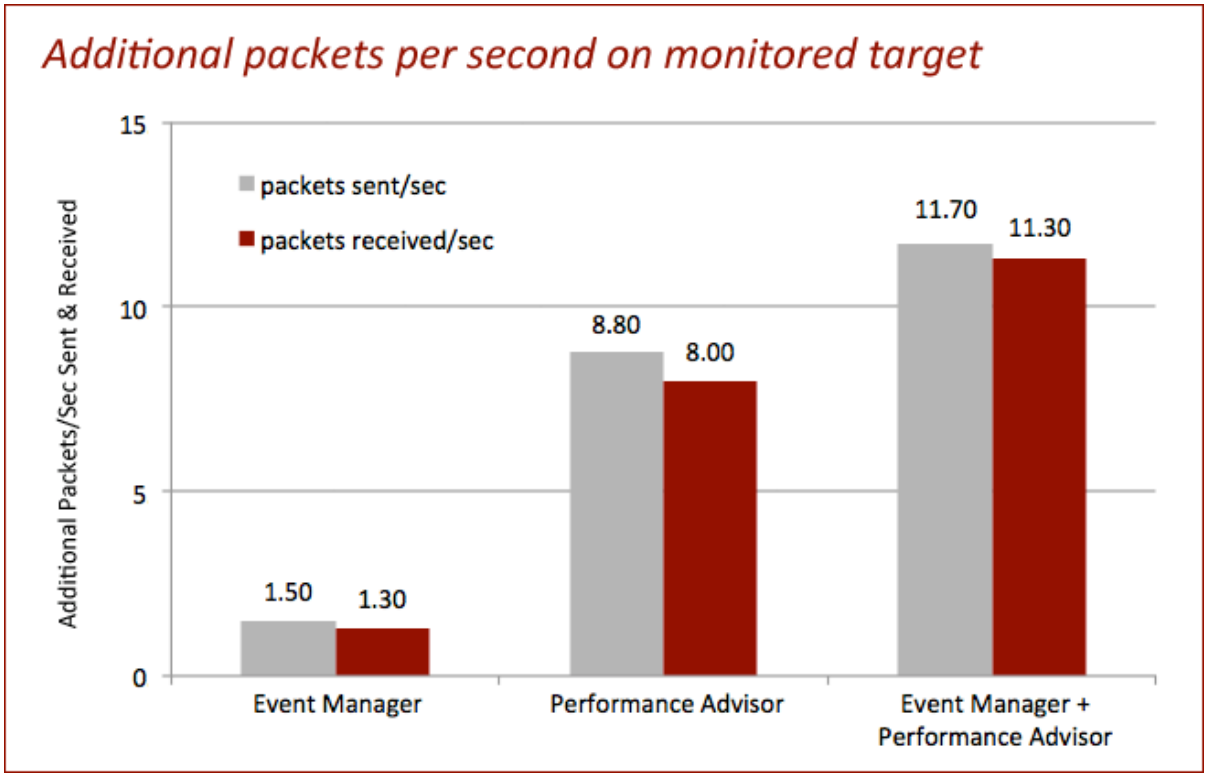
Network (Mbps received / sent)

In order to determine the network traffic caused solely by monitoring an instance, we measured the bytes sent and received through the adapter on the physical host, IMC04, connected to the other IMS machines. While it would be difficult to eliminate all other traffic (for example, the performance counters being collected), this allowed us to ignore as much unrelated network traffic as possible – by using a different network interface, we could ignore both the data flowing between the service and the repository, and any traffic against the monitored target due to the running workloads. The results show that when monitoring is enabled, the additional network load ranges from 0.02 to 0.14 Mbps. In most scenarios, even on a network limited to 100 Mbps, this additional data flow will be negligible.



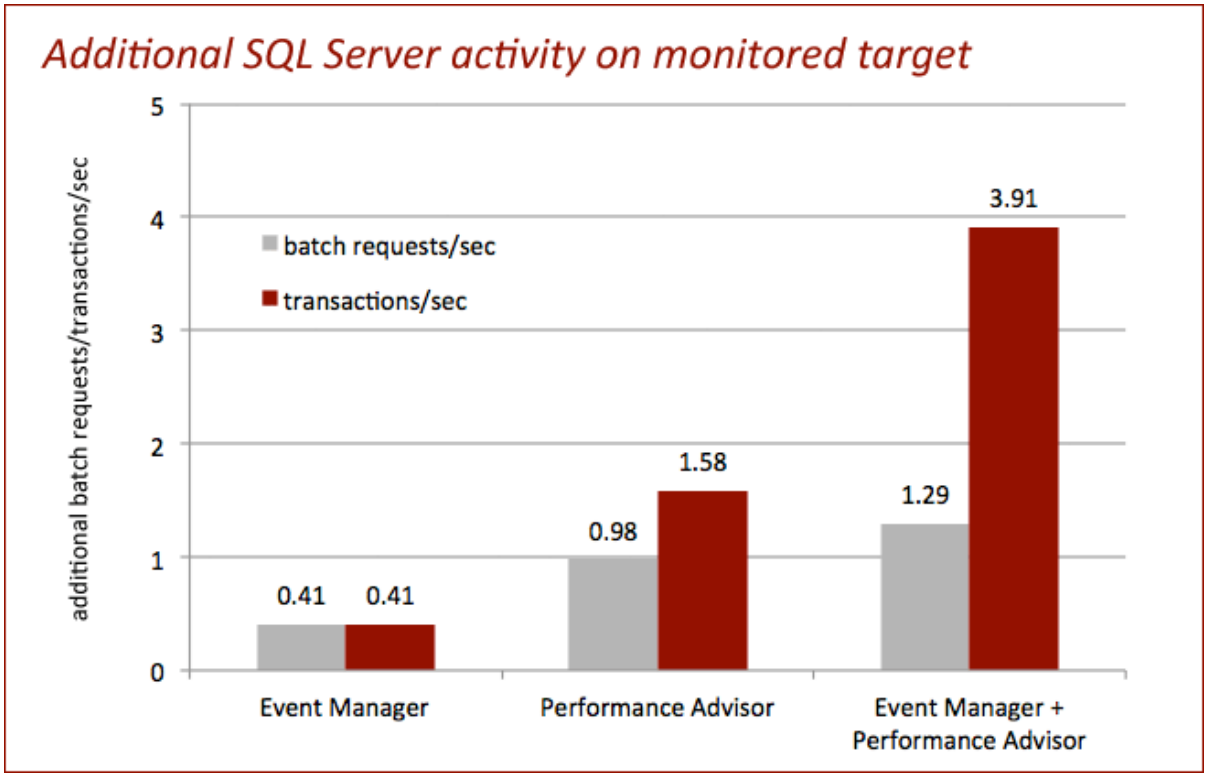
Network (packets / second)

Correspondingly, the additional packets moving between the service and the monitored target when monitoring are negligible as well. Watching an instance with both Event Manager and Performance Advisor yields an increase of about 12 packets per second coming in, and another 12 packets per second going out.



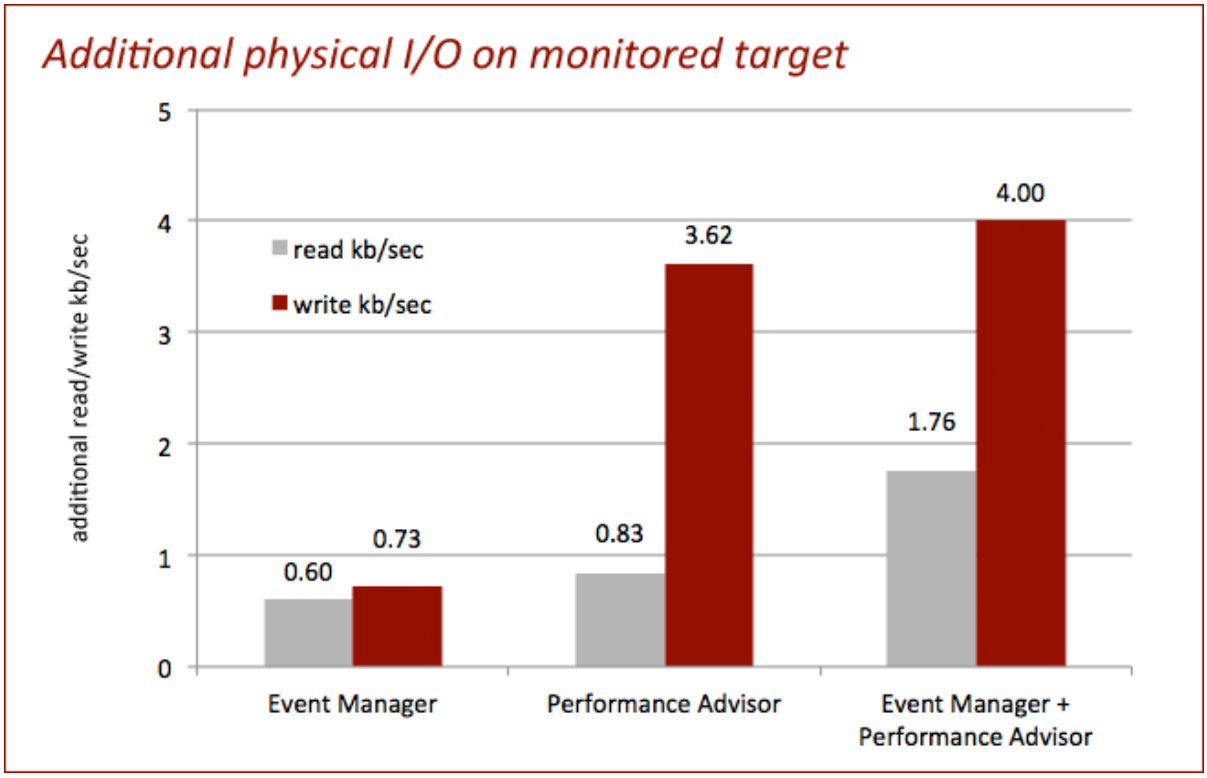
SQL Server activity (batch requests and transactions / second)

Two of the counters often used to judge the "busy-ness" of a server are batch requests per second and transactions per second. Since SQL Sentry issues frequent lightweight queries, it is no surprise that monitoring will create some minor overhead in this area. You can see in the following chart that full monitoring with both features had the greatest impact: average transactions per second went up by 3.91, and average batch requests per second increased by 1.29.



I/O (physical disk read bytes/sec, physical disk write bytes/sec)

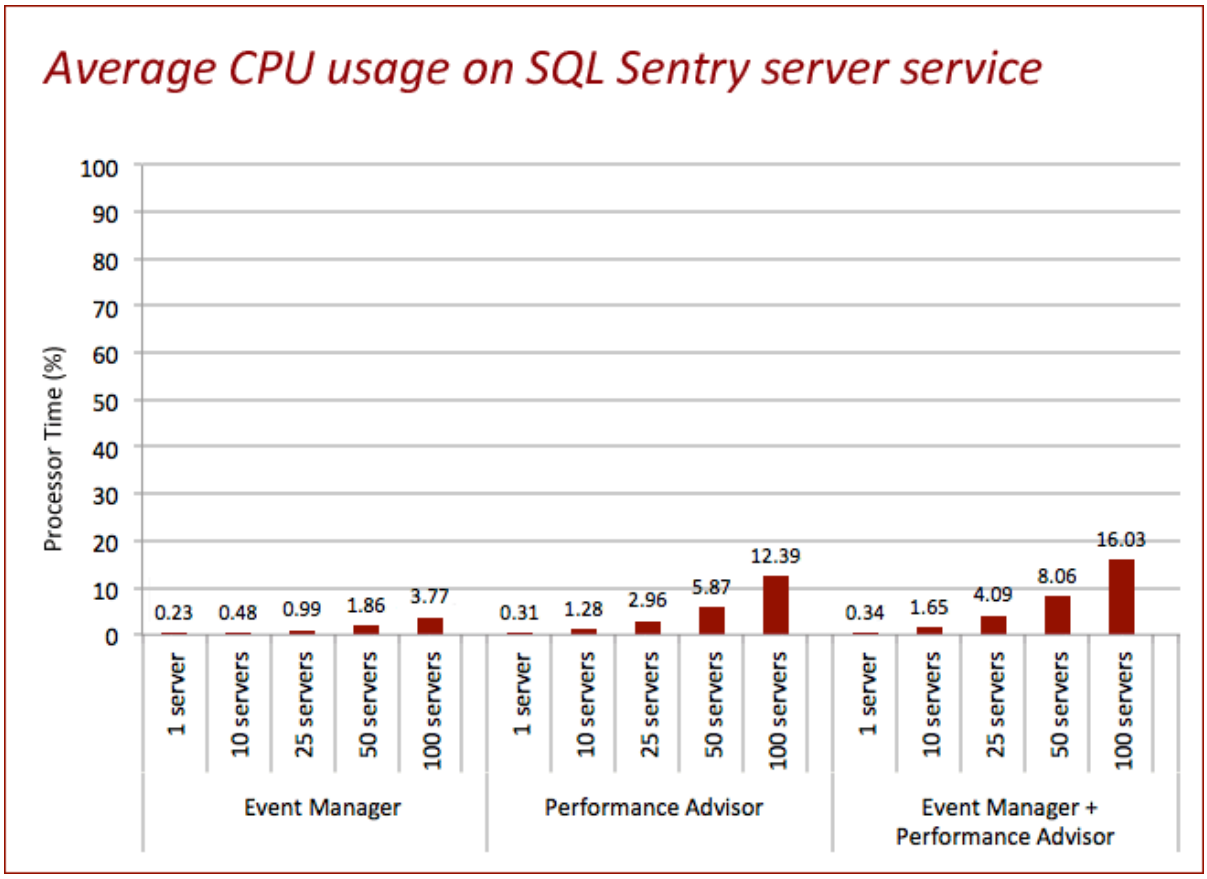
Again, we see that the monitoring solution has some impact on these metrics, but nowhere near any kind of tipping point. When monitoring with Event Manager, the read rate increases by about 0.6 kb per second, and the write rate increases by roughly 0.73 kb per second. With both Event Manager and Performance Advisor, the read and write rates increase by 1.76 and 4 kb per second, respectively. For the monitored target, these measurements include both data and log activity against all databases.



SQL Sentry Server Service

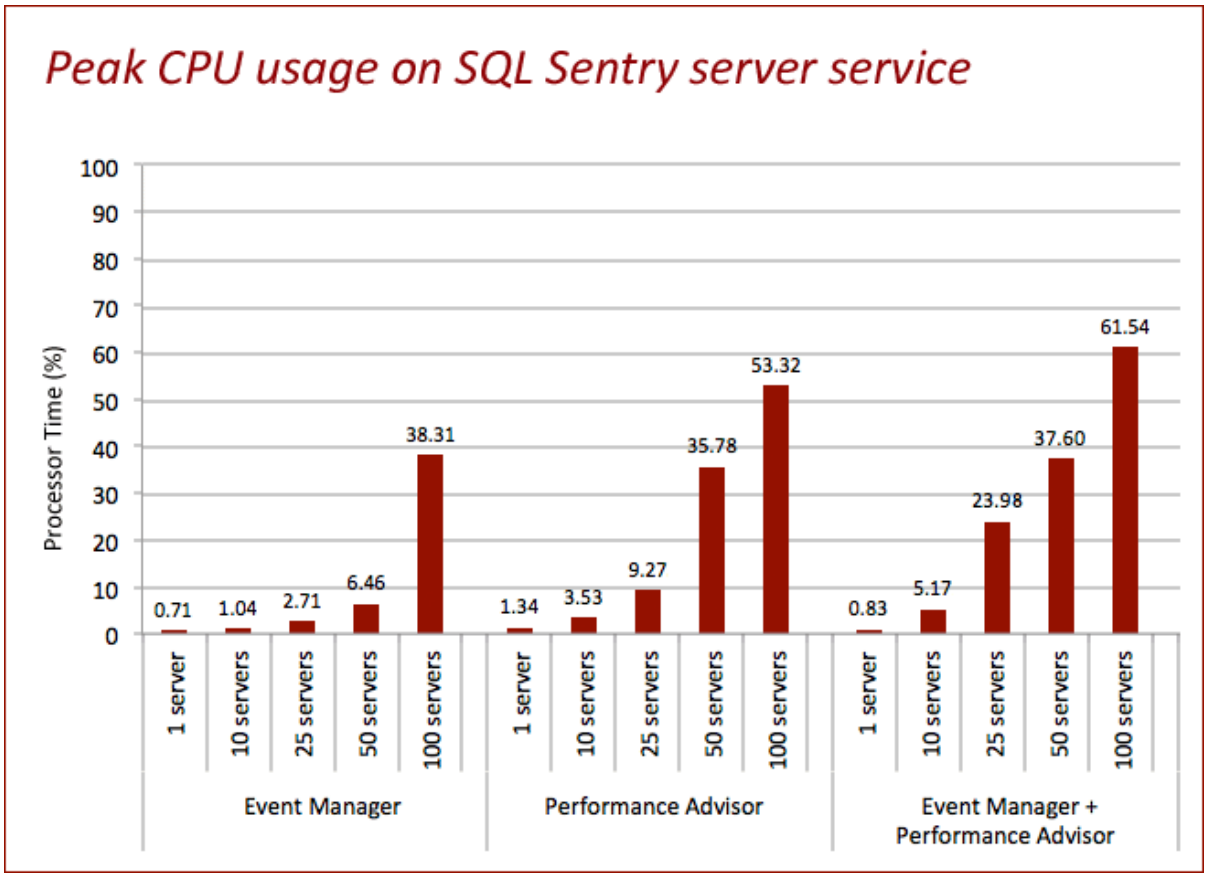
Average CPU

The following graphs show average and peak CPU usage on the machine running the SQL Sentry server service while monitoring 1, 10, 25, 50 and 100 servers with Event Manager, Performance Advisor, and both features combined. This shows that the average CPU load on the server service, when monitoring 100 instances with both Event Manager and Performance Advisor, is just over 16%; with a more modest 50 servers, the load drops to about 8%. This is again on a virtualized, 2-CPU box with 2GB of RAM.

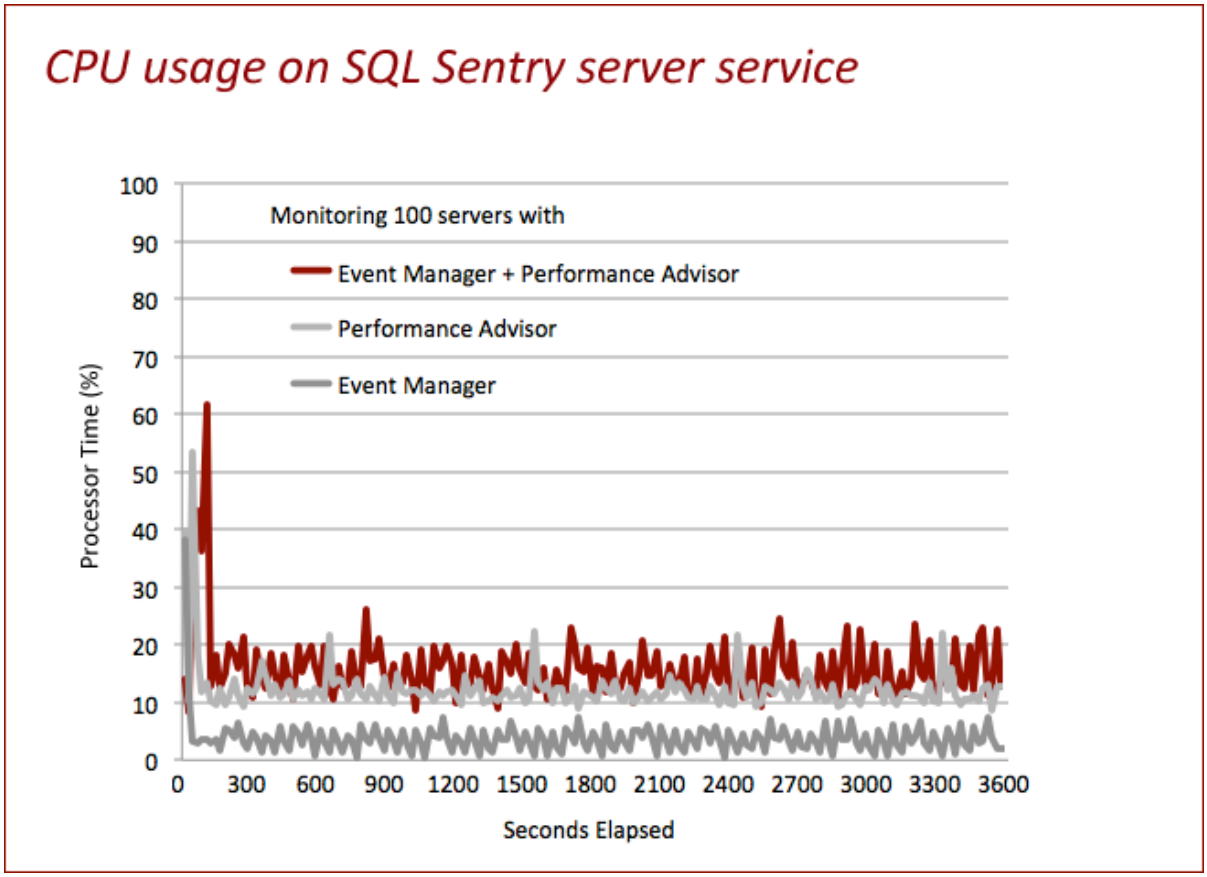


Peak CPU

If we take a look at peak CPU utilization, the data trends in the same way: the highest observed usage (about 61.5%) occurred when monitoring 100 servers with both Event Manager and Performance Advisor.

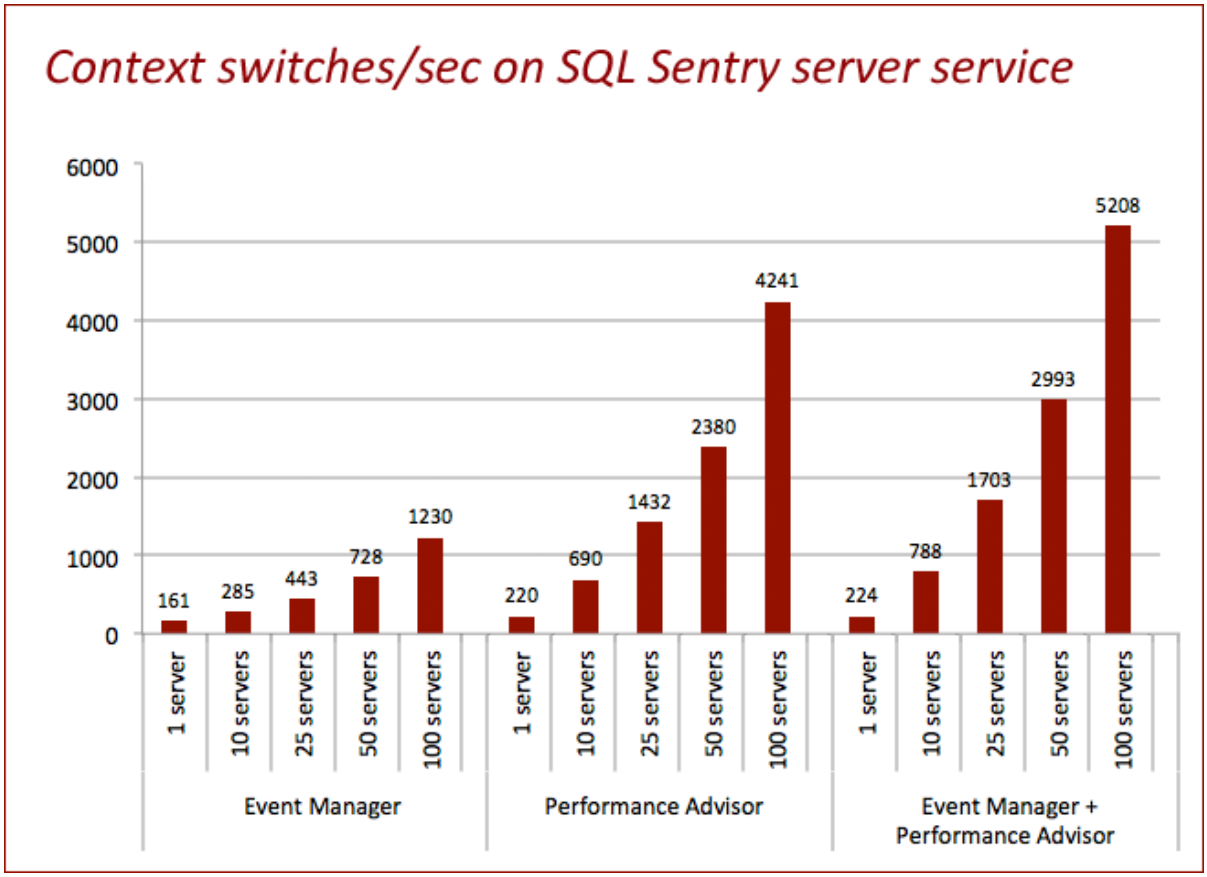


In addition, if we plot CPU utilization in a series, we can see that the spikes noted above are not regular, but rather seem to be related to initialization and startup costs. Throughout the rest of the hour-long workload tests, CPU usage remained stable.



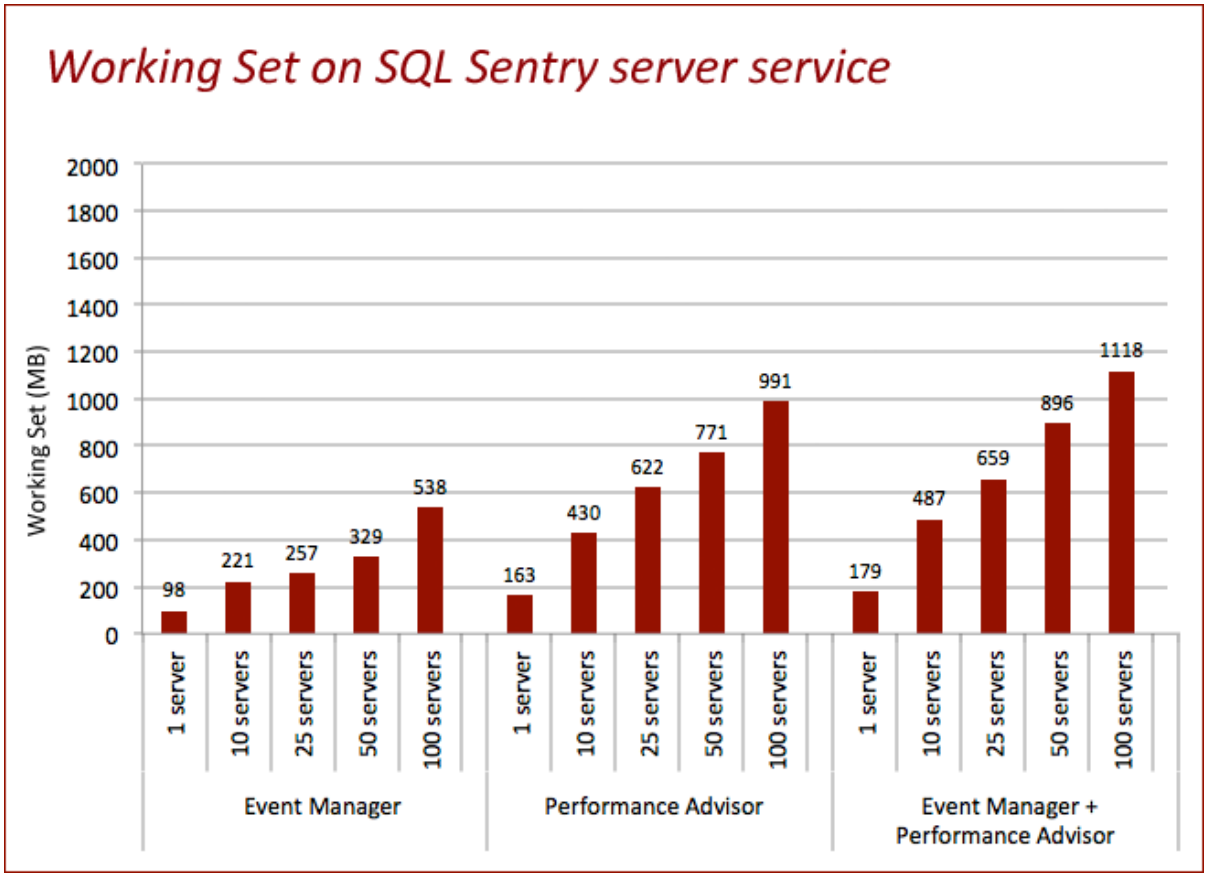
CPU (context switches/sec)

As you might expect, when the server service is handling a large number of monitored servers, it is going to be relatively busy. The following graph shows what you should expect in terms of context switches per second – to the tune of just over 5,200 per second per monitored server at the high end. Notice that there isn't a very high cost to adding Event Manager on top of Performance Advisor monitoring. Event Manager has a lower overall CPU burden on the service, since it is not doing as much inline processing of the raw data before storage.



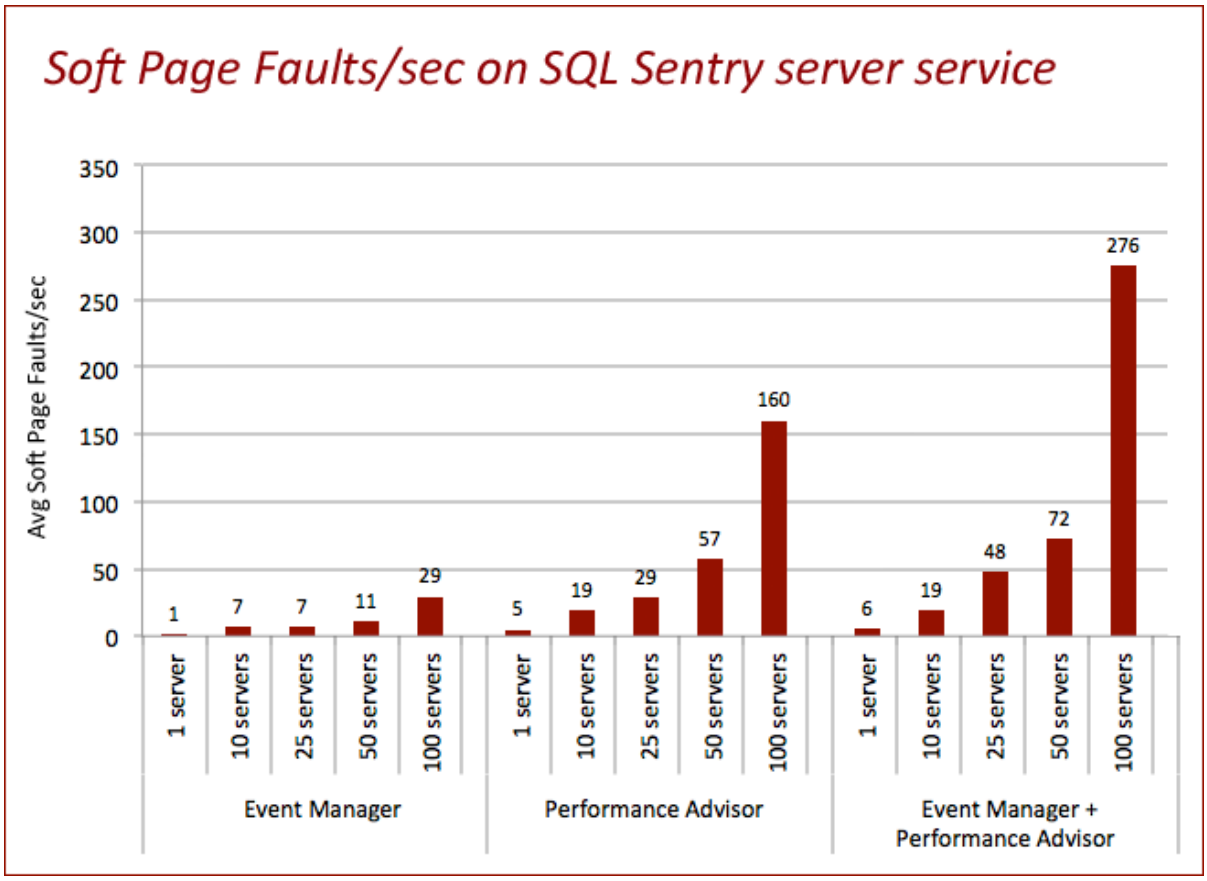
Memory (working set)

The range for required memory for the SQL Sentry server service goes from about 98 MB for 1 monitored server, up to 1.1 GB for 100 monitored servers. There are quite noticeable economies of scale at work here, and it works out to about 11 MB per monitored server over time. And once again, the combined memory footprint for monitoring both Event Manager and Performance Advisor from the same server service ranges from only 1% to 16% higher than monitoring with Performance Advisor alone.



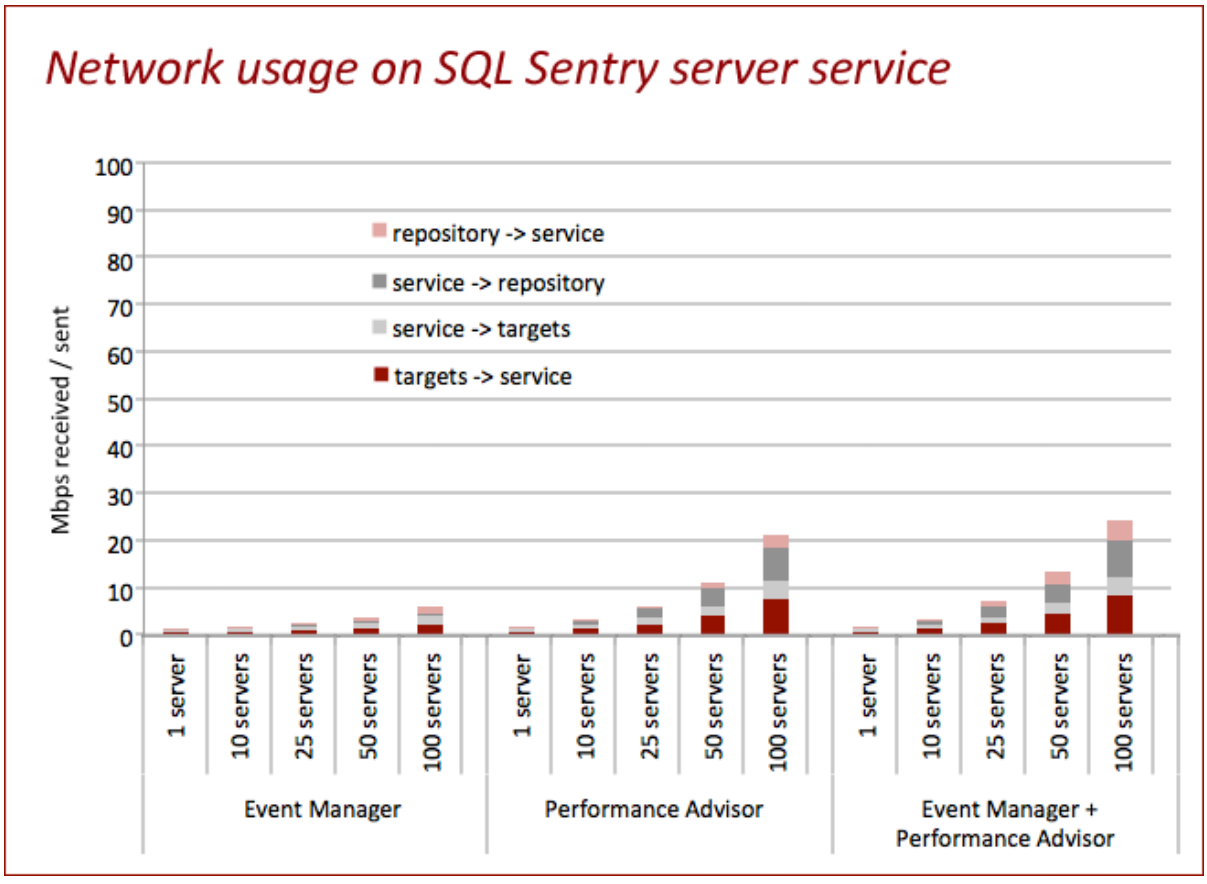
Memory (soft page faults / second)

Page faults per second can be a valid indicator regarding an application's use of physical memory and memory pressure on the server as a whole. While performance monitor allows you to monitor the overall page faults for a specific process, you can use a couple of other system-wide counters (pages input/sec and pages output/sec) to deduce whether these page faults are hard (meaning the process had to go to disk for the page) or soft (meaning the application just had to go elsewhere in its own space). Since pages input/sec and pages output/sec never averaged more than 0.5, we can therefore see that the page faults registered by the SQL Sentry server service are, in fact, soft page faults. During the hour-long workload tests, here are the results:



Network (Mbps)

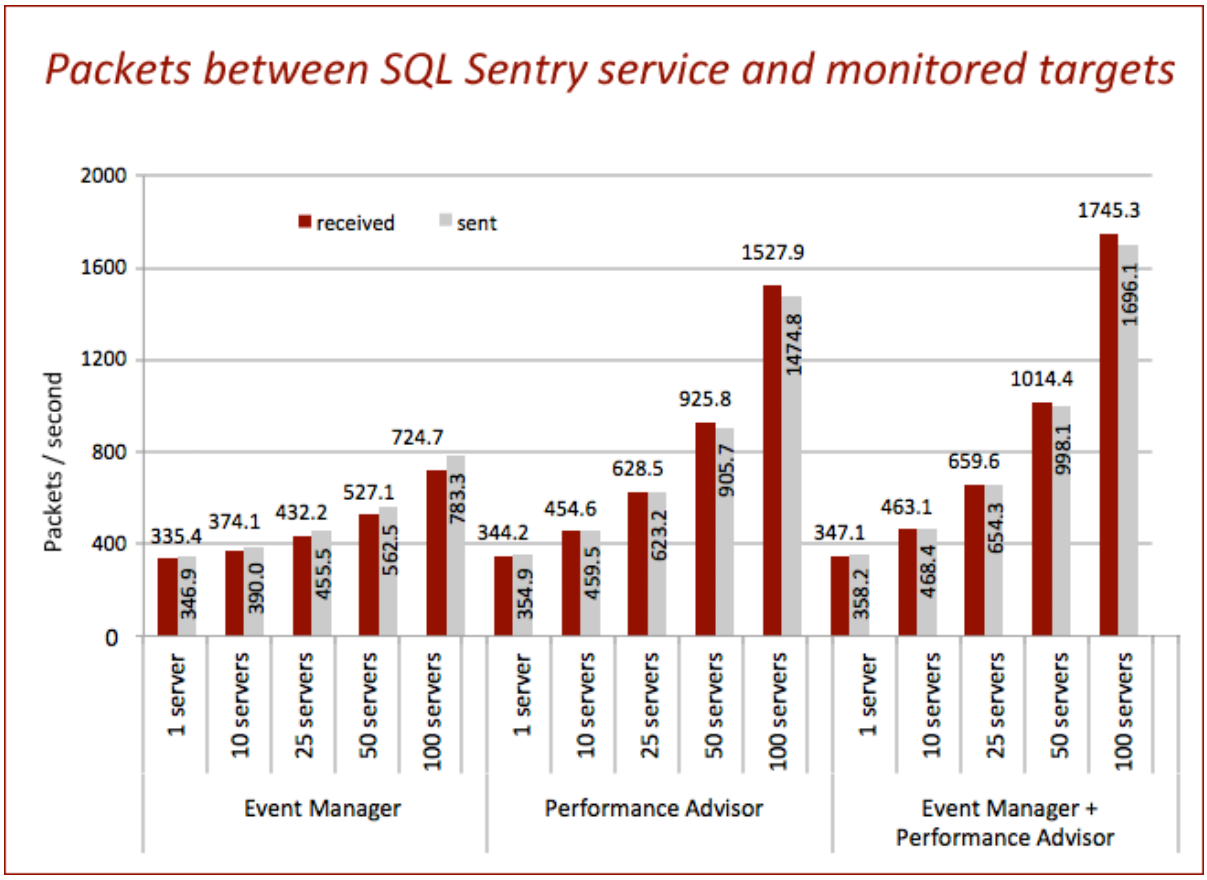
The SQL Sentry server service is all about moving data, and has two main pieces of functionality that utilize network resources to do so: reading data from monitored targets, and writing data to the repository. In order to separate this activity, even on a virtual machine with only one virtual network adapter, we measured the network utilization from "the other side." To get an idea of the network traffic involving the monitored targets, we looked at a specific adapter on ICM04, the physical host; and to see network activity going to the repository, we measured from the repository machine. It is non-trivial to eliminate all network noise from the performance counter data, including the performance collection itself, the workloads, and other essential services – but this should give a fairly decent picture of network utilization in a typical environment. The data below has been normalized from bytes/second to Mbps (by dividing the counter values by 1024*128). At the very high end, when monitoring 100 servers with both products, the total bandwidth is roughly 24.2 Mbps, or 2.4% of a 1000 Mbps pipe. The chart below is scaled to 100 Mbps.



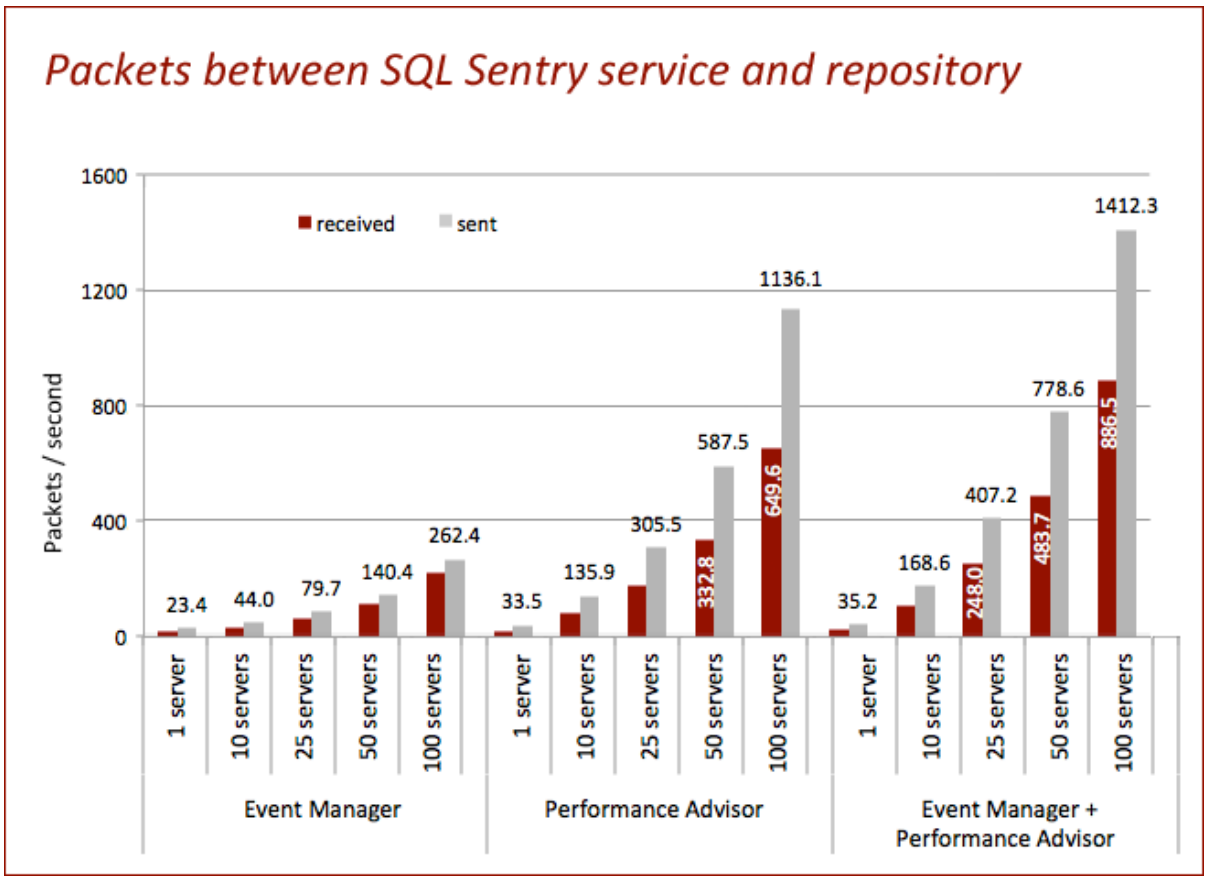
Note that this represents total impact on network infrastructure for our test environment. Depending on how your server service(s) and repository are configured, this may represent traffic that is moving over the same or different network adapters, so should not necessarily be interpreted as a sum of network impact. Other factors that can affect throughput are inherent latency, number of routers, and NIC configuration.

Network (packets / second)

We can also look at the packets transmitted per second, to ensure that we are not encroaching on any limitations here (which will depend largely on packet size and network configuration). The following graph represents the packets transmitted between the server service and the monitored targets with an MTU of 1,500. You can see that under a full duplex configuration, you will see far less than 2,000 packets per second – even when monitoring 100 servers with both products.



And finally we can look at the packets transmitted between the repository and the server service. These numbers come in a little lower (less than 1,500 packets per second at the high end) than the number of packets between the service and the monitored targets, both because the service performs some work on the data before it is sent to the database, and because it doesn't have to read a lot of data back from the repository.



While this is certainly not threatening to most modern network configurations, it may be wise to ensure the server service and/or repository machines are on Gigabit or better once you move beyond monitoring 50 servers. For more information on the throughput of various network devices, including maximum throughput given a defined capacity (e.g. 100Mbps), see the following Wikipedia and Cisco articles:

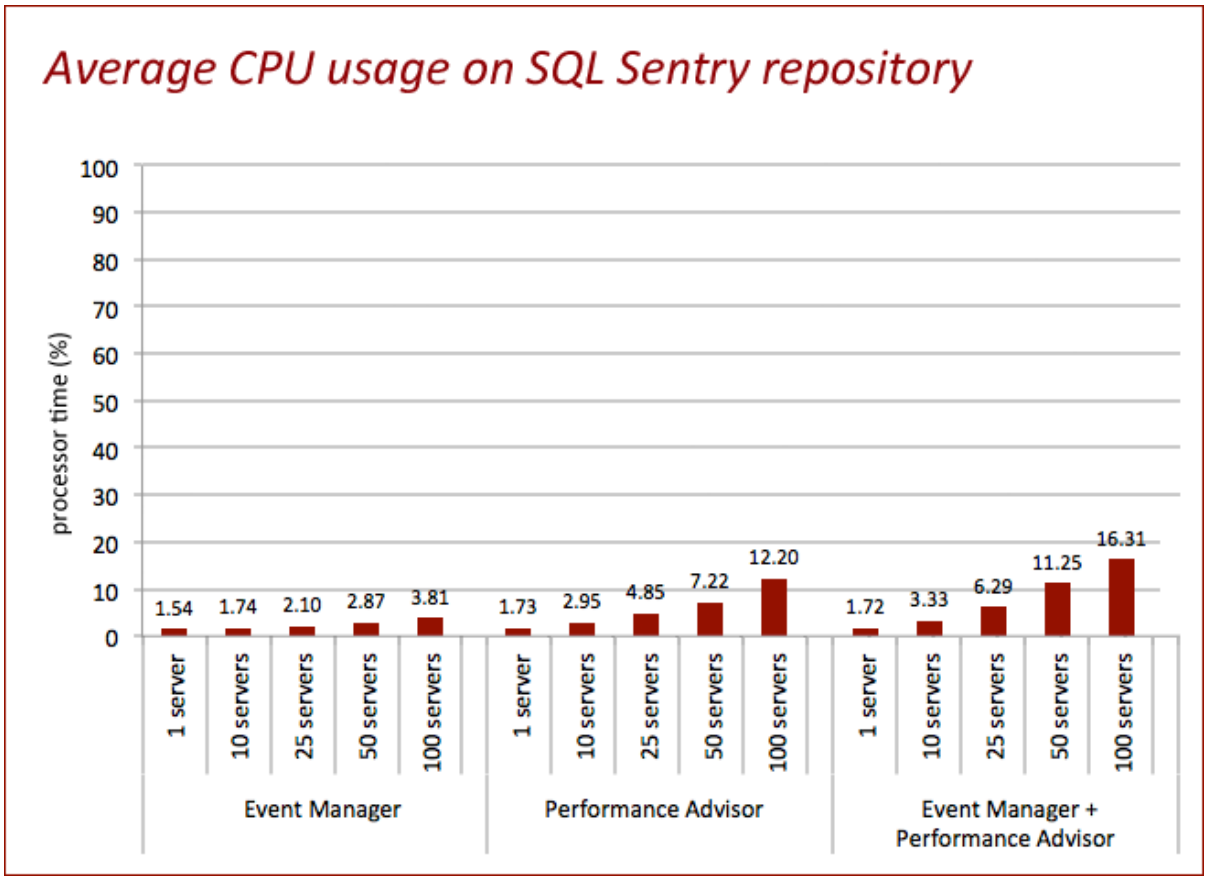
http://en.wikipedia.org/wiki/Throughput

http://bit.ly/Cisco-Throughput

SQL Sentry Repository

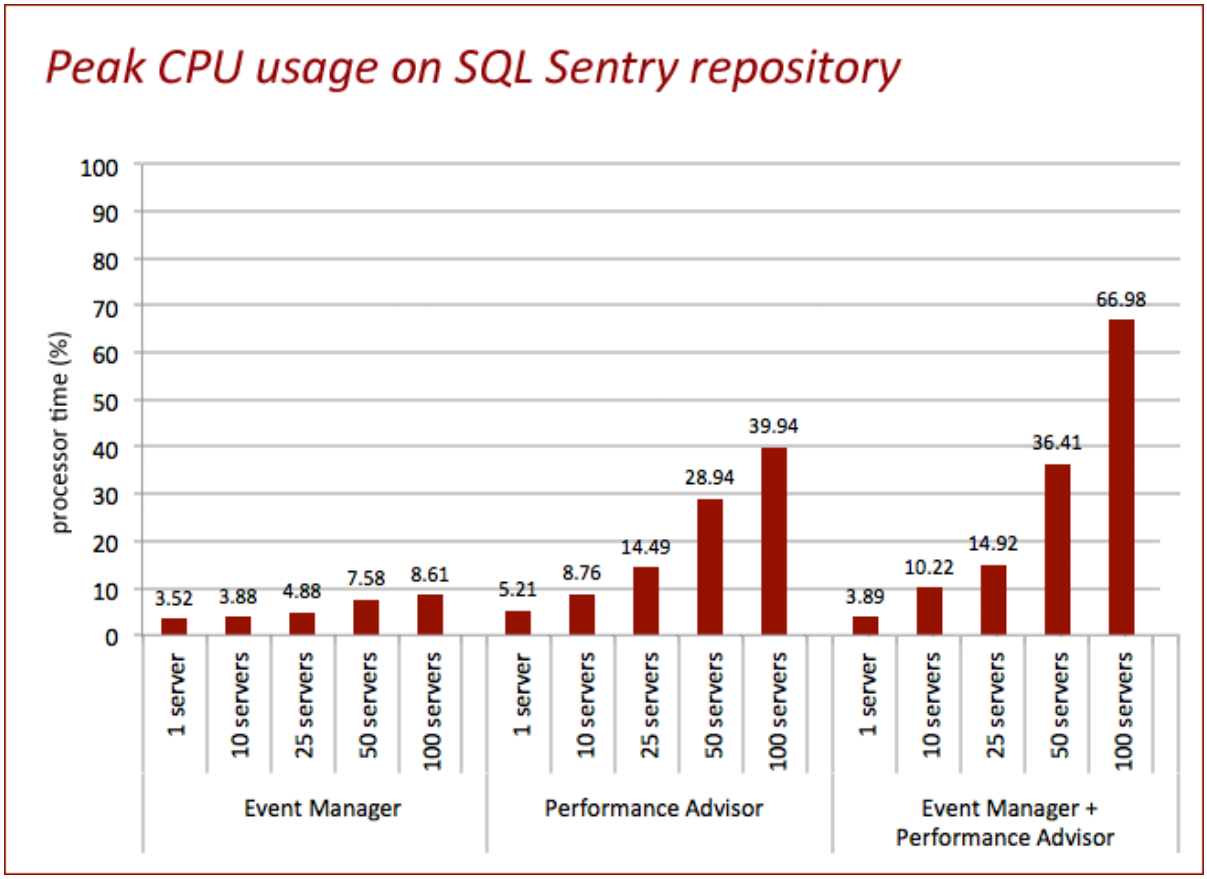
Average CPU

As with many other metrics, the CPU requirements on the SQL Sentry repository server are fairly predictable with the number of servers being monitored. When both Event Manager and Performance Advisor are watching 100 servers, the average CPU utilization of the server hosting the repository server is around 16%. When watching one server, much like the impact on the SQL Sentry server service, the CPU effect is barely noticeable. The following graph shows the average CPU on this 2-CPU database server throughout a 1-hour load test:

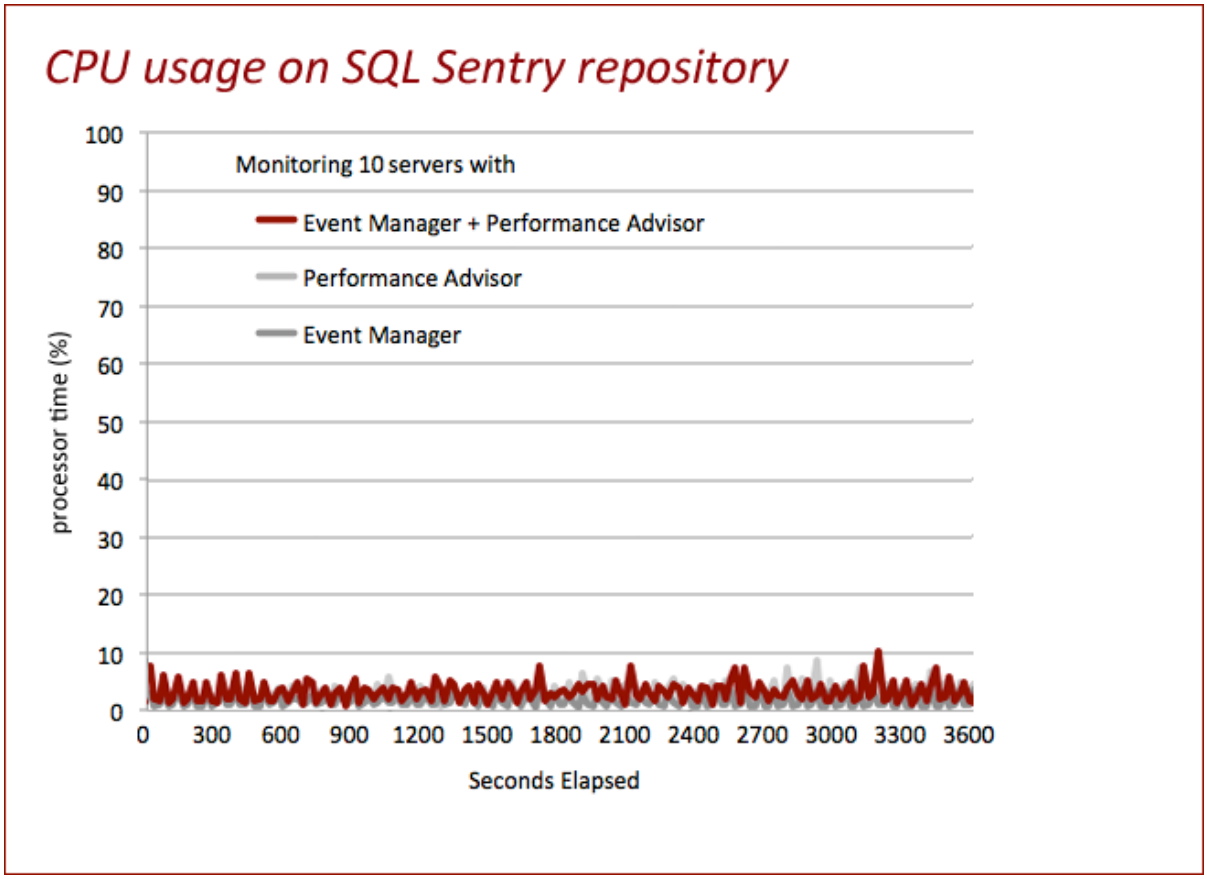


Peak CPU

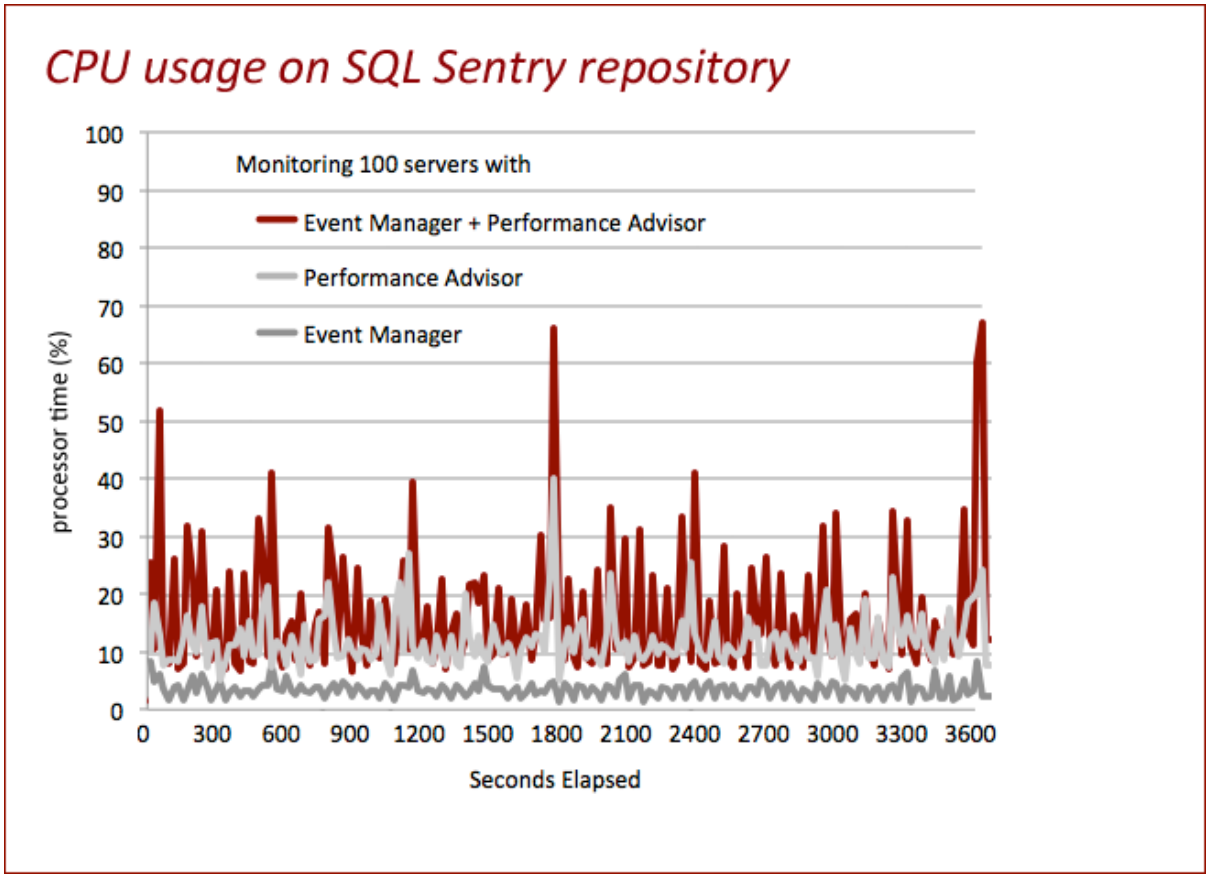
Plotting peak CPU, we see a similar trend – highest CPU usage corresponds to the highest number of monitored servers, and when both products are being used. In this case, the highest observed CPU usage was about 67%. As a reminder, this is on a 2-CPU, virtualized SQL Server, so you can interpolate that on a higher-powered physical server, the impact will be far smaller.



The following two graphs show the time series of CPU usage on the repository machine while monitoring 10 servers and 100 servers. The spikes (most noticeable on the second graph) represent activity relating to the periodic rollup procedure, which aggregates data at intervals of 2, 10, 30, 240 and 1,440 minutes, as well as after 2 days and 3 days. Note that each rollup includes each of the lower rollups – for example, the 10-minute rollup also includes a 2-minute rollup, the 30-minute rollup includes a 2-minute and a 10-minute rollup, and so on. You should expect higher resource utilization as you observe the more inclusive rollups, capping out at the daily rollups, which run at midnight UTC.

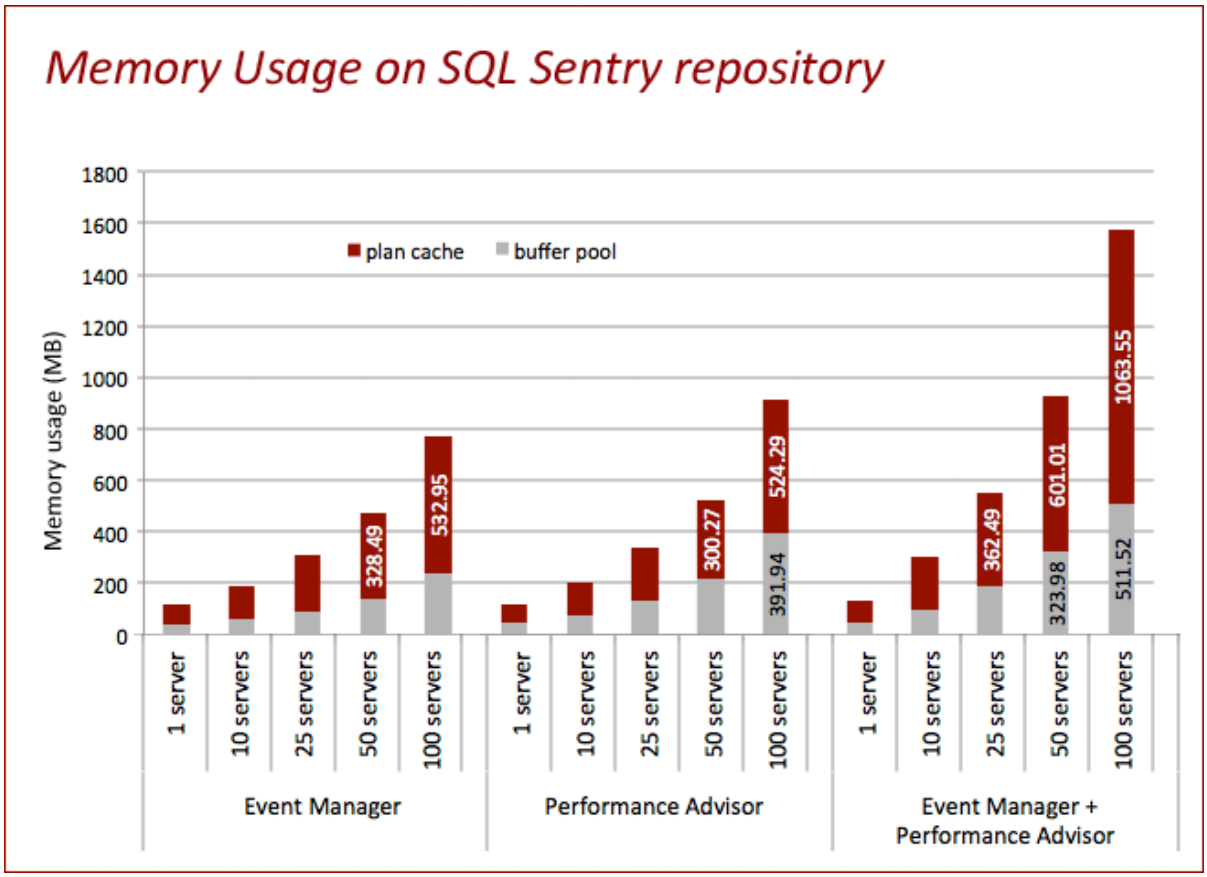


You can see that through an hour of testing, the highest peaks are quite isolated (these represent the 30-minute rollups, though this is overall CPU on the box, so it is not necessarily true that the rollup processes are the only contributor). For more information on observed resource utilization by these rollup procedures in a production environment, see Appendix B, "Production Rollup Observations."



Memory (plan cache and buffer pool)

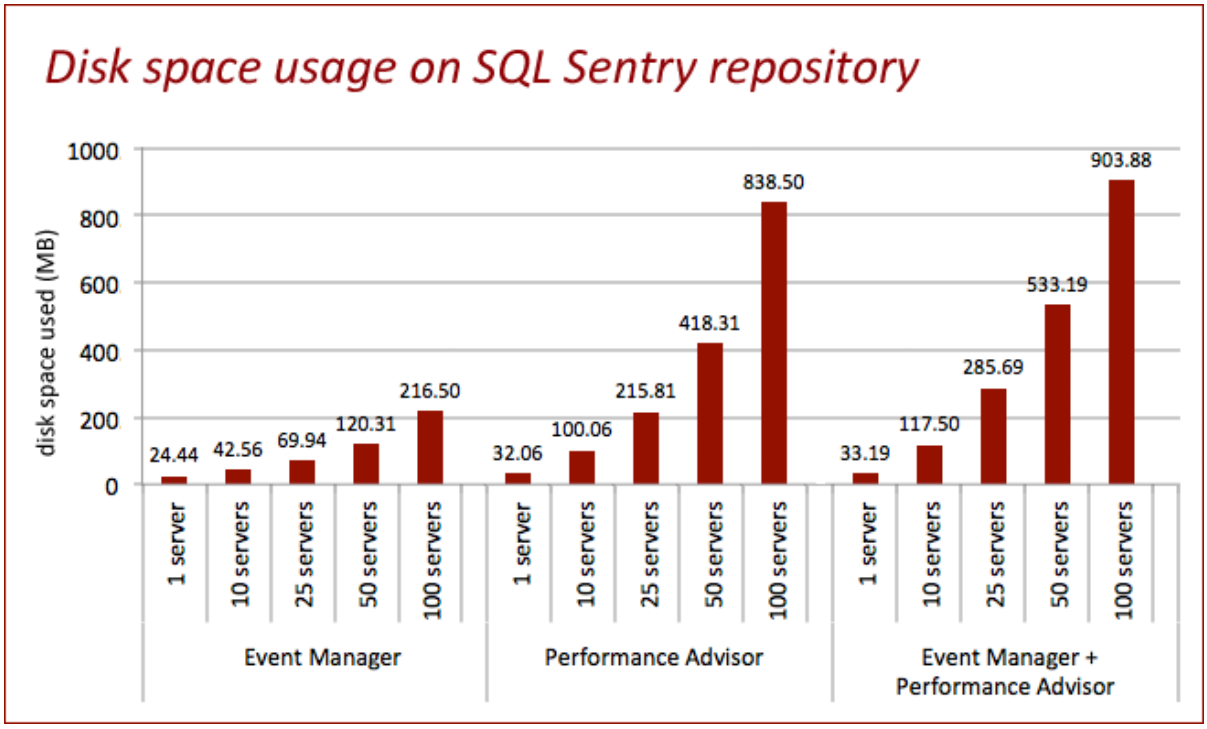
Memory usage on the repository, as with several other resources, grows relatively linearly with the number of instances being monitored. We measured plan cache and buffer pool usage throughout our workload tests, and found that the total memory used by the repository would range from 120 MB (Event Manager monitoring one server) to under 1.6 GB (both products monitoring 100 servers). Plan cache represented anywhere from 57% of the total (Performance Advisor alone) to 70% of the total (Event Manager alone). When both products were in use, plan cache hovered in the 65% range of the combined total.



It is important to note that this memory usage reflects the fact that the repository had not yet hit its peak size, because we re-initialized the database for each hour-long test. There was also minimal console usage during this test in terms of actively pulling ad hoc monitoring data and reports. In short, the results above should not be used as a guideline for long-term memory requirements. These will be driven much more directly by the amount of data you collect from each instance and how the console is being used. Longer-term observations have shown that for environments monitoring 50-100 servers with both Event Manager and Performance Advisor, and seeing regular console and reporting usage, 16GB of available memory on the repository server is typically sufficient to avoid memory pressure and maintain good console/reporting performance.

Disk Space

We measured the space required to house repository data for Event Manager, Performance Advisor, and both services combined, for our defined sets of monitored instances: 1, 10, 25, 50, and 100. As shown in the following graph, the growth is predictably linear with the number of instances being monitored. For this particular set of workloads, which ran for 60 minutes, full monitoring required about 9MB, per server, per hour:



Disk space usage within the SQL Sentry repository is driven by a number of factors, including number of instances, number of databases, number of database files, the workloads, and the frequency of events that trigger extended data collection (e.g. Top SQL or deadlock events). Due to the way our pruning and rollups work, you can expect the largest growth to occur within the first week of monitoring a server, flattening out roughly after the first month. Based on a healthy sample of observations, you can expect long-term requirements to average less than 1GB per monitored instance for Event Manager, and 1GB per monitored instance for Performance Advisor.

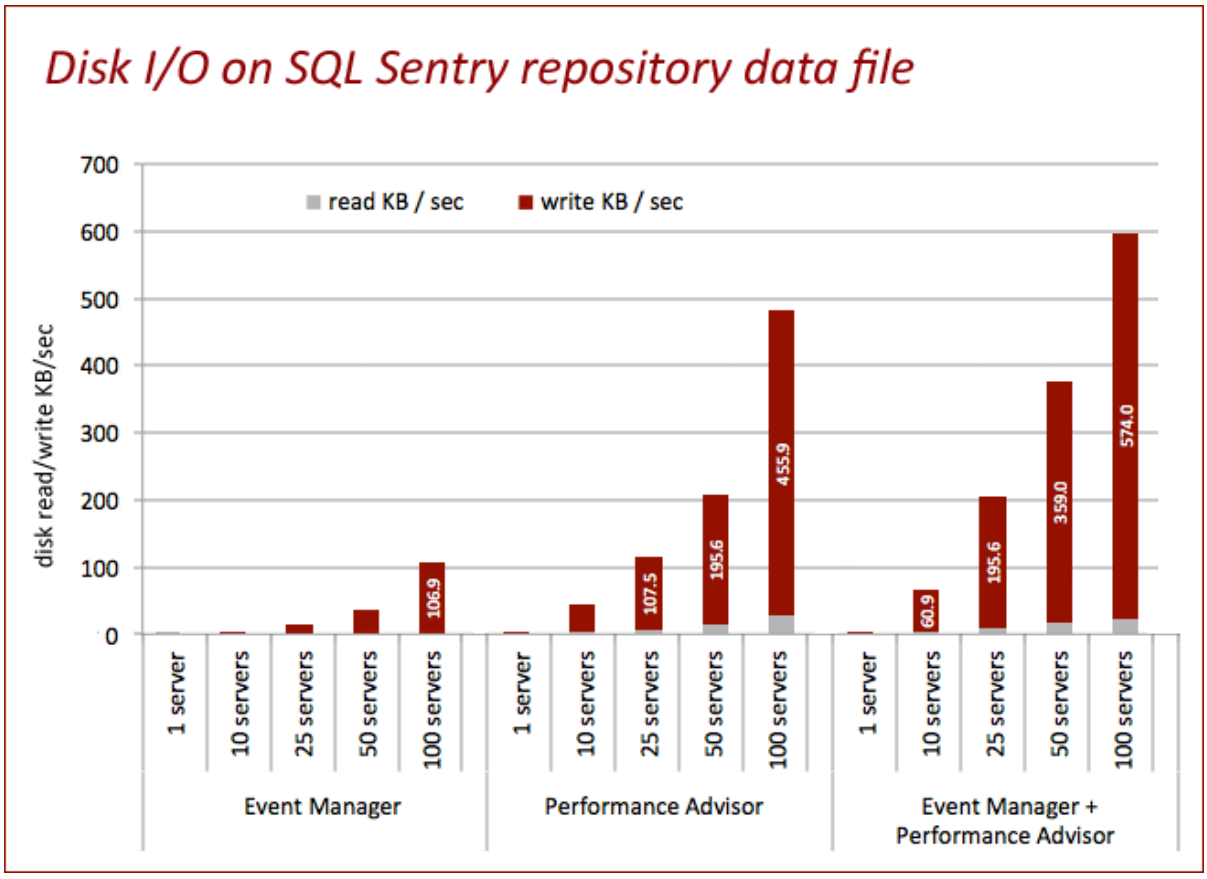
You can read more about capacity planning and what factors drive storage requirements in the following forum post, which also describes how to control collection and rollup intervals to best suit your environment:

http://bit.ly/SQLSentry-Capacity

In a future study, we will explore how data growth is most pronounced when you first start monitoring a server, and then levels out to a roughly flat line. This is because the rollup and pruning processes aggregate and expire older data to the point that they keep pace with new data.

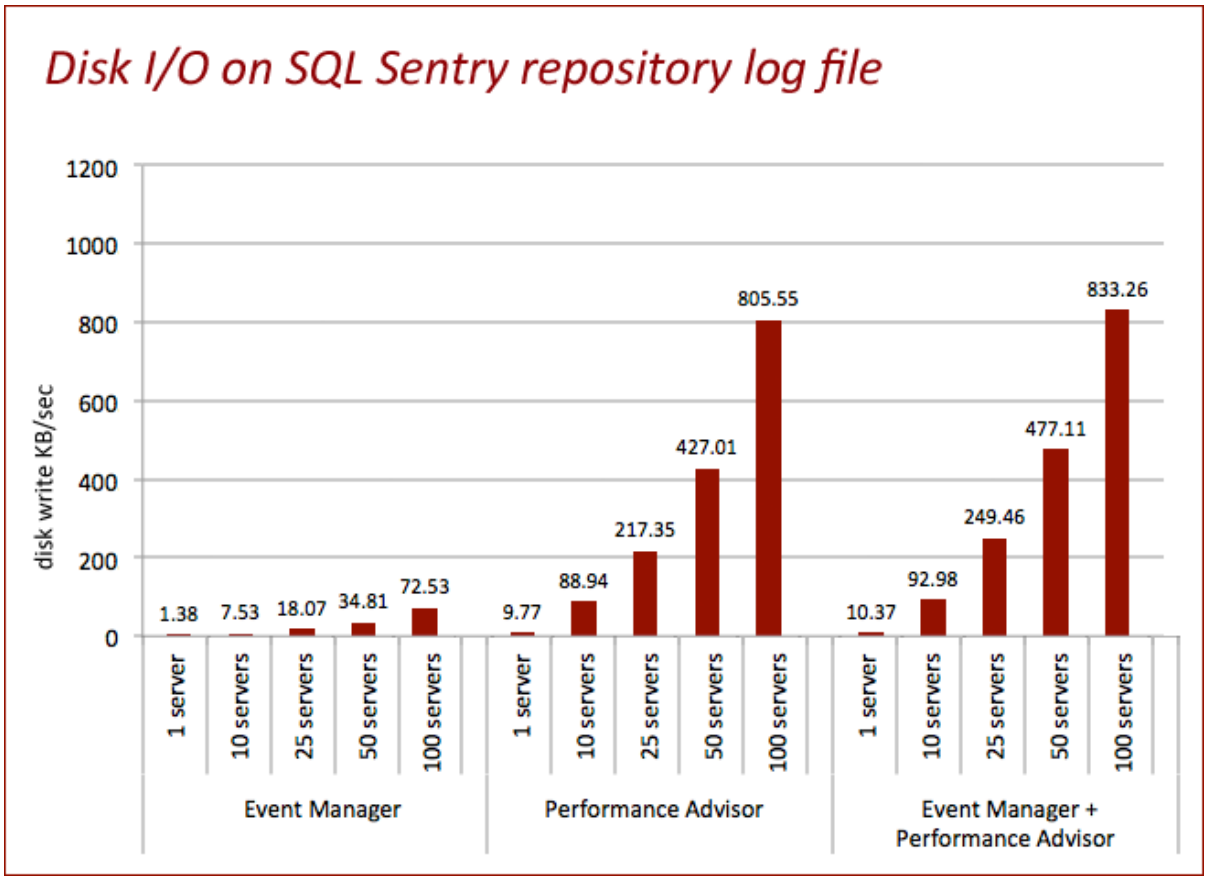
I/O (physical disk read bytes/sec, physical disk write bytes/sec)

As you might expect, when storing a lot of performance data about a large number of servers, you are going to incur some physical I/O on the repository SQL Server. If you plan to monitor 100 servers with both Event Manager and Performance Advisor, you should anticipate approximately 600 KB/sec, almost exclusively in write activity. This will vary slightly depending on factors such as your Top SQL thresholds and how much QuickTrace data you collect.



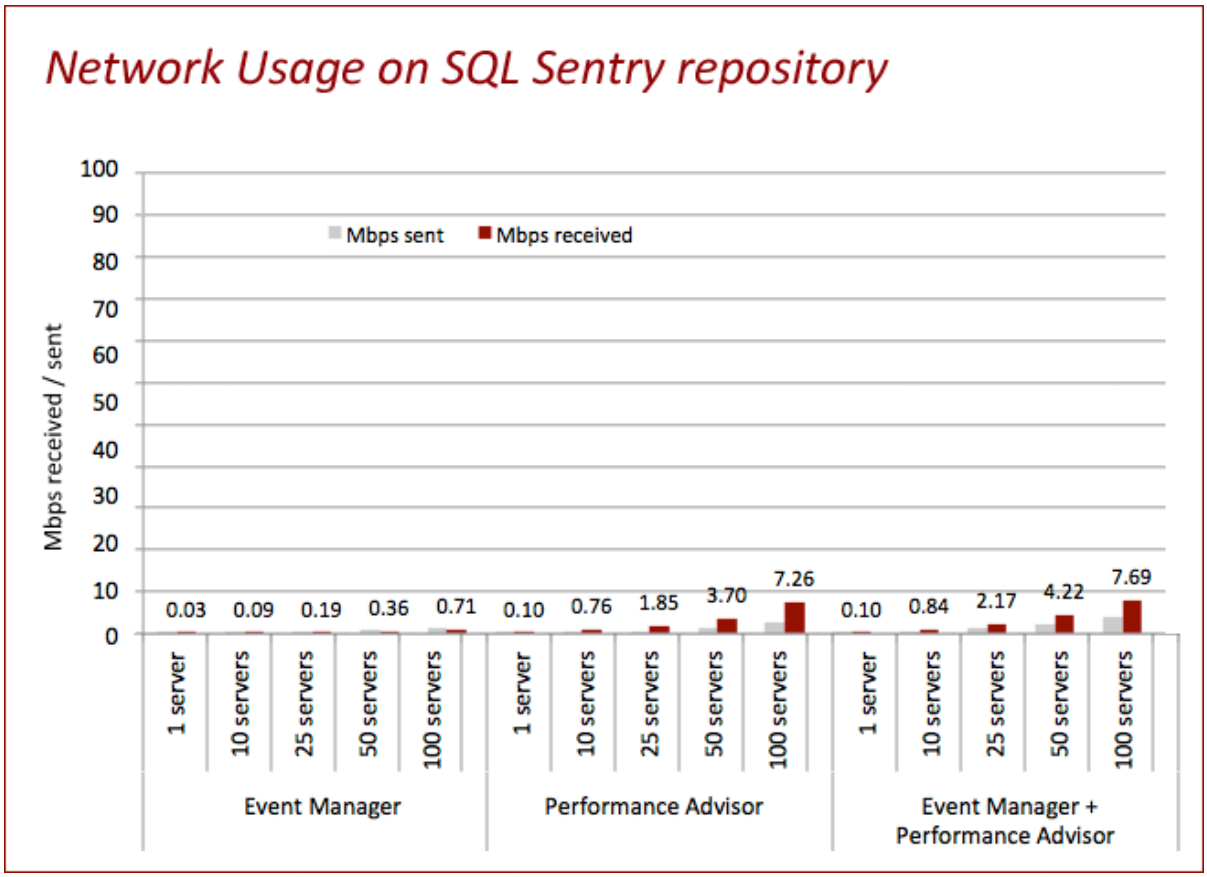
The log activity on the repository is subject to the volume of data coming in, and can often account for more write activity than the data itself. Still, at the higher end, the combination of data and log activity is well under 1.5 MB per second, which seems reasonable given the amount and quality of data being stored. For best results, as with any SQL Server implementation, you should consider using separate disks and spindles for data and log files, which will help spread this load out.

We did measure read activity against the log file, but we are not reporting it because we were not performing any operations on the log drive that would cause any measurable load during the test (in fact, the counters were 0 throughout all tests). If you are using log shipping, mirroring, or other log-reading activities on the same drive as the repository log files – not necessarily against the repository database itself – you will want to take this activity into account when assessing overall I/O.



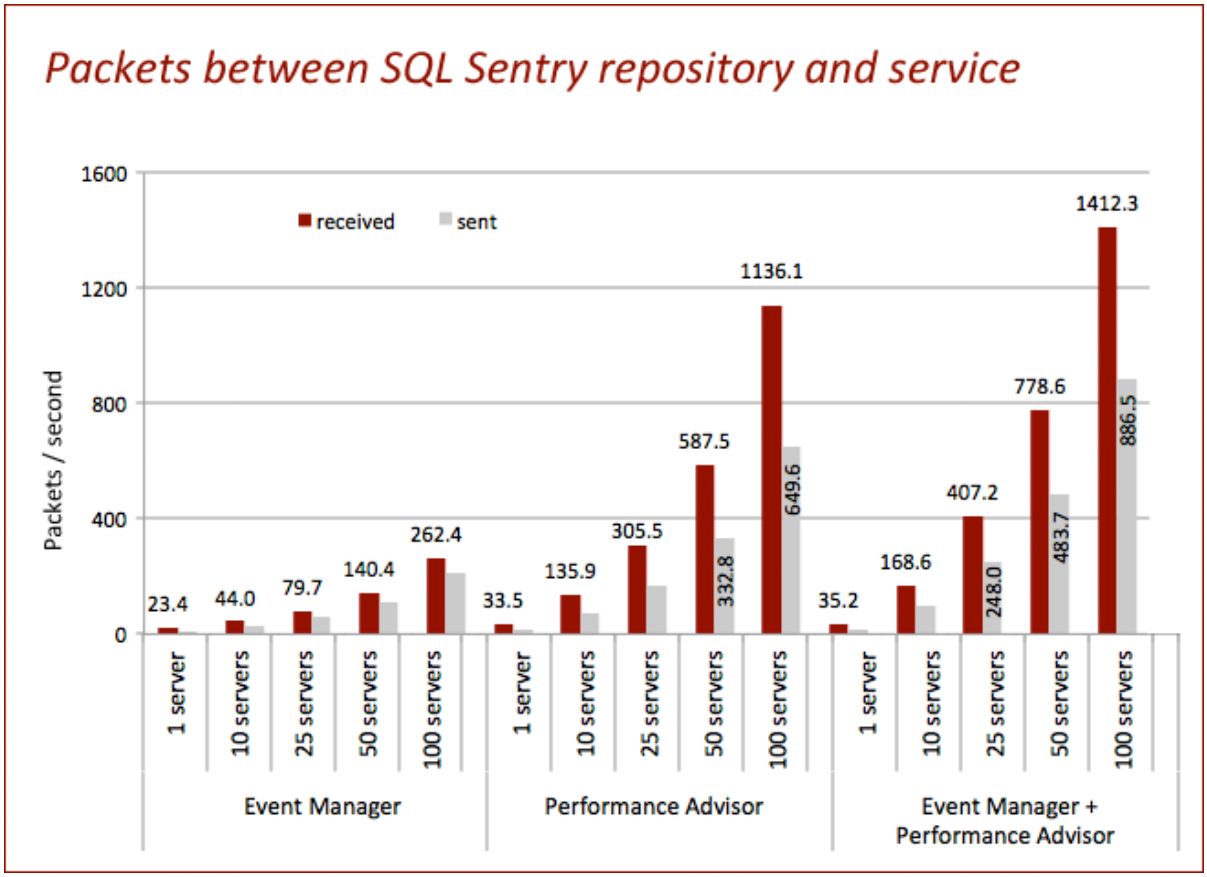
Network (Mbps)

As mentioned in the SQL Sentry server service portion of this paper, one of the primary functions of the solution is to move data from the server service to the repository. As such, you would expect to see some network traffic coming in and out of the repository machine. For this set of tests, we observed negligible network overhead at the low end (0.03 Mbps), peaking at 7.69 Mbps when monitoring 100 servers with both Event Manager and Performance Advisor. As with the network overhead for the monitored target, on a decent circuit, this should be considered quite tolerable – especially given the value of the data being sent.



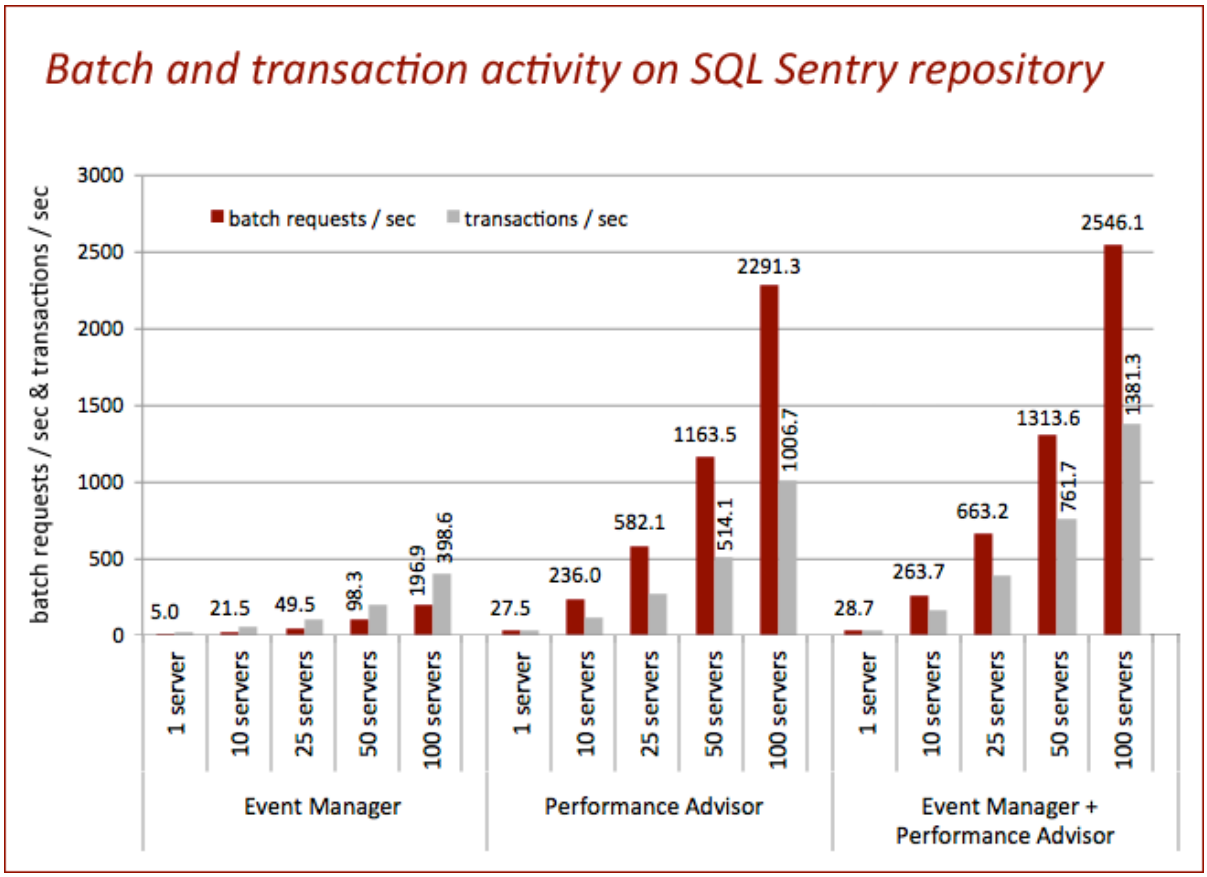
Network (packets / second)

The packets transmitted here represent the traffic that goes between the repository and the server service. As such, it is a mirror image of the packets / second graph shown in the server service section above. The number of packets received by the repository is not quite double the number of packets sent back to the service, and this should be expected as the service writes a lot more data than it reads. In our tests this capped out at about 1,400 packets per second when monitoring 100 servers with both products.



SQL Server activity (batch requests and transactions per second)

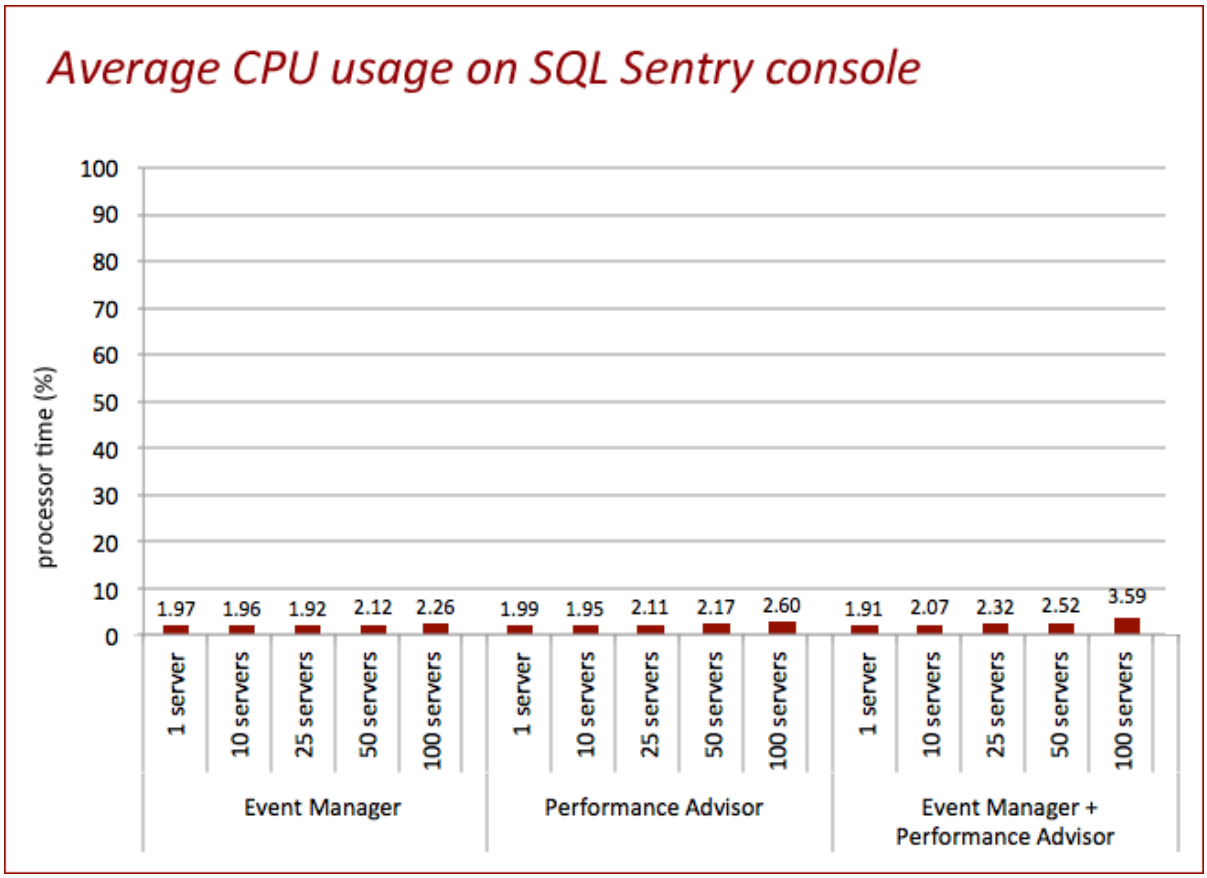
As with the monitored target, we measured average batch requests per second and transactions per second on the repository, to see how much activity was actually being generated from the monitoring solution. As expected, the server service and console processes will generate far more activity against the repository than the server service alone will generate against the monitored targets (since the latter is highly optimized for least impact). When monitoring 100 servers with both Event Manager and Performance Advisor, an average of 2,546 batch requests and 1,381 transactions per second were registered against the repository. This ratio demonstrates a concerted streamlining effort in Performance Advisor – by combining many individual statements together, larger batches with more bundled statements help enhance throughput, and as a result of fewer individual transactions, a lower number of log flushes leads to reduced overall I/O.



SQL Sentry Console

Average CPU

The console requires a very modest amount of CPU – below 4%, on average, even when watching 100 servers with both Event Manager and Performance Advisor. Remember that this is a virtual machine with a single CPU. There may be periodic spikes, depending on how you are interacting with the console – in these tests, we observed up to 13% at the high end. Typically the console is installed on a workstation or dedicated machine hooked up to a wall monitor, and you can install as many consoles as you wish; in most cases, CPU contention should not be an issue.



Memory (working set)

With 100 servers registered, the console requires just under 190 MB of RAM to operate with the Event Manager Calendar, Performance Advisor Global View, and a single Performance Advisor Dashboard open. The number of servers being actively monitored has very little impact on the memory usage of the console; much more important is how many tabs you have open. In our observations, you can expect an additional 10 MB of memory utilization for each additional active Dashboard, with variances for certain types of interactions that require additional resources. The SQL Sentry console is a .NET application, and therefore benefits from the framework's inherent garbage collection and memory cleanup – so the memory usage will reflect what you are actively working with as opposed to everything you've done since starting the console. For more information about how .NET handles memory cleanup, see the following article:

http://msdn.microsoft.com/en-us/library/f144e03t.aspx

Summary

We have worked quite hard to measure our monitoring tools' impact on key performance indicators in a realistic environment. The results are a testament to the level of effort that has gone into the development of our monitoring products, leading up to the v6 release. We are quite pleased with the outcome and are happy to share this data with you.

For the target SQL Server instance, these are the observations we have made regarding additional resources required to support monitoring with SQL Sentry:

Monitored Instance

Monitored Target		Event Manager	Performance Advisor	Event Manager + Performance Advisor
Additional CPU %	average	0.23	0.39	0.51
	peak	0.12	1.5	2.19
Additional buffer pool (MB)		0.01	6.26	11.26
Additional plan cache (MB)		0.37	4.27	15.75
Additional network	Mbps out	0.01	0.04	0.05
	Mbps in	0.01	0.08	0.09
	packets/sec out	1.50	8.80	11.70
	packets/sec in	1.30	8.00	11.30
Additional batch requests / sec		0.41	0.98	1.29
Additional transactions / sec		0.41	1.58	3.91
Additional disk I/O (kb/sec)	read	0.60	0.83	1.76
	write	0.73	3.62	4.00

And for the individual SQL Sentry components, these are our observations:

SQL Sentry Server Service

Server Service		Event Manager					Performance Advisor					Event Manager + Performance Advisor				
	Server count	1	10	25	50	100	1	10	25	50	100	1	10	25	50	100
CPU %	average	0.23	0.48	0.99	1.86	3.77	0.31	1.28	2.96	5.87	12.39	0.34	1.65	4.09	8.06	16.03
	peak	0.71	1.04	2.71	6.46	38.31	1.34	3.53	9.27	35.78	53.32	0.83	5.17	23.98	37.60	61.54
Context switches/sec		161	285	443	728	1,230	220	690	1,432	2,380	4,241	224	788	1,703	2,993	5,208
Working Set (MB)		98	221	257	329	538	163	430	622	771	991	179	487	659	896	1,118
Soft page faults/sec		1	7	7	11	29	5	19	29	57	160	6	19	48	72	276
Network I/O	to repository	0.03	0.09	0.19	0.36	0.71	0.1	0.76	1.85	3.7	7.26	0.1	0.76	1.85	3.7	7.26
(Mbps)	from repository	0.03	0.14	0.34	0.7	1.42	0.05	0.28	0.66	1.3	2.55	0.06	0.45	1.21	2.37	4.15
	to target	0.47	0.63	0.87	1.28	2.1	0.52	1.17	2.23	3.98	7.54	0.54	1.21	2.37	4.28	8.3
	from target	0.59	0.70	0.87	1.16	1.75	0.61	0.89	1.34	2.12	3.68	0.61	0.90	1.41	2.29	4.04
	total	1.12	1.56	2.27	3.50	5.97	1.28	3.10	6.08	11.10	21.02	1.31	3.32	6.84	12.63	23.76
Packets/sec	from targets	335.4	374.1	432.2	527.1	724.7	344.2	454.6	628.5	925.8	1,527.9	347.1	463.1	659.6	1,014.4	1,745.3
	to targets	346.9	390.0	455.5	562.5	783.3	354.9	459.5	623.2	905.7	1,474.8	358.2	468.4	654.3	998.1	1,696.1
	from repository	8.9	26.5	57.2	109.3	214.5	14.7	72.8	169.0	332.8	649.6	15.9	99.4	248.0	483.7	886.5
	to repository	23.4	44.0	79.7	140.4	262.4	33.5	135.9	305.5	587.5	1,136.1	35.2	168.6	407.2	778.6	1,412.3

SQL Sentry Repository

Repository		Event Manager					Performance Advisor					Event Manager + Performance Advisor				
	Server count	1	10	25	50	100	1	10	25	50	100	1	10	25	50	100
CPU %	average	1.54	1.74	2.1	2.87	3.81	1.73	2.95	4.85	7.22	12.2	1.72	3.33	6.29	11.25	16.31
	peak	3.52	3.88	4.88	7.58	8.61	5.21	8.76	14.49	28.94	39.94	3.89	10.22	14.92	36.41	66.98
Buffer Pool (MB)		40.5	60.4	89.2	142.4	240.3	46.1	77.2	131.2	220.9	391.9	46.7	94.2	189.0	324.0	511.5
Plan Cache (MB)		81.1	126.0	217.4	328.5	533.0	75.4	126.0	208.8	300.3	524.3	83.8	206.1	362.5	601.0	1,063.6
Disk Space (MB)		24.4	42.6	69.9	120.3	216.5	32.1	100.1	215.8	418.3	838.5	33.2	117.5	285.7	533.2	903.9
Physical I/O	write (data file)	0.1	2.5	12.8	35.6	106.9	3.5	40.5	107.5	195.6	455.9	3.7	60.9	195.6	359.0	574.0
(kb/sec)	read (data file)	0.0	0.3	0.3	0.8	0.6	0.4	2.7	6.8	13.3	23.9	0.6	4.3	9.6	18.2	26.5
	write (log file)	1.4	7.5	18.1	34.8	72.5	9.8	88.9	217.4	427.0	805.6	10.4	93.0	249.5	477.1	833.3
Network I/O	in	0.03	0.09	0.19	0.36	0.71	0.10	0.76	1.85	3.70	7.26	0.10	0.76	1.85	3.70	7.26
(Mbps)	out	0.03	0.14	0.34	0.70	1.42	0.05	0.28	0.66	1.30	2.55	0.06	0.45	1.21	2.37	4.15
Packets/sec	in	23.4	44.0	79.7	140.4	262.4	33.5	135.9	305.5	587.5	1,136.1	35.2	168.6	407.2	778.6	1,412.3
	out	8.9	26.5	57.2	109.3	214.5	14.7	72.8	169.0	332.8	649.6	15.9	99.4	248.0	483.7	886.5
SQL activity	batch requests	5.0	21.5	49.5	98.3	196.9	27.5	236.0	582.1	1,163.5	2,291.3	28.7	263.7	663.2	1,313.6	2,546.1
(/sec)	transactions	20.1	53.4	108.7	205.1	398.6	28.8	117.9	265.5	514.1	1,006.7	30.6	161.9	395.3	761.7	1,381.3

SQL Sentry Console

Console		Event Manager					Performance Advisor					Event Manager + Performance Advisor				
	Server count	1	10	25	50	100	1	10	25	50	100	1	10	25	50	100
CPU %	average	1.97	1.96	1.92	2.11	2.26	1.99	1.95	2.10	2.17	2.60	1.91	2.07	2.32	2.52	3.59
	peak	7.91	7.97	7.22	7.91	8.60	7.12	7.13	7.28	8.54	10.17	6.97	7.67	7.96	9.17	12.97
Working Set (MB)		175.7	176.0	177.8	179.3	185.4	175.7	176.7	177.5	180.0	185.8	175.9	176.9	179.1	181.9	188.6

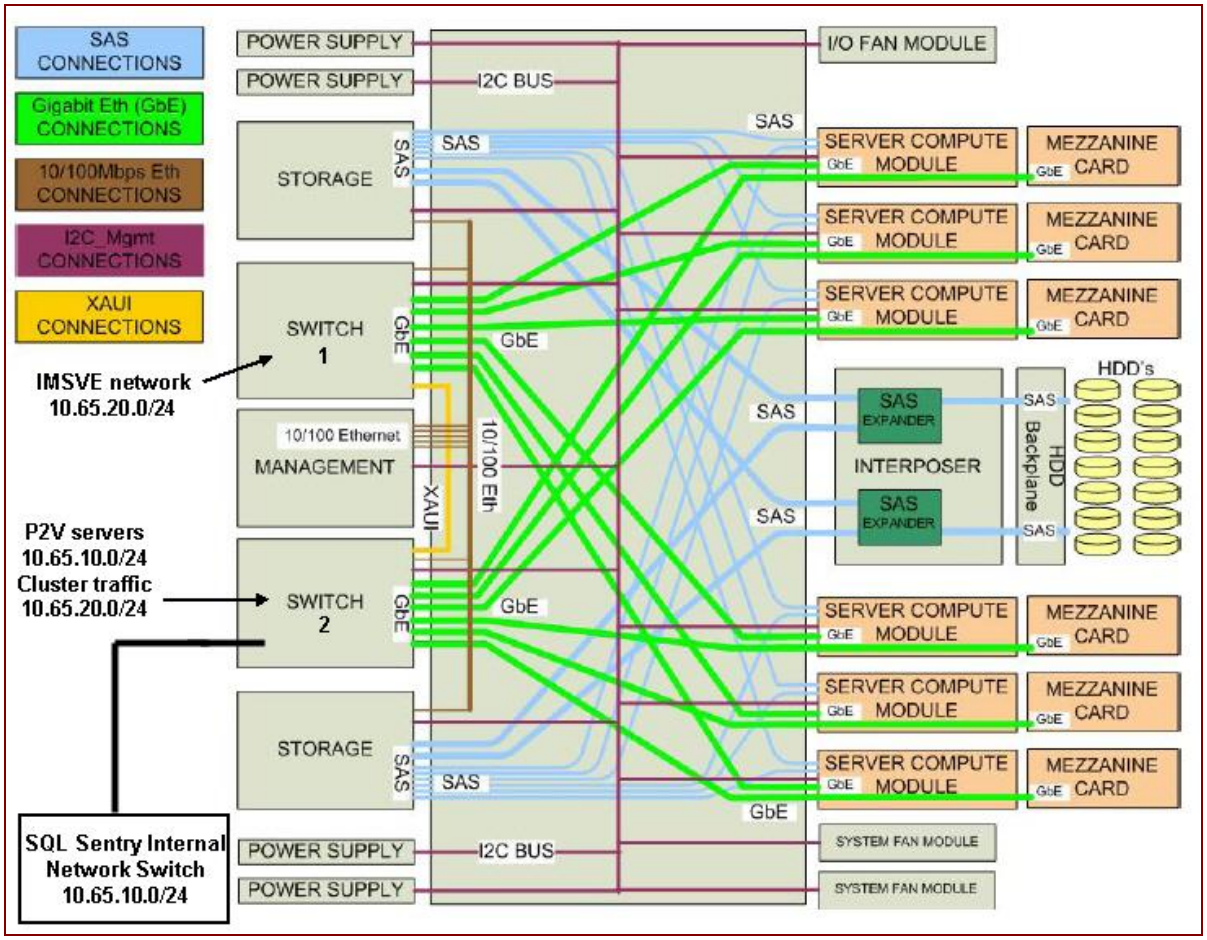
As previously mentioned, we are quite happy with these results. However, as we are always striving to do better, we have certainly identified a couple of areas to focus on moving forward. In the meantime, we sincerely hope this paper helps you understand the overhead our software solution will place both on your production servers as well as the computers used to assist in the monitoring process.

For more information on this study, please contact Aaron Bertrand at abertrand@sqlsentry.net.

Appendix A – Test Environment Details

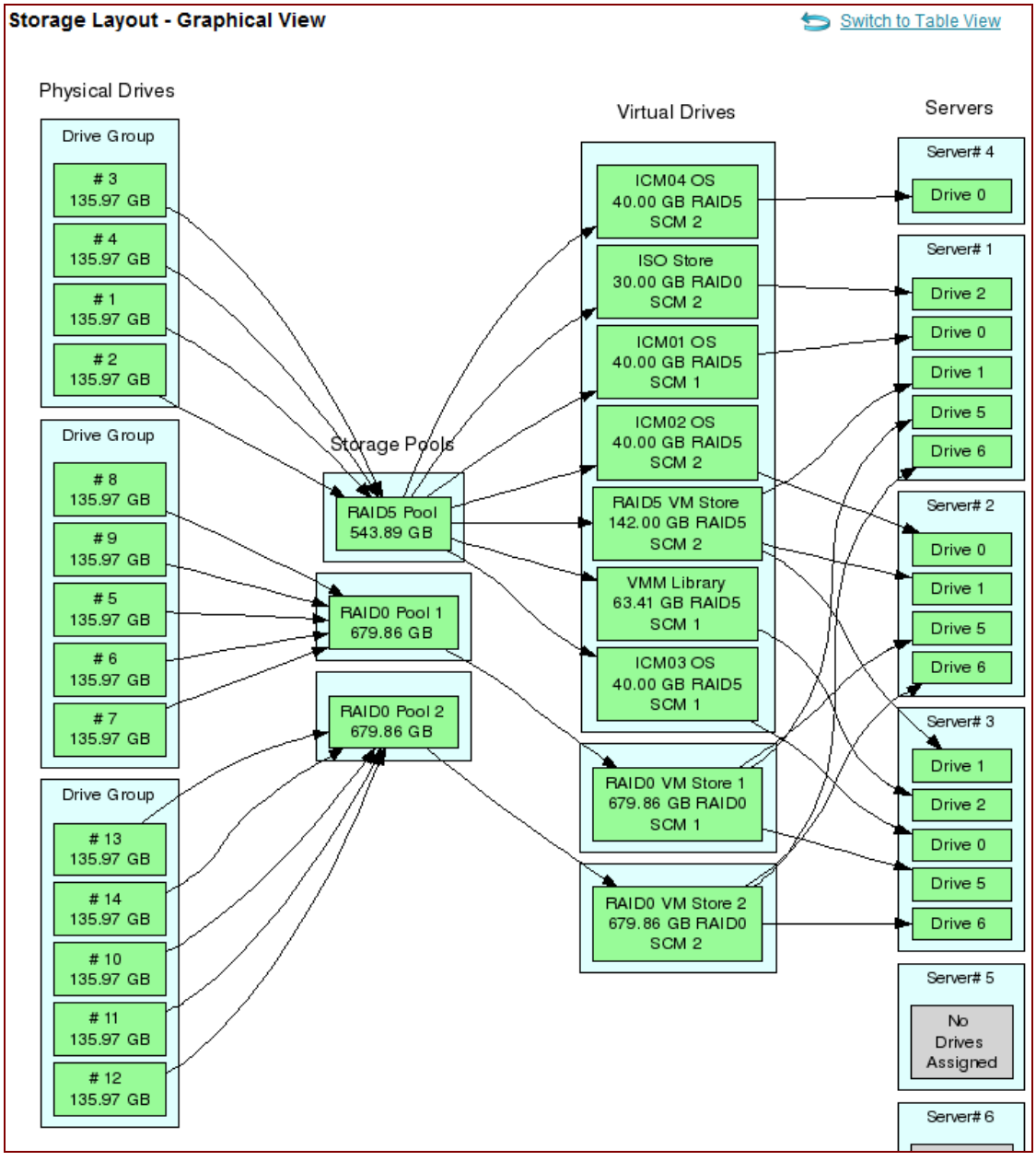
Network Details

The diagram below shows the Intel Modular Server (IMS) architecture, overlaid with some of the test environment network details. Switch 1 was dedicated to the overhead test environment, and was isolated from the virtual environment cluster traffic as well as internal traffic for the SQL Sentry network, which also makes use of the IMS.



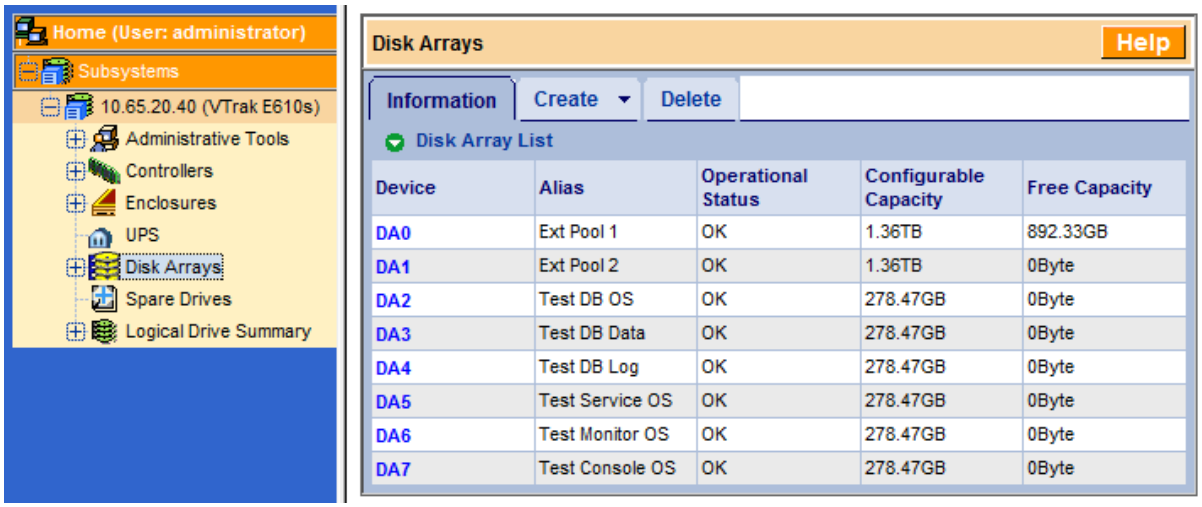
Storage

The diagram below shows the storage layout for the test environment. The two RAID0 pools were dedicated to two VM stores holding the VHDs for the 100 test target VMs, and all host OS drives shared the same RAID5 pool. The two VM Stores are mapped to servers 1, 2 and 3 since these are CSVs (Clustered Shared Volumes), which allow Hyper-V 2008 R2 to seamlessly move test VMs between hosts as needed.



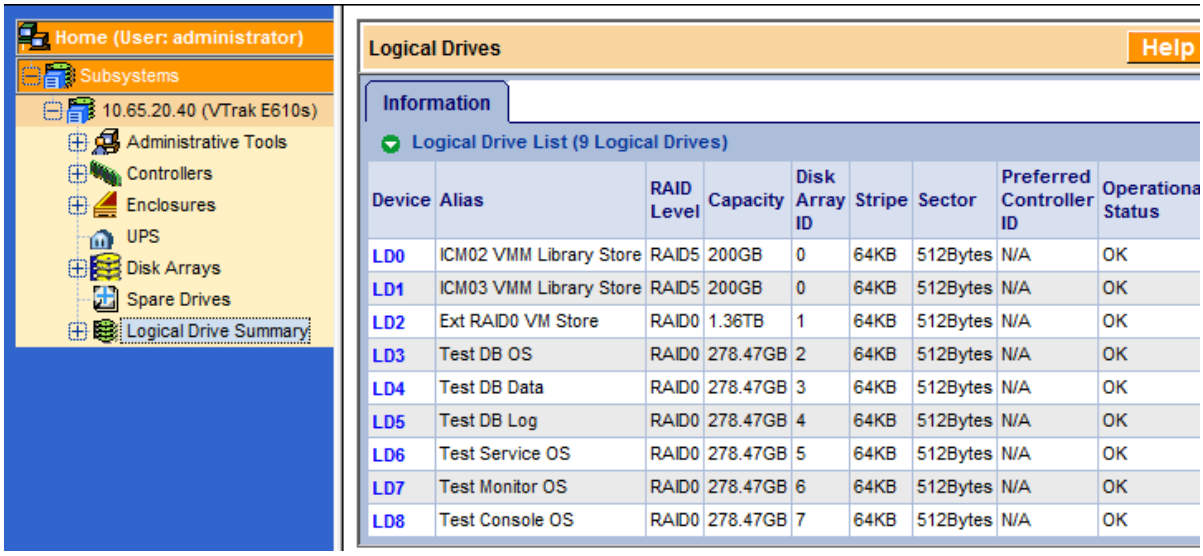
In the diagram above, the monitoring host, server #4 (ICM04) only shows a single drive, for the host OS. This is because all of the drives used for the SQL Sentry components were dedicated on the external VTrak array, as shown below by DA2-DA7 and LD3-LD8:

Physical Disks:



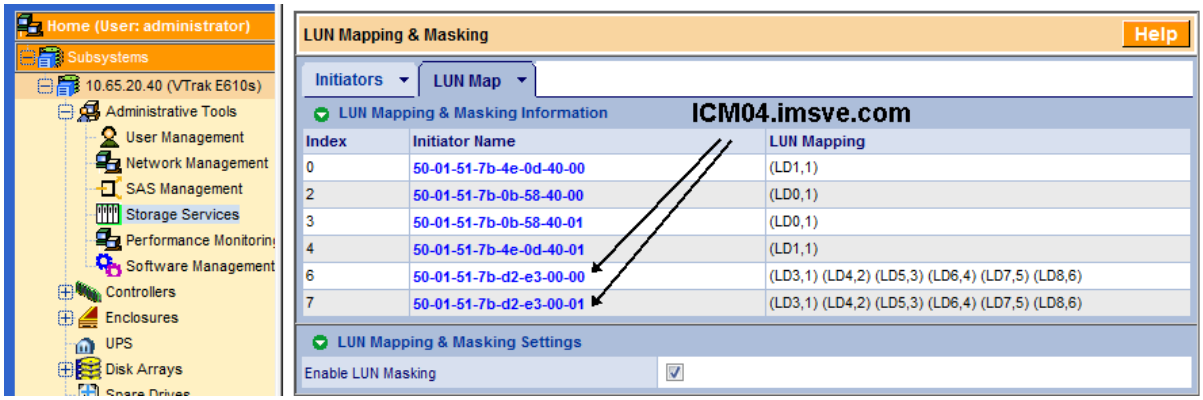
Device	Alias	Operational Status	Configurable Capacity	Free Capacity
DA0	Ext Pool 1	OK	1.36TB	892.33GB
DA1	Ext Pool 2	OK	1.36TB	0Byte
DA2	Test DB OS	OK	278.47GB	0Byte
DA3	Test DB Data	OK	278.47GB	0Byte
DA4	Test DB Log	OK	278.47GB	0Byte
DA5	Test Service OS	OK	278.47GB	0Byte
DA6	Test Monitor OS	OK	278.47GB	0Byte
DA7	Test Console OS	OK	278.47GB	0Byte

Logical Drives:



Device	Alias	RAID Level	Capacity	Disk Array ID	Stripe	Sector	Preferred Controller ID	Operational Status
LD0	ICM02 VMM Library Store	RAID5	200GB	0	64KB	512Bytes	N/A	OK
LD1	ICM03 VMM Library Store	RAID5	200GB	0	64KB	512Bytes	N/A	OK
LD2	Ext RAID0 VM Store	RAID0	1.36TB	1	64KB	512Bytes	N/A	OK
LD3	Test DB OS	RAID0	278.47GB	2	64KB	512Bytes	N/A	OK
LD4	Test DB Data	RAID0	278.47GB	3	64KB	512Bytes	N/A	OK
LD5	Test DB Log	RAID0	278.47GB	4	64KB	512Bytes	N/A	OK
LD6	Test Service OS	RAID0	278.47GB	5	64KB	512Bytes	N/A	OK
LD7	Test Monitor OS	RAID0	278.47GB	6	64KB	512Bytes	N/A	OK
LD8	Test Console OS	RAID0	278.47GB	7	64KB	512Bytes	N/A	OK

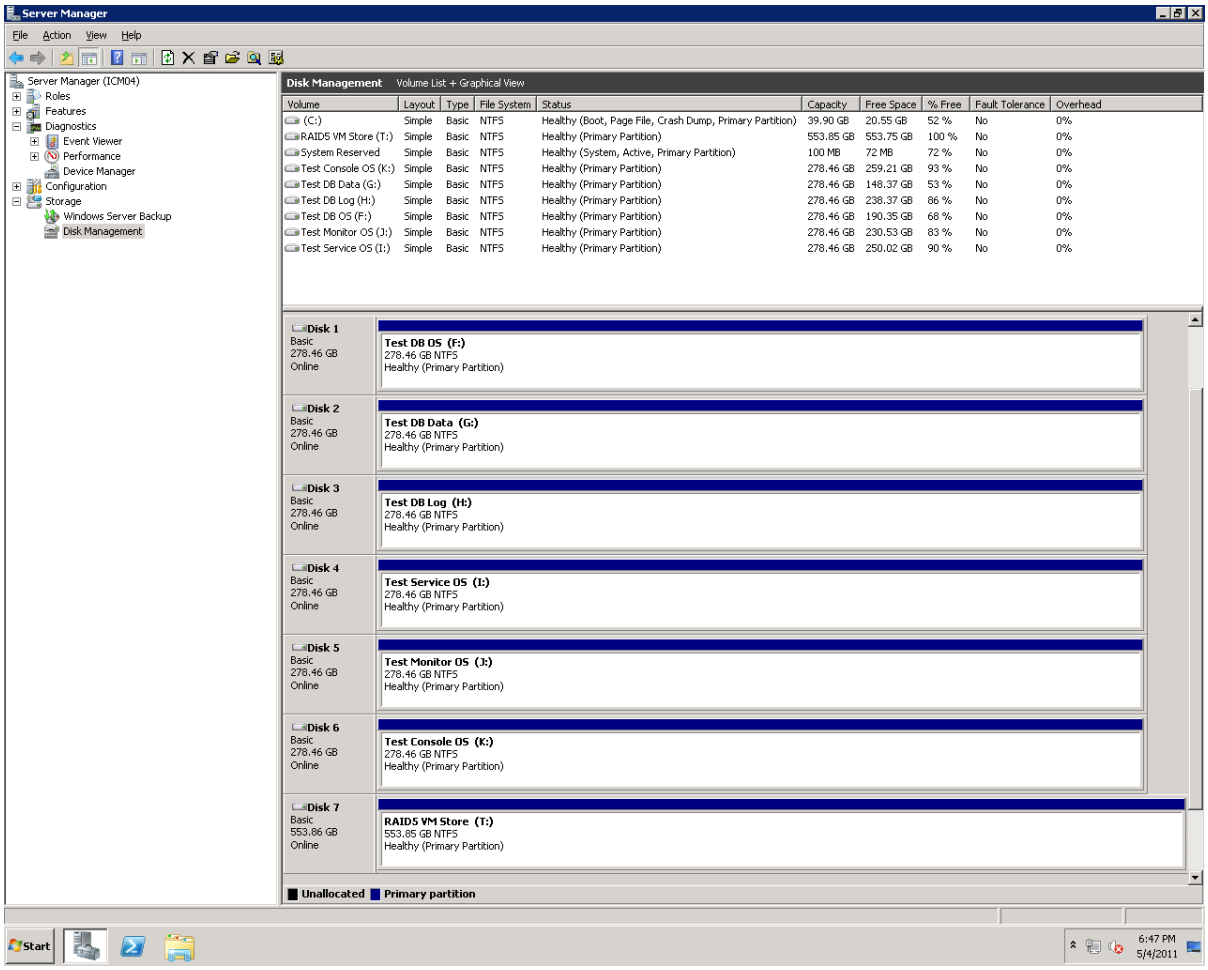
LUN Mapping of Logical Drives to ICM04:



The two initiator entries above represent the 2 redundant SCM (storage controller modules) on the Intel Modular Server.

Windows Disks

Below is a view of the local storage on ICM04. Again, each disk held a single fixed VHD for the SQL Sentry component VMs, in order to keep the associated I/O as isolated as possible.



Appendix B – Production Rollup Observations

In the production environment of one of our clients, we monitored the rollup process on their repository for a 6-hour period, including at least one rollup at each interval (2, 10 and 30 minutes, and 4 and 24 hours). At the time of testing, they were monitoring 54 servers with Event Manager, and 34 servers with Performance Advisor. The following graphs demonstrate the predictable nature of the rollups as they occur for each pre-defined interval. Highlighted are the heavier rollups – several instances of the 30-minute rollup, which includes 2- and 10-minute rollups, and one instance each for the 4-hour and 24-hour rollup. This server has 4 Intel Xeon E5430 2.66 GHz CPUs and 16GB of RAM.