Performance Characterization of Microsoft SQL Server on VMware vSphere 6

Performance Study

TECHNICAL WHITE PAPER

vmware®

Table of Contents

Executive Summary

Large Microsoft SQL Server® databases can achieve great performance and fully utilize current-generation server and storage technology using VMware vSphere®. With the introduction of vSphere 6, larger physical hosts are supported more than ever before (up to 480 processors and 12 TB).   In this paper, a variety of vCPU and virtual machine combinations were tested on current-generation hardware to show that vSphere 6 can handle hundreds of thousands of online transaction processing (OLTP) database operations per minute.

Introduction

VMware vSphere provides an ideal platform on which customers can virtualize their business-critical applications, including databases, ERP systems, email servers, and emerging technologies such as Hadoop. A full discussion of the benefits is included in the whitepaper "Virtualizing Business-Critical Applications on vSphere [1]".

A business-critical application that is often run on vSphere is Microsoft SQL Server, which is "one of the most widely deployed database platforms in the world, with many organizations having dozens or even hundreds of instances deployed in their environments [1]." Consolidating these deployments onto modern multi-socket, multi-core, multi-threaded server hardware is an effective solution for administrators and their organizations.

Achieving optimal SQL Server performance on vSphere has been a focus for VMware. Last year a series of performance tests with SQL Server on vSphere 5.5 [2] using a four-socket Intel Xeon E7-4800 based server was published. That whitepaper showed excellent performance up to the maximum size virtual machine supported at the time. This new study sizes the VMs similarly and uses the same methodology, but with vSphere 6 and Intel Xeon E7-4800 v2 based processors (codenamed "Ivy Bridge EX"). With vSphere 6, VMware supports much larger "monster" virtual machines that can scale up to 128 virtual CPUs and 4 TB of RAM [3]; however, scaling to these new limits was not possible with the hardware used in this study. There is another study published recently using vSphere 6 and Oracle 12c [4] that scales a database workload from 15 vCPUs to 120 vCPUs.

The new tests show large SQL Server database instances continue to run extremely efficiently, achieving great performance in a variety of virtual machine configurations on vSphere 6. Only minor tunings to the SQL Server application and the vSphere 6 ESXi hypervisor were needed to fully optimize these large virtual machines.

Additionally, this new study includes several tests with CPU affinity to show the performance differences between physical cores and logical processors (Hyper-Threads), the maximum throughput achieved using multiple virtual machines that vary in virtual CPU and memory sizes, and experiments with the advanced Latency Sensitivity setting offered by vSphere.

Test Environment

In order to test the performance of large SQL Server database virtual machines, a benchmark test bed was assembled with sufficient compute, storage, and networking hardware. This infrastructure was tested with many small virtual machines as well as a smaller number of large virtual machines. Multiple, large databases were created within these virtual machines, and the open-source DVD Store workload was used to measure the aggregate throughput.

Test Bed Configuration

The server, storage, and networking hardware used for this performance study are shown in Figure 1. The Dell PowerEdge R920 server was the system under test (SUT), and was equipped with four Intel Xeon E7-4890 v2 15-core processors, 1 TB of physical memory, a quad-port 1GbE NIC, two 10GbE NICs, and a 16Gb Fibre Channel HBA.

The latest system BIOS and iDRAC firmware were used. Since the goal of this study was to achieve maximum application performance, the BIOS System Profile was set to Performance in order to disable processor power

management. Other BIOS settings, such as Intel Hyper-Threading and Turbo Mode, were left at their defaults (enabled) because they provided the highest throughput.

ESXi 6.0 was installed on an internal disk, while all the virtual machines' disks resided on an EMC VNX5800 array. This array had a mixture of 6Gbps 300GB 15K SAS drives and 200GB SSDs. 16 of the SAS drives were combined into a RAID0 LUN, and this is where the guest operating system and log file virtual disks were stored. The SSDs were split into multiple 6-drive RAID0 LUNs, and each SQL database instance was placed on the SSDs to achieve optimal disk read performance.

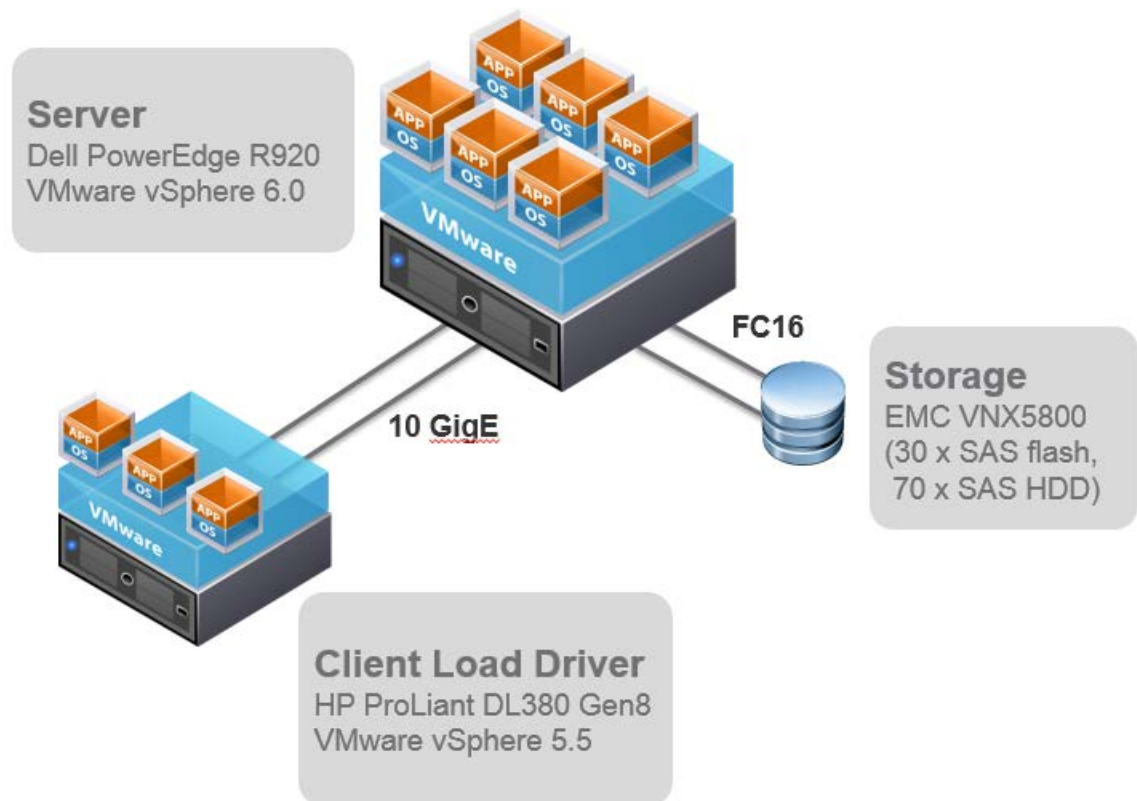Refer to "Appendix A" for more hardware and software configuration details.

Figure 1. Test bed architecture

Host Processor Architecture

Figure 2 shows the physical host's processor sockets, cores, and logical processors available to ESXi. Note that while the host has 4 physical processor packages, there are 120 logical processors available to the hypervisor.
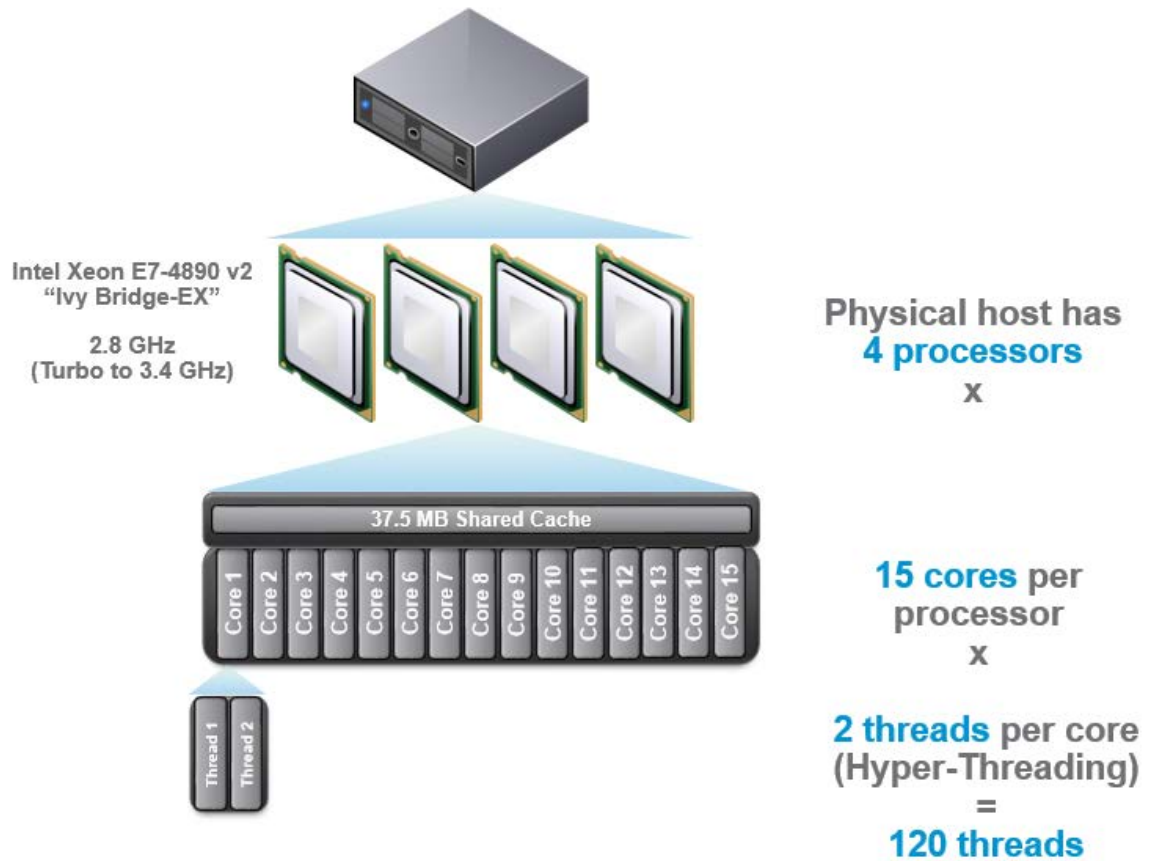
Figure 2. Host processor architecture

Operating System and SQL Server Installation/Tuning

ESXi 6.0 was installed on the Dell PowerEdge R920. Microsoft Windows® Server 2012 R2 was used as the guest operating system for the virtual machines and was updated using Windows Update. The number of virtual CPUs and virtual RAM varied depending upon the test, and is detailed later. The operating system, SQL logs, and backups were stored on a 350GB virtual disk. Each virtual machine was upgraded to virtual hardware version 11, the latest version of VMware Tools installed, and each virtual hard disk was connected to a separate VMware Paravirtual SCSI (PVSCSI) adapter to ensure optimal throughput and lower CPU utilization [5]. The number of virtual network adapters varied depending on the number of SQL Server database instances, but all used the VMXNET3 adapter, which was designed for optimal performance [6]. SQL Server 2012 SP1 Enterprise Edition was installed for each database instance, with the following features: Database Engine Services, Full-Text Search, Client Tools Connectivity, Client Tools Backwards Compatibility, Management Tools – Complete. The Database Engine was configured with Mixed Mode Authentication. Each instance was split across two 132GB virtual hard disks and was created based on the `sqlserverds2_create_all_large2.sql` script provided with the DVD Store kit.

For tuning SQL Server, here are two great resources:

1. The SQL Server on VMware Best Practices Guide [7] has a wealth of recommendations and guidance for running SQL Server in a vSphere-based environment.

2. Performance Tuning Guidelines for Windows Server 2012 R2 [8] is a Microsoft document with a specific set of tuning recommendations for OLTP workloads [9]. This document includes tuning guidelines that are applicable to virtualized configurations. The tunings were found to have a **noticeable benefit** for the DVD

Store workload and are summarized below:

- Set the power plan to **High Performance**.
- Enable SQL Server to use large pages by enabling the **Lock pages in memory** user right assignment for the account that will run the SQL Server in Group Policy.
- Enable the TCP/IP protocol to allow communication with client systems:

  In Server Configuration Manager, navigate to **SQL Server Network Configuration > Protocols for MSSQL Server**, right-click **TCP/IP**, and click **Enable**.
- SQL Server Startup Parameters:
  - -x Disable SQL Server performance monitor counters
  - -T661 Disable the ghost record removal process
  - -T834 Use Microsoft Windows large-page allocations for the buffer pool
    Note: As with the previous study on vSphere 5.5, this parameter was not used because it hurt performance, caused slower startup time for SQL Server services, and required limiting each instance to a static slice of memory. Even if this parameter is not set, SQL Server *does* use large pages for other internal structures. This can be verified by looking for lines like the below in the startup log:

    ```
    2014-08-07 15:18:55.23 Server      Using locked pages in the memory manager.

    2014-08-07 15:18:55.24 Server      Large Page Allocated: 32MB
    ```

Other common SQL tunings, their values for these tests, and references to more information are noted below:

- Max degree of parallelism (MAXDOP) [10] is an advanced configuration option that controls the number of processors used to execute a query in a parallel plan. Setting this value to 1 disables parallel plans altogether, which is not recommended in a large virtual machine because it could result in unused virtual CPUs and wasted resources. Microsoft has guidelines for setting this value, depending on whether the host has NUMA and (or) Hyper-Threading, and there are numerous recommendations online, but many predate virtualization and modern server hardware [11]. The default value of **0** is recommended because it allows SQL Server to utilize all virtual CPUs available, and this was the value used for all testing.
- Cost threshold for parallelism [12] is an option that specifies the threshold at which parallel plans are used for queries. The value is specified in seconds, and the default is 5, which means a parallel plan for queries is used if SQL estimates it would take longer than 5 seconds if run serially. This value started to show very high response times with large virtual machines with 8 or more virtual CPUs, particularly for the DVD Store "login" query. Increasing this value to 50 [13] resolved this performance bottleneck and this value was used for all testing.
- Processor affinity [14] and I/O affinity [15] masks allow database administrators to assign SQL CPU and I/O to specific vCPUs and (or) vNUMA nodes, as shown in Figure 3.
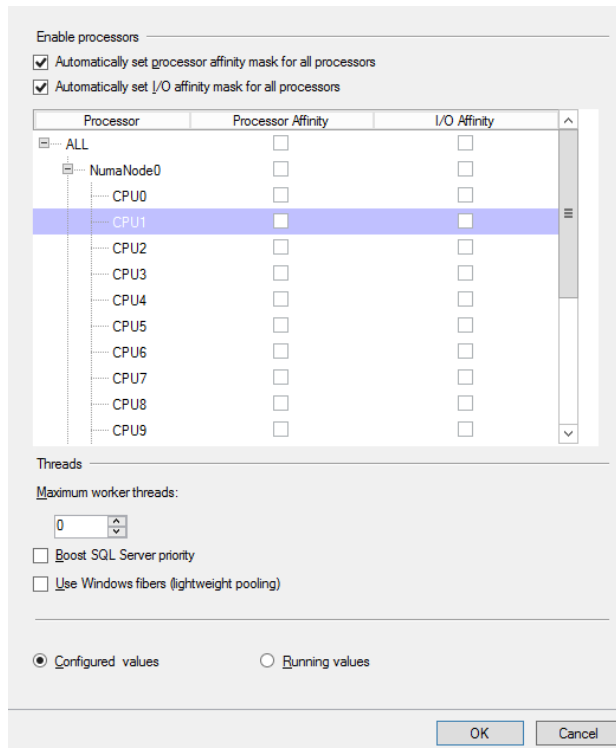
Figure 3. SQL Server Affinity settings (kept at defaults)

Additionally the number of **worker threads** can be set, **process priority** can be raised, and "Windows fibers" can be used (as shown above). All experiments with enabling these features resulted in lower performance, however, which is consistent with Microsoft guidance that these should only be set under rare circumstances, so the defaults are recommended.

- SQL Server licensing can affect performance [16] by potentially limiting scaling. Product features vary depending upon the edition (the three main editions are Standard, Business Intelligence, and Enterprise), and SQL Server 2012 licensing has two different licensing models (Core-Based or Server+CAL). During testing, the type of license was determined to be limiting the performance of virtual machines with more than 20 vCPUs. The SQL instance's ERRORLOG showed this message when the VM had more than 20 vCPUs:

```
SQL Server detected 32 sockets with 1 cores per socket and 1 logical processors
per socket, 32 total logical processors; using 20 logical processors based on SQL
Server licensing.
```

This was due to a 20 logical processor limitation imposed on the Server+CAL licenses we had used initially [17]. Fortunately, using a Core-Based licensing model can allow customers to license by VMs/vCPUs instead of the entire physical host.

See "Licensing SQL Server 2012 in a Virtualized Environment" in the *SQL Server 2012 Licensing Guide* [18] for all definitive statements about SQL Server 2012 licensing in virtualized environments.

- To scale up load on a SQL Server virtual machine, large tables can be partitioned, multiple databases can be hosted on one SQL instance, or multiple "named instances" of SQL Server can be installed.

For ease of benchmarking, multiple instances of the SQL Server Database Engine were installed, each with its own set of dedicated virtual resources (virtual NIC and virtual disks). This choice was made because the DVD Store workload is written to stress a specific database with a defined database schema. By using named instances, no code modifications need to be made to the load driver program; each simply listens on its own unique IP address.

Figure 4 shows how two instances can be configured to listen on different IP addresses (NICs) but the same TCP port within the SQL Server management interface.
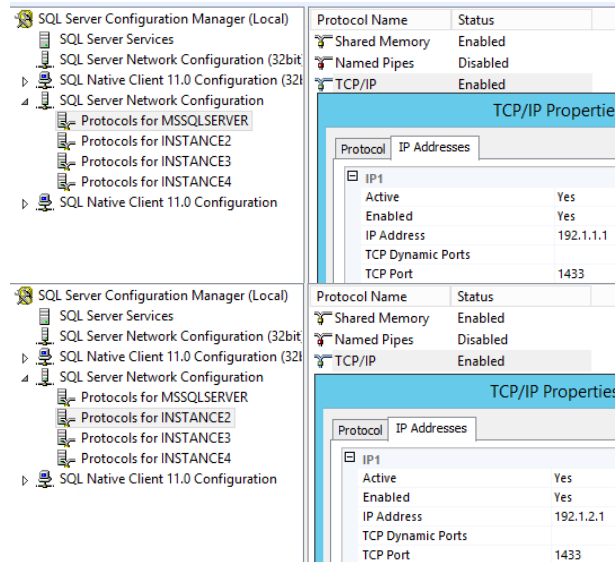
Figure 4. SQL Server instances listening on different IPs

For more information on named instances, see Multiple Instances of SQL Server [19].

DVD Store 2 Benchmark

Performance tests in this paper were undertaken using the DVD Store 2 benchmark [20]. DVD Store is an open-source tool that simulates an E-commerce Web application and advanced transactions against a backend database tier. It has been used in numerous third-party performance studies, and is a key application workload in the VMmark 2 benchmark [21]. For the tests in this study, the Web layer was bypassed so it was not a potential bottleneck, so the client load drivers could stress the database tier directly.

DVD Store 2 comes in three standard database sizes: small, medium, and large, as shown in Table 1.

DATABASE	SIZE	CUSTOMERS	ORDERS	PRODUCTS
Small	10 MB	20,000	1,000/month	10,000
Medium	1 GB	2,000,000	100,000/month	100,000
Large	100 GB	200,000,000	10,000,000/month	1,000,000

Table 1. DVD Store 2 database sizes

Given the goal of these tests was to stress large virtual machines, the large 100GB database size was selected. This simulates 200 million customers ordering 10 million DVDs per month from a catalog of 1 million DVDs.

In order to saturate virtual machines with large amounts of virtual CPUs and memory, multiple SQL Server databases (named instances) were used. Each SQL Server instance has its own 100GB large DVD Store database. For example, a 64-vCPU virtual machine had four 100GB database instances for a 400GB total working set.

Repeatability was ensured by restoring a backup of each database instance before each run. Steady state was achieved by ensuring throughput gradually increased during the 5-minute warmup period and reached/maintained steady state during each 10-minute run time. The number of driver processes and threads were adjusted as needed to saturate the number of virtual machines and database instances running for each test.

Generational Performance Comparison

The first test performed was to compare the maximum throughput from the previous-generation "Westmere-EX" server to the current-generation "Ivy Bridge-EX" server. The previous generation was a 4-socket 10-core (40 cores, 80 threads with Hyper-Threading) system, running ESXi 5.5, and the current generation system is also a 4-socket server, but with 15 cores per socket (60 cores, 120 threads total), and running ESXi 6.0. While this is by no means an apples-to-apples comparison, the intention was to measure the expected generational performance improvements in both host hardware and virtualization software.

The best-performing test case from the previous generation was used as a baseline (8 VMs, each with 8 vCPU). This configuration had fully saturated that host's processing capabilities while achieving efficient throughput results. For the current-generation platform, the same 8 VMs were used, but the number of vCPUs needed to be doubled from 8 to 16. This was necessary in order to fully utilize the additional processing cores of the newer server generation.

As shown in Figure 5 below, the maximum throughput from the new hardware and software platform outperformed the previous generation used in the October 2014 study by 83%.
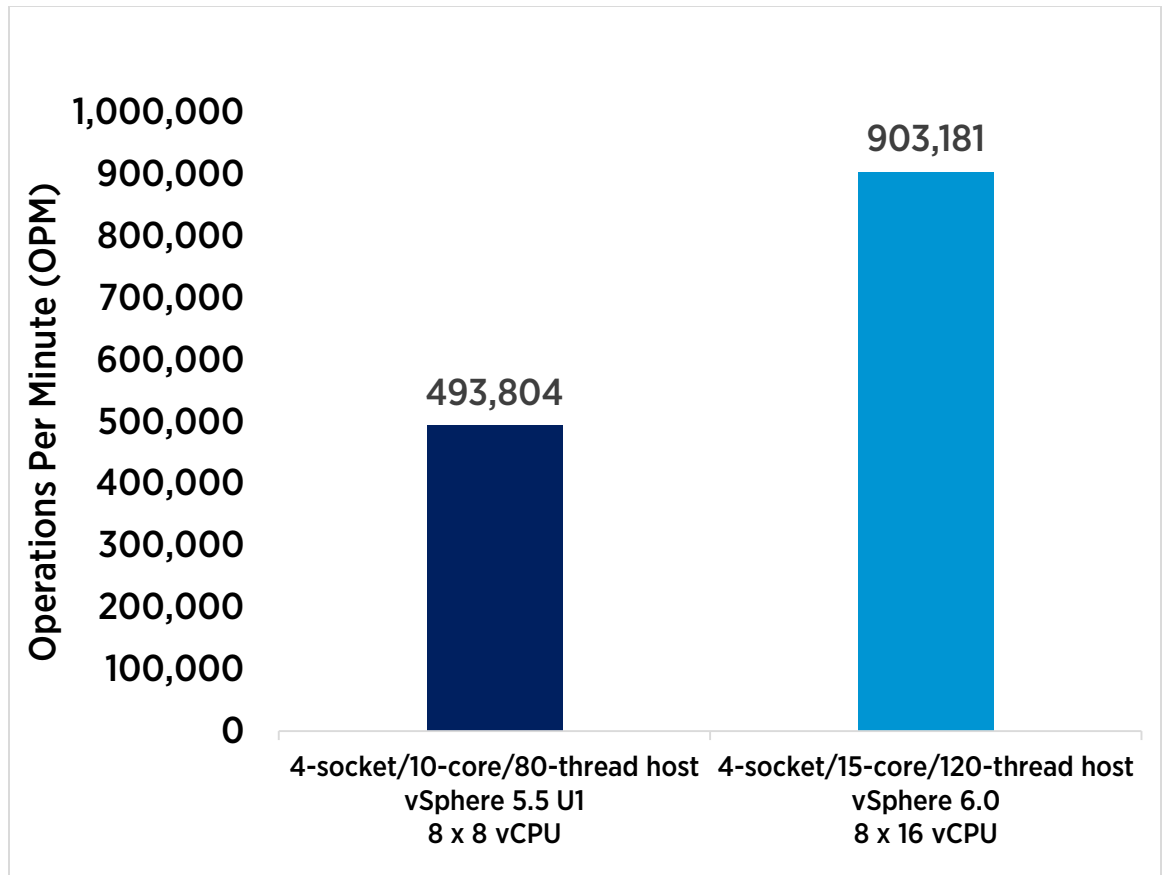
Figure 5. Generational performance improvement

Comparing Physical versus Logical Processors

The tests that follow were performed to determine the effects of pinning SQL Server virtual machines to specific physical and logical processors. Using CPU Affinity [22] allows vSphere administrators to "pin" a virtual machine's vCPUs to a specific set of physical CPUs (Hyper-Threads). While pinning is usually not recommended for production use, it sometimes provides better performance in specific controlled benchmark environments. CPU Affinity can be set through the vSphere Web Client, as shown in Figure 6. For more information, see Assign a Virtual Machine to a Specific Processor in the vSphere Web Client [23].
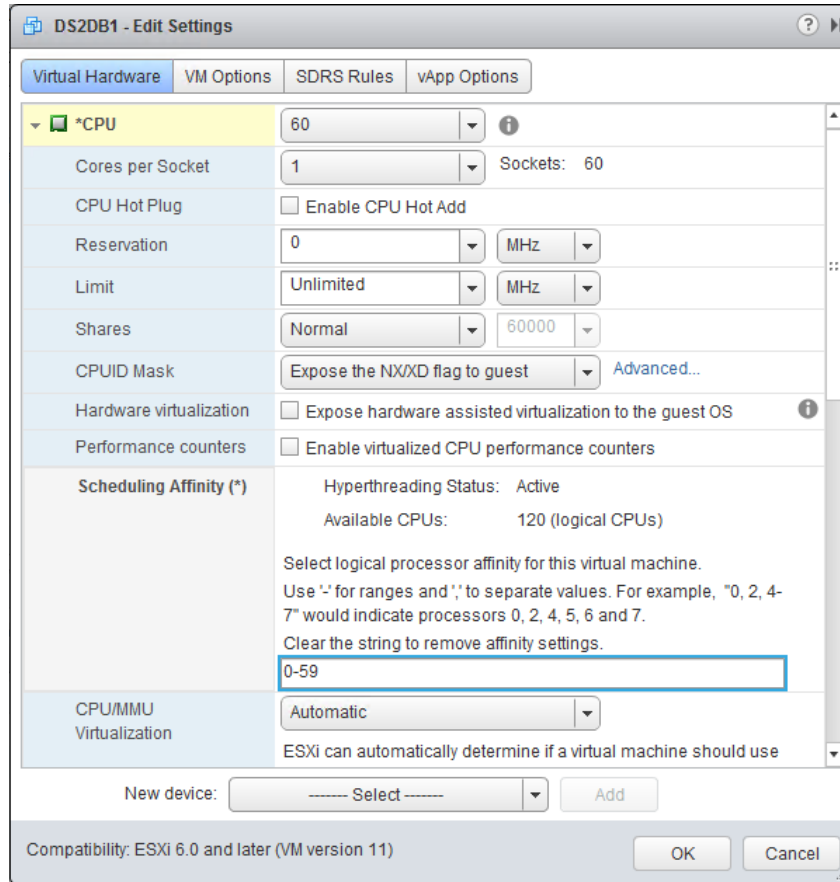
Figure 6. Assigning a virtual machine to specific CPUs

Note there are potential issues when using CPU affinity [24] that should be considered before users experiment with this feature. The ESXi CPU scheduler is aware of all the server and CPU architectures and makes decisions about where to schedule vCPUs based on this in combination with current usage patterns on the system. In order to manually make these decisions and decide which logical processor threads should be used for the vCPUs of a VM, an understanding of how the ESXi host enumerates the physical CPUs, also known as processor sockets, is needed. The following command was used to obtain this information:

```
vim-cmd hostsvc/hosthardware
```

As previously shown in Figure 2, the SUT used in this paper is a 4-socket server, each socket consisting of a 15-core Intel Xeon Processor E7-4890 v2 (120 total threads with Hyper-Threading). The following cpuInfo structure shows how ESXi interprets this internally; this is the ESXi physical CPU view.

```
    cpuInfo = (vim.host.CpuInfo) {
    numCpuPackages = 4,
    numCpuCores = 60,
    numCpuThreads = 120,
    hz = 2799999398
},
```

This confirms the ESXi host sees 4 processors (called packages here), 60 cores (15 physical cores per processor), and 120 logical processors (also known as Hyper-Threads).

Figure 7 shows how ESXi breaks down the processor packages into 120 threads, numbered 0-119:

15 cores (30 threads) within each "CPU package" (processor)

```
cpuPkg = (vim.host.CpuPackage)        (vim.host.CpuPackage) {     (vim.host.CpuPackage) {     (vim.host.CpuPackage) {
   (vim.host.CpuPackage) {               index = 1,                  index = 2,                  index = 3,
      index = 0,                         vendor = "intel",           vendor = "intel",           vendor = "intel",
      vendor = "intel",                  hz = 2799999651,            hz = 2799999609,            hz = 2799999367,
      hz = 2799999338,                   busHz = 99999952,           busHz = 99999945,           busHz = 99999945,
      busHz = 99999969,                  description = "Intel        description = "Intel         description = "Intel
      description = "Intel(R) }          threadId = (short) [        threadId = (short) [        threadId = (short) [
      threadId = (short) [                  30,                         60,                         90,
         0,                                 31,                         61,                         91,
         1,                                 32,                         62,                         92,
         2,                                 33,                         63,                         93,
         3,                                 34,                         64,                         94,
         4,                                 35,                         65,                         95,
         5,                                 36,                         66,                         96,
         6,                                 37,                         67,                         97,
         7,                                 38,                         68,                         98,
         8,                                 39,                         69,                         99,
         9,                                 40,                         70,                         100,
         10,                                41,                         71,                         101,
         11,                                42,                         72,                         102,
         12,                                43,                         73,                         103,
         13,                                44,                         74,                         104,
         14,                                45,                         75,                         105,
         15,                                46,                         76,                         106,
         16,                                47,                         77,                         107,
         17,                                48,                         78,                         108,
         18,                                49,                         79,                         109,
         19,                                50,                         80,                         110,
         20,                                51,                         81,                         111,
         21,                                52,                         82,                         112,
         22,                                53,                         83,                         113,
         23,                                54,                         84,                         114,
         24,                                55,                         85,                         115,
         25,                                56,                         86,                         116,
         26,                                57,                         87,                         117,
         27,                                58,                         88,                         118,
         28,                                59                          89                          119
         29
```
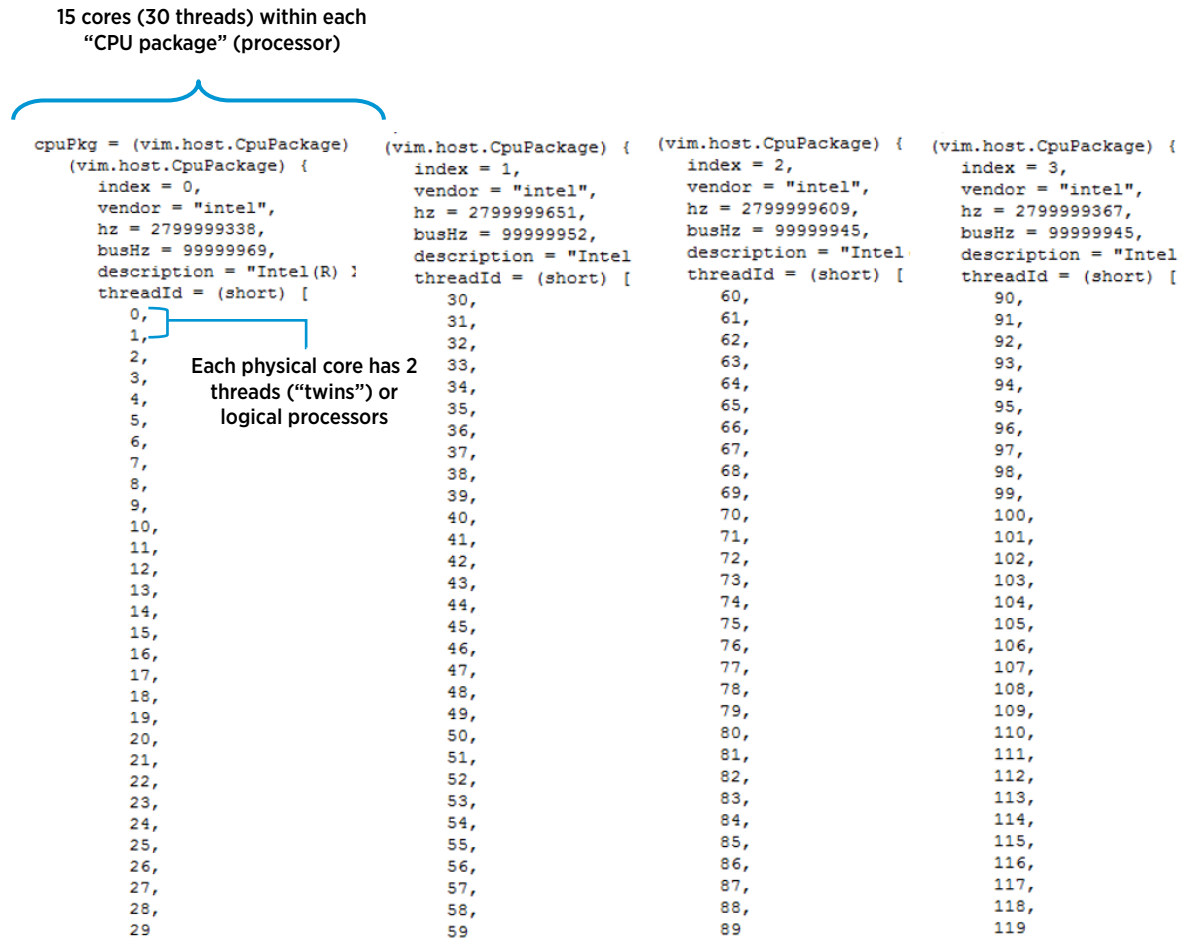
Each physical core has 2 threads ("twins") or logical processors

Figure 7. ESXi processor threads view

A key takeaway from this figure is that each physical core is sequentially numbered; the first physical core has threads 0 and 1, the second physical core has threads 2 and 3, and so on. Since each core's pair of threads share physical processor resources (namely cache), virtual machines are preferentially scheduled on **two different cores** rather than on two logical processors on the same core. The next section demonstrates why the ESXi scheduler uses this placement policy.

60 Physical Cores versus 60 Logical Processors

The purpose of this test is to show the performance of a 60 vCPU VMs when ESXi is allowed to schedule the VM across all of the host's processors (again, by default, physical cores are preferred) versus a VM that is affinitized (restricted) to the first two processor packages, by assigning it to logical CPUs 0-59 (see Figure 7 above).

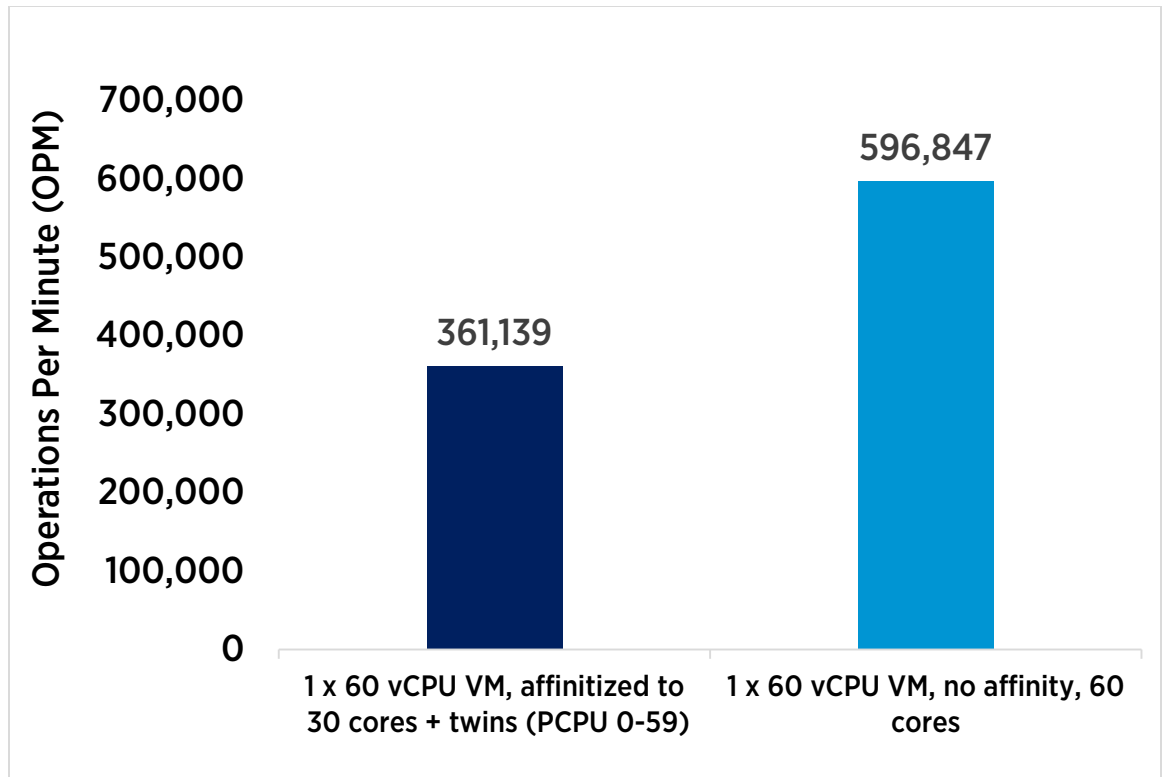The results are shown below in Figure 8.

Figure 8. 60 physical cores vs. 60 logical processors on 30 cores

Results show that the test using 60 physical cores achieved **596,847** operations per minute (OPM), while 30 cores could only achieve 361,139 OPM, or approximately 60% of 60 core performance. This confirms that the ESXi scheduler is making the right decision about preferring physical cores over Hyper-Threads when there are idle cycles of the former. When physical cores are fully utilized, however, Hyper-Threading does provide a noticeable benefit, which is why it is enabled by default.

Virtual Machine Sizing Tests

The goal of the tests in this section was to achieve maximum throughput with different numbers of virtual machines, where the number of vCPUs per virtual machine starts at 16, then is doubled to 32, then 64. Although vSphere 6 now supports a maximum of 128 virtual CPUs per VM, there were only 120 logical processors (and only 60 physical cores) on the system under test.

In addition to the 16/32/64 vCPU testing, the VMs were subsequently "right-sized" with respect to the underlying 15-core Xeon E7-4890 v2 processors, then doubling that to 30, then 60. The goal here was to confirm that this best practice does actually result in better performance.

The results of these alternative ways of partitioning the VM sizes are shown in Figure 9 below.
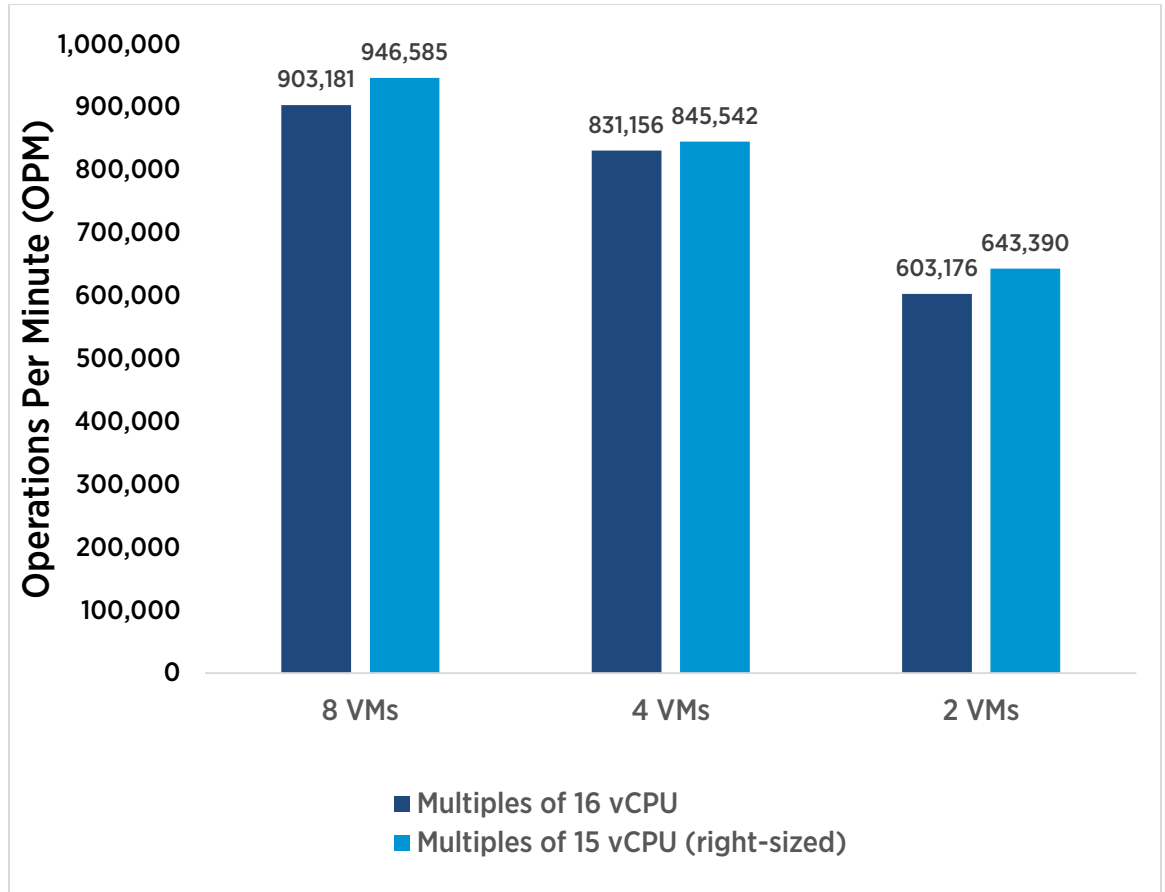
Figure 9. Aggregate scale-up throughput

This data reveals several important points. There is definitely an advantage to right-sizing the VMs with respect to the underlying host processor architecture, because the VMs with multiples of 15 vCPUs are consistently higher than VMs with multiples of 16 vCPUs. Bigger is not always better; it is always a best practice to use the smallest VM possible to get the job done for best performance.

Latency Sensitivity Tests

ESXi is optimized for high *throughput* to minimize virtualization overhead for a wide range of applications, including business-critical applications such as databases. DVD Store measures throughput in **orders per minute (OPM)**. *Latency*, on the other hand, measures the time it takes to transfer data from one endpoint to another and is typically measured in milliseconds. Latency-sensitive applications typically include VoIP services or high-frequency trading, where low response times are critical; transactional database applications usually do not fall into this category.

vSphere includes an advanced setting to support virtual machines with strict latency requirements to lower response time and reduce jitter. This is set through the vSphere Web Client on a per-VM basis, as Figure 10 shows. For more information, see Deploying Extremely Latency-Sensitive Applications in VMware vSphere 5.5 [25]. Although this technical paper was written for vSphere 5.5, the behavior is unchanged in 6.0.
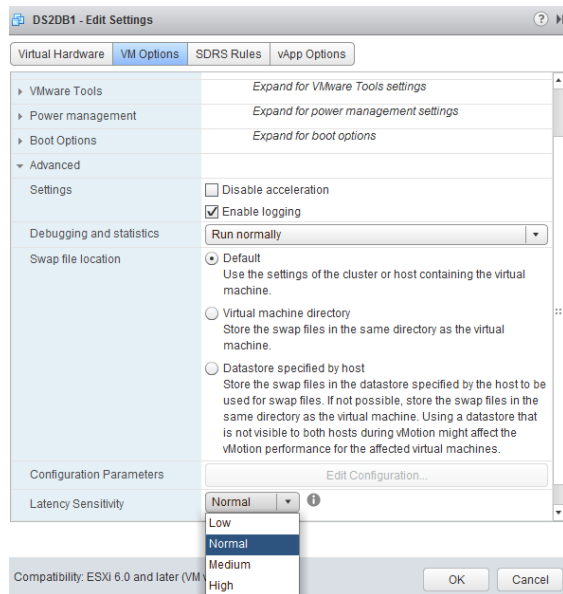
Figure 10. vSphere Latency Sensitivity setting choices

To verify that DVD Store is not a latency-sensitive workload, eight 15-vCPU virtual machines (the best performing case from the preceding throughput tests) were tested with the two supported Latency Sensitivity choices: **Normal** (default) and **High**. Note that the **Low** and **Medium** settings are currently unsupported, which can be confirmed by clicking the Help icon.
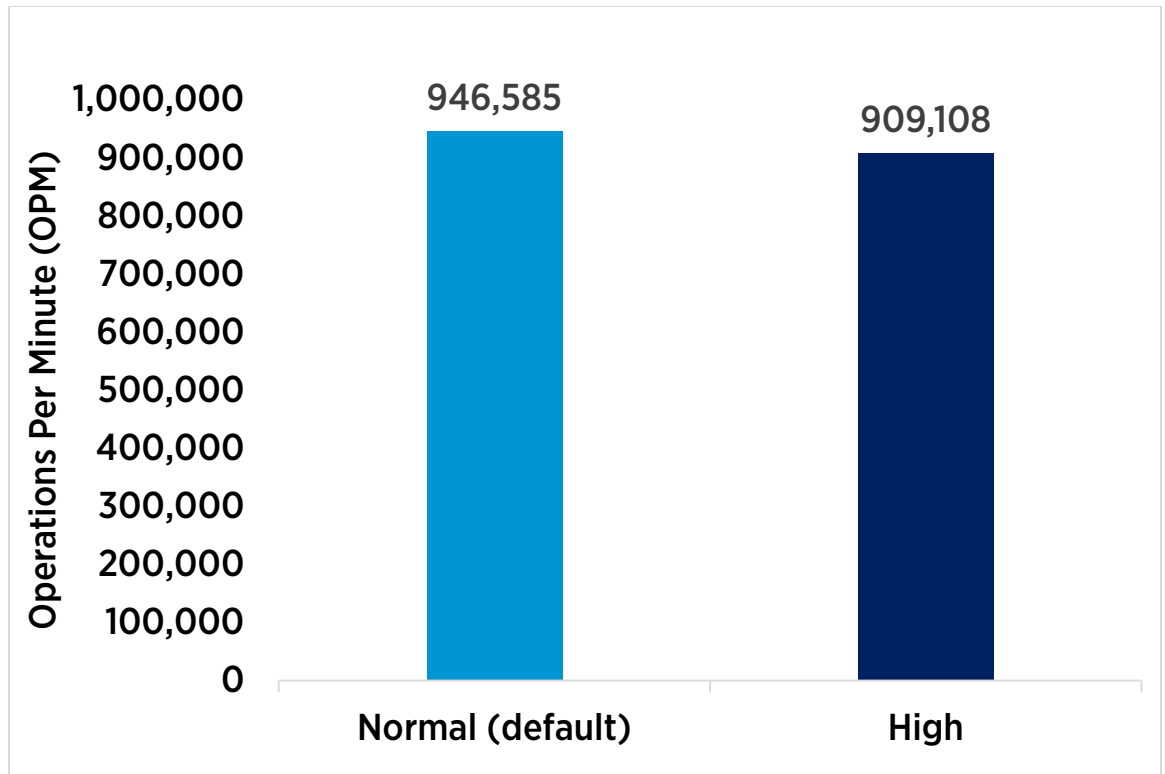
Figure 11. Impact of Latency Sensitivity setting for 8 x 15-vCPU configuration

These results (Figure 11) show that this OLTP test on SQL Server is not a latency-sensitive workload. By switching from Normal (the default setting) to High, the aggregate throughput dropped by 4%. The drop occurs because tuning for latency comes with a significant CPU cost, which is why the default is not "High."

Best Practices

- **Make sure guest operating systems are running VMware Tools, as well as the latest virtual hardware version.**
  Important: Configuring a large number of vCPUs is possible only by upgrading to the latest virtual hardware version, and VMware Tools allows the use of paravirtual storage and network drivers.

- **Size virtual machines with respect to the underlying processor architecture.**
  The virtual NUMA defaults are optimal for most cases, but VMs that do not cleanly pack well (both vCPU counts and memory) into physical NUMA nodes can cause suboptimal virtual NUMA layouts. For more information, see the *vSphere Resource Management* guide [26].

- **Let the ESXi scheduler balance processor load for virtual machines. It does so automatically and efficiently.**
  Setting CPU affinity for individual virtual machines may provide a modest performance boost, but this action requires an intimate understanding of the underlying physical architecture, and may interfere with vSphere features such as vMotion.

- **Tune SQL Server for virtual machines with large databases and (or) large numbers of vCPUs.**
  Microsoft SQL Server 2012 SP1 is largely auto-tuning; NUMA is automatically detected and used, even in a virtual environment. Soft NUMA, processor affinity, and I/O affinity masks are unnecessary. However, some minimal tunings for using large vCPU counts that do not cleanly match the server processor core counts are necessary. Below is a summary of what benefited this particular DVD Store setup, but each environment is different.
  - **Parallelism settings:** The Max Degree of Parallelism should be left at 0, but the Cost Threshold for Parallelism setting default of 5 may be too low (it was set to 50 for these tests).
  - **Database scaling:** If the SQL Database Engine, or the database itself, seems to be a bottleneck, consider installing multiple DB engine "named instances," and bind them to different IPs/virtual NICs.

  For details, see "Operating System and SQL Server Installation/Tuning" on page 3. These are also useful third party references:
  - Five SQL Server Settings to Change [13]
  - Top 10 Keys to Deploying SQL Server on VMware [27]
  - Benchmark Your SQL Server Instance with DVD Store [28]

- **Defaults for Virtual NUMA and Cores per Socket should be optimal.**
  While selection of the total number of virtual CPUs is important, the virtual NUMA topology ESXi selects by default should be optimal. CoresPerSocket defaults to 1, which allows ESXi to select the best vNUMA topology based on the underlying hardware. It is recommended that this setting not be changed except for some particular use cases.

- **SQL workloads are not highly sensitive to latency.**
  Latency Sensitivity is an advanced setting designed for high performance computing applications like financial workloads. This setting does not benefit DVD Store, an OLTP workload designed to achieve maximum throughput. Also see "Latency Sensitivity" in the *VMware vSphere 6.0 Documentation Center* [29].

Conclusion

SQL Server workloads achieve excellent performance with vSphere 6 on top of today's massively multi-core, large memory server hardware. When scaling up the number of virtual machines or vCPUs, some minimal tuning of the database software and ESXi host can help achieve maximum throughput.

Appendix A

The following table describes the hardware and software used in the test bed.

VIRTUALIZATION PLATFORM	
Hypervisor Vendor, Product, Version, and Build / Availability Date	VMware ESXi 6.0 build 2494585 / 03-12-2015
Datacenter Management Software Vendor, Product, Version, and Build / Availability Date	VMware vCenter Server Appliance 6.0 build 2503528 / 03-12-2015
SERVER PLATFORM	
Server Manufacturer/Model	Dell PowerEdge R920
Processor Vendor/Model	Intel® Xeon® E7-4890 v2
Processor Speed	2.8GHz (Intel Turbo Boost Technology up to 3.4GHz)
Total Sockets/Total Cores/Total Threads	4 Sockets / 60 Cores / 120 Threads
Primary Cache	32KB I + 32KB D on chip per core
Secondary Cache	256KB I + D on chip per core
Tertiary Cache	37.5MB I + D on chip per chip
BIOS Version	1.3.2 9/30/2014
Memory Size (in GB, Number of DIMMs)	1024GB, 64 x 16GB
Memory Type and Speed	Dual rank x4 PC3L-12800 (DDR3L-1600 MT/s) Registered ECC
Disk Subsystem Type	FC SAN
Number of Disk Controllers	2
Disk Controller Vendor and Model	Dell PERC H730P RAID Controller
Number of Host Bus Adapters	1 dual-port
Host Bus Adapter Vendors and Models	QLE2662 dual port 16Gb Fibre Channel HBA

STORAGE	
Array Vendor, Model, and Firmware Version	EMC VNX 5800 (version 05.33.000.5.081)
Fibre Channel Switch Vendor and Model	Brocade 300 8Gb
Array Cache Size	32GB per SP
Total Number of Physical Disks Used	100 (70 HDDs, 30 SSDs)
Total Number of Enclosures/Pods/Shelves Used	4
Disk Vendors, Models, and Speeds	30 x SAS SSD (SS162512 CLAR200) 70 x SAS HDD (SS930065 CLAR300)
NETWORK	
Network Switch Vendor and Model	HP ProCurve 5406zl
Network Speed	10Gbps, 1Gbps
Network Controller Vendors/Models	1 x Intel I350 1GbE Quad-Port 2 x Intel 82599EB 10-Gigabit SFP+ Dual-Port
DATABASE VM INFORMATION	
Guest Operating System	Windows Server 2012 R2
Database Application	SQL Server 2012 SP1
Virtual Processors	16, 32, 64 vCPU 15, 30, 60, 120 vCPU
Virtual RAM	16GB-1TB
Benchmarking Software	DVD Store 2.1

Table 2. Hardware and software

References

[1] VMware, Inc. Virtualizing Business-Critical Applications on vSphere.
http://www.vmware.com/files/pdf/solutions/VMware-Virtualizing-Business-Critical-Apps-on-VMware_en-wp.pdf

[2] Priya Sethuraman. (2009) Performance and Scalability of Microsoft SQL Server on VMware vSphere 4.
http://www.vmware.com/files/pdf/perf_vsphere_sql_scalability.pdf

[3] VMware, Inc. (2015) Configuration Maximums for VMware vSphere 6.0.
https://www.vmware.com/pdf/vsphere6/r60/vsphere-60-configuration-maximums.pdf

[4] Todd Muirhead. (2015, March) VMware vSphere 6 and Oracle 12c Scalability Study: Scaling Monster Virtual Machines. http://blogs.vmware.com/performance/2015/03/vmware-vsphere-6-oracle-12c-scalability-study-scaling-monster-virtual-machines.html

[5] VMware, Inc. (2014, June) Configuring disks to use VMware Paravirtual SCSI (PVSCSI) adapters (1010398).
http://kb.vmware.com/kb/1010398

[6] VMware, Inc. (2014, July) Choosing a network adapter for your virtual machine (1001805).
http://kb.vmware.com/kb/1001805

[7] VMware, Inc. (2012) SQL Server on VMware Best Practices Guide.
http://www.vmware.com/files/pdf/solutions/SQL_Server_on_VMware-Best_Practices_Guide.pdf

[8] Microsoft. (2014, April) Performance Tuning Guidelines for Windows Server 2012 R2.
http://msdn.microsoft.com/en-us/library/windows/hardware/dn529133

[9] Microsoft. Performance Tuning for Online Transaction Processing (OLTP). http://msdn.microsoft.com/en-us/library/windows/hardware/dn567642

[10] Microsoft. Recommendations and guidelines for the "max degree of parallelism" configuration option in SQL Server. http://support.microsoft.com/kb/2806535

[11] Brent Ozar. (2010, September) SQL Server Virtualization: Get Your Hands Off That MAXDOP.
http://www.sqlskills.com/blogs/brent/sql-server-virtualization-get-your-hands-off-that-maxdop/

[12] Microsoft. (2012) Configure the cost threshold for parallelism Server Configuration Option.
http://msdn.microsoft.com/en-us/library/ms190949(v=sql.110).aspx

[13] Brent Ozar. (2013, September) Five SQL Server Settings to Change.
http://www.brentozar.com/archive/2013/09/five-sql-server-settings-to-change/

[14] Microsoft. (2012) affinity mask Server Configuration Option. http://msdn.microsoft.com/en-us/library/ms187104(v=sql.110).aspx

[15] Microsoft. (2012) affinity I/O mask Server Configuration Option. http://msdn.microsoft.com/en-us/library/ms189629(v=sql.110).aspx

[16] Michael Otey. (2012, December) SQL Server 2012 Licensing Help. http://sqlmag.com/sql-server-2012/sql-server-2012-licensing-help

[17] Jonathan Kehaylas. (2012, November) Performance Problems with SQL Server 2012 Enterprise Edition Under CAL Licensing. http://sqlperformance.com/2012/11/system-configuration/2012-cal-problems

[18] Microsoft. (2012) Microsoft SQL Server 2012 Licensing Guide.
http://download.microsoft.com/download/7/3/C/73CAD4E0-D0B5-4BE5-AB49-D5B886A5AE00/SQL_Server_2012_Licensing_Reference_Guide.pdf

[19] Microsoft. (2000) Multiple Instances of SQL Server. http://technet.microsoft.com/en-us/library/aa174516.aspx

[20] Dave Jaffe and Todd Muirhead. DVD Store. http://en.community.dell.com/techcenter/extras/w/wiki/dvd-store.aspx

[21] VMware, Inc. (2015) VMmark 2.5 - Virtualization Platform Benchmark.
http://www.vmware.com/products/vmmark

[22] VMware, Inc. (2015) Using CPU Affinity. http://pubs.vmware.com/vsphere-60/topic/com.vmware.vsphere.resmgmt.doc/GUID-5FCC0AAA-9D58-4EDF-9A5F-8628F1C8B830.html

[23] VMware, Inc. Assign a Virtual Machine to a Specific Processor. http://pubs.vmware.com/vsphere-60/index.jsp?topic=%2Fcom.vmware.vsphere.hostclient.doc%2FGUID-523DAA21-202A-469B-9A89-1B192C4556F7.html

[24] VMware, Inc. (2015) Potential Issues with CPU Affinity. http://pubs.vmware.com/vsphere-60/index.jsp#com.vmware.vsphere.resmgmt.doc/GUID-199F1ADD-3B9B-490C-8F0C-440B53577B3A.html

[25] Jin Heo and Lenin Singaravelu. (2013, October) Deploying Extremely Latency-Sensitive Applications in VMware vSphere 5.5. http://www.vmware.com/resources/techresources/10383

[26] VMware, Inc. (2015) vSphere Resource Management. http://pubs.vmware.com/vsphere-60/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-60-resource-management-guide.pdf

[27] Brent Ozar. (2011, May) Top 10 Keys to Deploying SQL Server on VMware.
http://www.brentozar.com/archive/2011/05/keys-deploying-sql-server-on-vmware/

[28] David Klee. (2013, July) Benchmark Your SQL Server Instance with DVD Store.
http://www.davidklee.net/2013/07/29/benchmark-your-sql-server-instance-with-dvdstore/

[29] VMware, Inc. (2015) Latency Sensitivity. http://pubs.vmware.com/vsphere-60/index.jsp#com.vmware.vsphere.resmgmt.doc/GUID-9F4FD589-A73B-454A-A5A5-FED4C0F918C3.html

[30] VMware, Inc. Performance Best Practices for vSphere 5.5.
http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.5.pdf

[31] Mark Achtemichuk. (2013, October) Does corepersocket Affect Performance?
http://blogs.vmware.com/vsphere/2013/10/does-corespersocket-affect-performance.html

[32] VMware, Inc. (2014, July) Configure virtual machines to use hyper-threading with NUMA in VMware ESXi 4.1.x / 5.0.x (2003582). http://kb.vmware.com/kb/2003582

[33] VMware, Inc. (2015) Using Virtual NUMA. http://pubs.vmware.com/vsphere-60/topic/com.vmware.vsphere.resmgmt.doc/GUID-17B629DE-75DF-4C23-B831-08107007FBB9.html

[34] VMware, Inc. (2015) Advanced Virtual NUMA Attributes. http://pubs.vmware.com/vsphere-60/index.jsp?topic=%2Fcom.vmware.vsphere.resmgmt.doc%2FGUID-89C52376-60C3-452A-A269-9F4F7FE489C6.html

About the Author

David Morse is a Staff Engineer on the Performance Engineering team at VMware. He has 16 years of benchmarking experience at VMware, Dell Inc., and NCR Corp. He has held a variety of roles within the Standard Performance Evaluation Corporation (SPEC), including chairing the Open Systems Steering Committee and serving on the Board of Directors. David has a B.S. in Computer Engineering from the University of South Carolina and is a Red Hat Certified Engineer (RHCE).

Acknowledgements

The author would like to thank: Todd Muirhead, for his DVD Store advice; David Klee, for his SQL Server expertise; Seongbeom Kim, for his vNUMA knowledge; and colleagues within the performance engineering organization, many of whom provided valuable feedback throughout this study.

vmware®