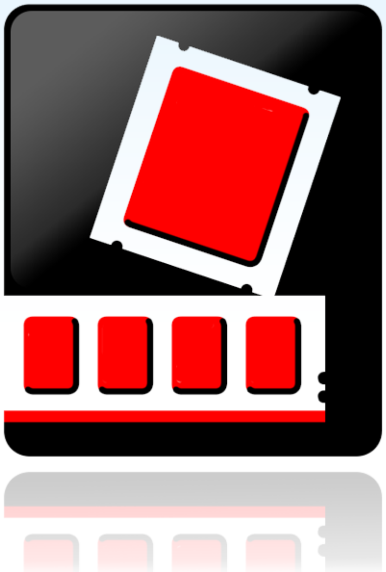
COMPARING CPU AND MEMORY PERFORMANCE: RED HAT ENTERPRISE LINUX 6 VS. MICROSOFT WINDOWS SERVER 2012

Red Hat® Enterprise Linux® 6



better memory and CPU performance

versus Microsoft® Windows Server® 2012

redhat

An operating system's ability to effectively manage and use server hardware often defines system and application performance. Processors with multiple cores and random access memory (RAM) represent the two most vital subsystems that can affect the performance of business applications. Selecting the best performing operating system can help your hardware achieve its maximum potential and enable your critical applications to run faster.

To help you make that critical decision, Principled Technologies compared the CPU and RAM performance of Red Hat Enterprise Linux 6 and Microsoft Windows Server 2012 using three benchmarks: SPEC® CPU2006, LINPACK, and STREAM.

We found that Red Hat Enterprise Linux 6 delivered better CPU and RAM performance in nearly every test, outperforming its Microsoft competitor in both out-of -box and optimized configurations.

APRIL 2013

A PRINCIPLED TECHNOLOGIES TEST REPORT

Commissioned by Red Hat, Inc.

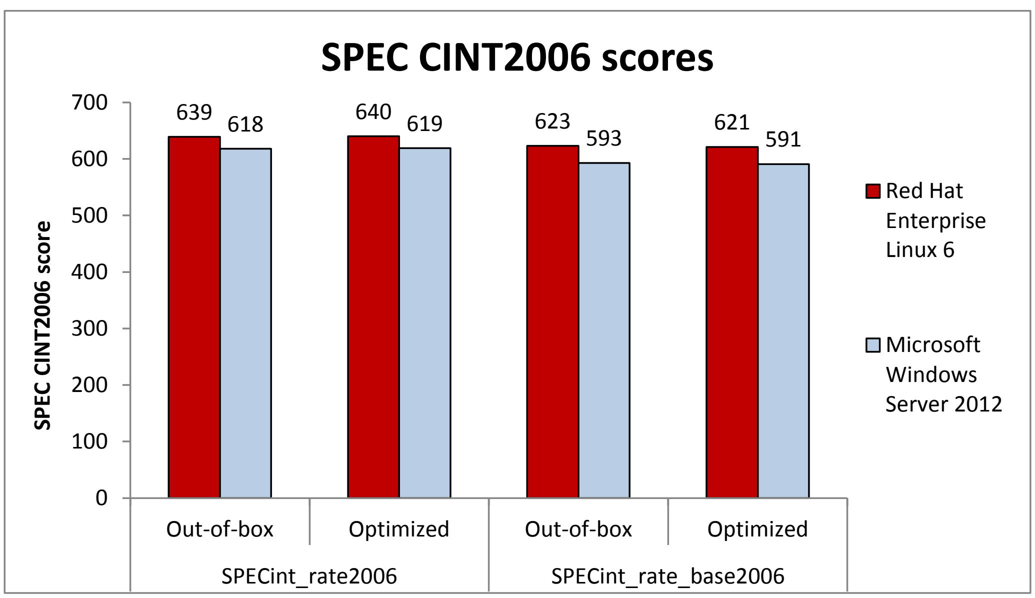
BETTER CPU AND RAM PERFORMANCE

We compared CPU and RAM performance on two operating systems: Red Hat Enterprise Linux 6 and Microsoft Windows Server 2012. For our comparison, we used the SPEC CPU2006 benchmark and LINPACK benchmark to test the CPU performance of the solutions using the different operating systems, and the STREAM benchmark to test the memory bandwidth of the two solutions. For each test, we first configured both solutions with out-of-box (default) settings, and then we tested those solutions using multiple tuning parameters to deliver optimized results. We ran each test three times and report the results from the median run. For detailed system configuration information, see Appendix A. For details on how we tested, see Appendix B.

Testing the CPU with SPEC CPU2006 and LINPACK

When we tested CPU performance, we found that the server running Red Hat Enterprise Linux 6 scored up to 5.1 percent higher than the Microsoft Windows Server 2012 solution on the SPEC CPU2006 benchmark, both out-of -box and optimized. The SPEC CPU2006 benchmark consists of two benchmark suites, each of which focuses on a different aspect of compute-intensive performance. SPEC CINT®2006 measures and compares compute-intensive integer performance, while SPEC CFP®2006 measures and compares compute-intensive floating-point performance. A “rate” version of each, which runs multiple instances of the benchmark to assess server performance, is also available. We ran the rate version of these benchmarks. Figures 1 and 2 show the scores that the systems achieved on both parts of the benchmark. For detailed test results, see Appendix C.

Figure 1: SPEC CPU2006 results, in SPEC CINT2006 scores, for the two solutions. Higher numbers are better.



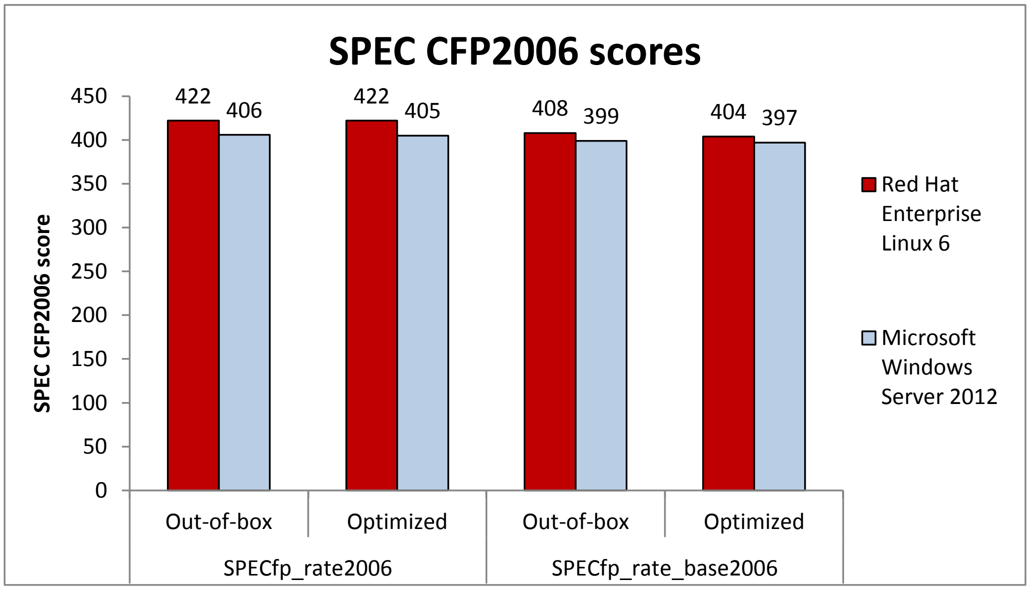


Figure 2: SPEC CPU2006 results, in SPEC CFP2006 scores, for the two solutions. Higher numbers are better.

Measuring floating point computing power assesses the performance of high-performance systems that need to do heavy work. We used the LINPACK benchmark to test the floating point performance of the platforms out-of -box and optimized, which shows how a system performs under an extreme load. As Figure 3 shows, Red Hat Enterprise Linux 6 outperformed Windows Server 2012 when using all 16 physical cores on our test system. In addition, tuning the Red Hat Enterprise Linux 6 system increased performance steadily when moving from 4 threads to 16 threads, while optimizing Windows Server 2012 had little effect on its performance at 16 threads. In our tests, we enabled Intel® Hyper-Threading Technology, but bound the workload to at most one thread per physical core. We found that allocating every available thread (out of 32) to the test did not increase LINPACK performance for either platform. For detailed test results, see Appendix C.

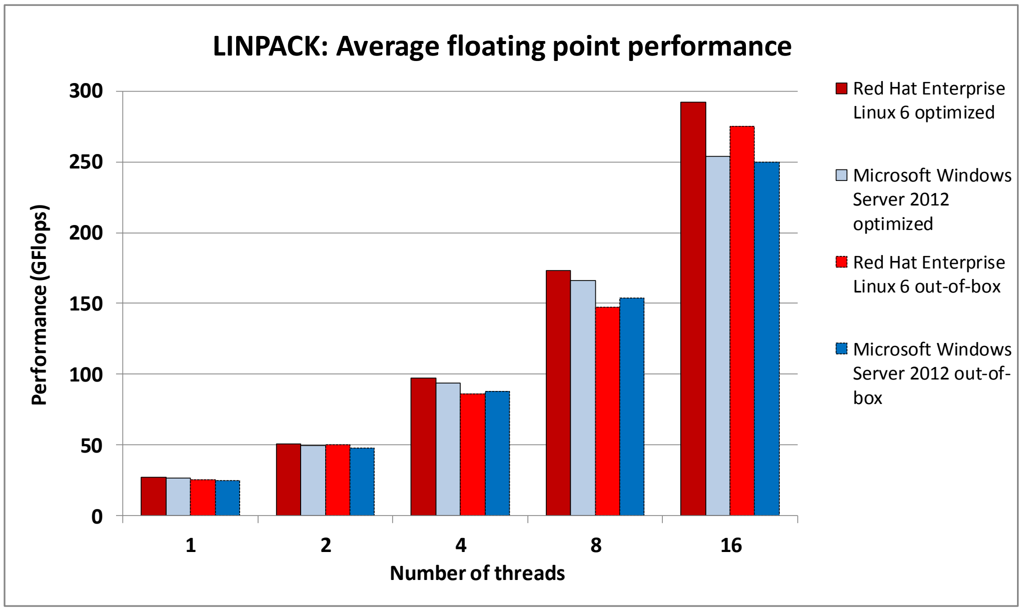


Figure 3: LINPACK floating point performance results for the two operating system solutions, both out-of-the-box and optimized. Higher numbers are better.

Testing RAM performance

Because operating systems manage physical memory, the effective memory bandwidth, which is the rate at which a system can read or write data from memory, is often dependent on OS capabilities and configuration. We used the STREAM benchmark to measure the memory bandwidth both Red Hat Enterprise Linux 6 and Microsoft Windows Server 2012 delivered. As Figures 4 and 5 show, both solutions performed comparably in this area, both out-of-box and optimized. In our tests, we enabled Intel Hyper-Threading Technology, but bound the workload to at most one thread per physical core. We found that allocating every available thread (out of 32) to the test did not increase STREAM performance for either platform. For detailed test results, see Appendix C.

Figure 4: Out-of-box memory bandwidth comparison using the STREAM benchmark.

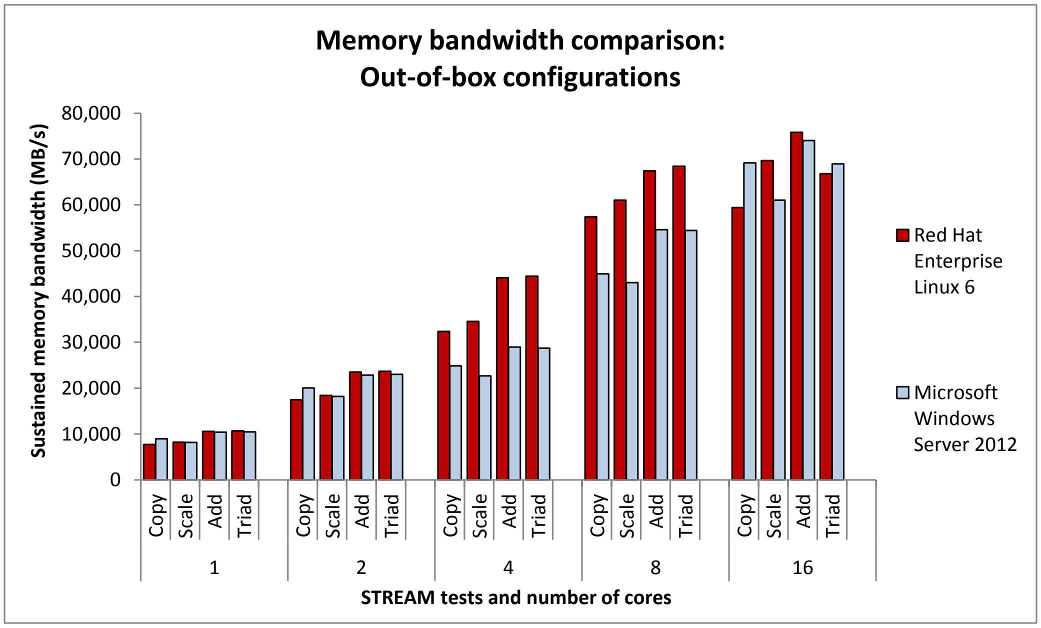
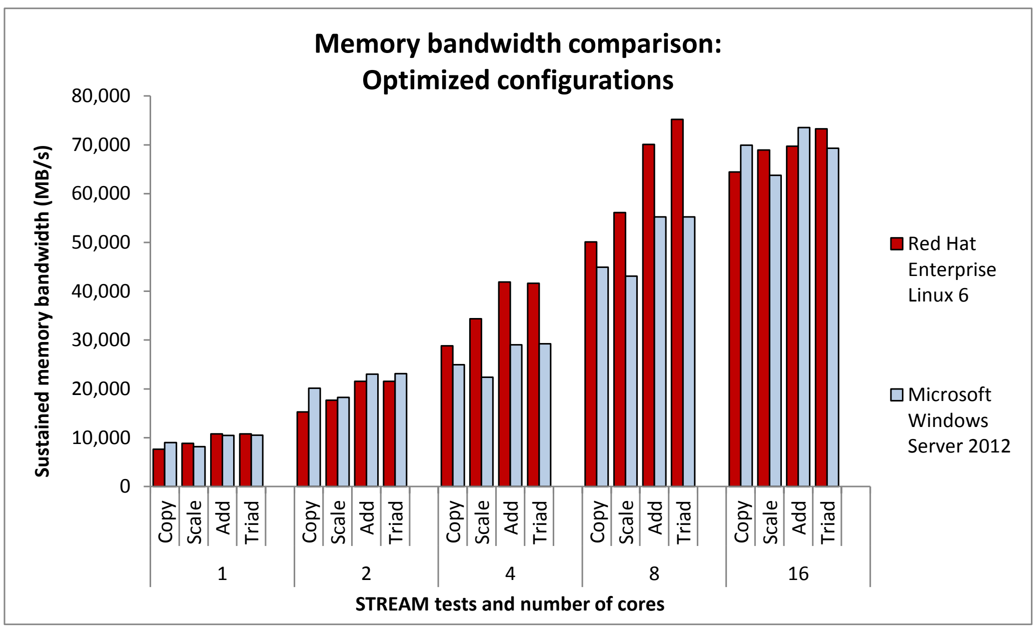


Figure 5: Optimized memory bandwidth comparison using the STREAM benchmark.



WHAT WE TESTED

About Red Hat Enterprise Linux 6

Red Hat Inc. positions Red Hat Enterprise Linux 6 as a prime competitor to proprietary operating systems found in enterprise data centers. Red Hat Enterprise Linux 6 is designed to deliver performance and scalability for both small and large servers with documented scalability up to 4,096 CPUs and 64 terabytes of RAM. It provides native support for the majority of the latest and most important enterprise data center technologies, such as 40Gb Ethernet networking and KVM virtualization as well as InfiniBand®, FCoE, and iSCSI protocols. According to Red Hat, Red Hat and its hardware partners are enabling reliability, availability, serviceability (RAS), and scalability features to help minimize downtime, increase availability, and protect data. Red Hat includes as part of its Linux offering tested, open source applications. For more information about Red Hat Enterprise Linux 6, see http://www.redhat.com/f/pdf/rhel/RHEL6_datasheet.pdf.

About SPEC CPU2006

SPEC CPU2006 is an industry-standard benchmark that uses a CPU-intensive workload to stress a system's processor(s), memory subsystem, and compiler. SPEC CPU2006 encompasses two types of tests: SPEC CINT2006, which reports results in both SPECint®_rate2006 and SPECint_rate_base2006 scores, and SPEC CFP2006, which reports results in both SPECfp®_rate2006 and SPECfp_rate_base2006 scores. These scores help compare a wide range of hardware. For more information about the SPEC CPU2006 benchmark, visit http://www.spec.org/cpu2006/.

About LINPACK

The LINPACK benchmark runs a program that solves a system of linear equations to measure the floating point rate of execution of a system. Often used to test the performance of supercomputers, LINPACK can help determine the peak performance that a system is capable of by using complex calculations to stress the processor. For more information about LINPACK, visit http://www.top500.org/project/linpack/.

About STREAM

STREAM is an industry-standard benchmark that measures the memory bandwidth of a system. The benchmark measures real-world sustained memory bandwidth for typical operations on data, instead of peak memory performance rates. For more information about the STREAM benchmark, visit http://www.streambench.org.

IN CONCLUSION

Understanding how your system resources are utilized and how well they perform can be extremely valuable as you plan your infrastructure, making the selection of the operating system a pivotal decision that could influence your IT strategy for many years to come. Throughout our CPU and RAM tests, we found that the open-source Red Hat Enterprise Linux 6 solution performed as well or better than Microsoft Windows Server 2012. In our SPEC CPU2006 tests, the Red Hat Enterprise Linux 6 solution achieved consistently higher scores than the Windows Server 2012 solution. When we used the LINPACK benchmark to test floating point performance of CPUs, we also found that tuning the operating system allowed us to get even greater performance out of the Red Hat Enterprise Linux 6 system. In our memory bandwidth tests, the Red Hat Enterprise Linux 6 solution outperformed the Windows Server 2012 solution at mid-range thread counts.

By choosing an operating system that can deliver strong performance on all subsystems out of the box and increase performance even more when tuned, you can ensure that you are giving your applications the necessary resources to perform well and providing your organization with a solid foundation for future growth.

APPENDIX A – SYSTEM CONFIGURATION INFORMATION

Figure 6 shows the system configuration information for the server we used in our tests.

System	Dell PowerEdge R720
Power supplies	
Total number	2
Vendor and model number	Dell D750E-S1
Wattage of each (W)	750
Cooling fans	
Total number	6
Vendor and model number	AVC DBTC0638B2V
Dimensions (h x w) of each	2.5” x 2.5”
Volts	12
Amps	1.2
General	
Number of processor packages	2
Number of cores per processor	8
Number of hardware threads per core	2
System power management policy	Performance Per Watt (DAPC) or Performance Per Watt (OS) (see text)
CPU	
Vendor	Intel
Name	Xeon
Model number	E5-2690
Stepping	6
Socket type	2011LGA
Core frequency (GHz)	2.90
Bus frequency	100
L1 cache	32 KB I + 32 KB D (per core)
L2 cache	256 KB on chip (per core)
L3 cache	20 MB
Platform	
Vendor and model number	Dell PowerEdge R720
Motherboard model number	OM1GCR
BIOS name and version	Dell 1.5.1
BIOS settings	Default
Memory module(s)	
Total RAM in system (GB)	128
Vendor and model number	Samsung M393B1K70BH1-CH9
Type	PC3-10600
Speed (MHz)	1,333
Speed running in the system (MHz)	1,333
Timing/Latency (tCL-tRCD-tRP-tRASmin)	9-9-9-36
Size (GB)	8

System	Dell PowerEdge R720
Number of RAM module(s)	8
Chip organization	Double-sided
Rank	Dual
Operating system #1	
Name	Red Hat Enterprise Linux 6.4
Filesystem	Ext4
Kernel	2.6.32-358.0.1.el6.x86_64, or 2.6.32-358.1.1.el6.x86_64 (see text)
Language	English
Operating system #2	
Name	Windows Server 2012 Datacenter Edition
Build number	9200
Filesystem	NTFS
Kernel	ACPI x64-based PC
Language	English
Graphics	
Vendor and model number	Matrox® G200eR
Graphics memory (MB)	16
Driver	Matrox Graphics, Inc. 2.3.3.0 (8/19/2011)
RAID controller	
Vendor and model number	Dell PERC H710P Mini
Firmware version	21.1.0-007
Cache size	1 GB
RAID configuration	OS #1: RAID 1 configuration of two type-1 disks. OS #2: RAID 1 configuration of two type-1 disks. Swap/Utility: Raid 1 configuration of twotype-2 disks.
Hard drives type #1	
Vendor and model number	Seagate Savio ST973451SS
Number of drives	4
Size (GB)	73
Buffer size (MB)	16
RPM	15K
Type	SAS
Hard drives type #2	
Vendor and model number	Dell WD300BKHG-18A29V0
Number of drives	2
Size (GB)	300
Buffer size (MB)	16
RPM	10K
Type	SAS

System	Dell PowerEdge R720
Ethernet adapters #1	
Vendor and model number	Intel Gigabit 4P I350-t rNDC
Type	Internal
Ethernet adapters #2	
Vendor and model number	Intel Ethernet Server Adapter X520-SR1
Type	PCIe
Optical drive(s)	
Vendor and model number	TEAC DV-28SW
Type	DVD-ROM
USB ports	
Number	4 external, 1 internal
Type	2.0

Figure 6: Configuration information for our test system.

APPENDIX B – HOW WE TESTED

In this section, we present the step-by-steps for how we tested. First, we cover operating system installation and optimization settings (for optimized solution testing only) for both solutions. Then, we detail how to install and test using each benchmark.

Installing Red Hat Enterprise Linux 6.4

We installed Red Hat Enterprise Linux 6.4 on the first of two logical volumes, configured with two mirrored disks.

1. Insert and boot from the Red Hat Enterprise Linux 6.4 x86_64 installation DVD.
2. At the welcome screen, select Install or upgrade an existing system, and press Enter.
3. At the Media test screen, select Skip, and press Enter.
4. At the Red Hat Enterprise Linux 6 title screen, click Next.
5. At the Choose an Installation Language screen, select English, and click Next.
6. At the Keyboard Type screen, select U.S. English, and click Next.
7. At the Storage Devices screen, select Basic Storage Devices, and click Next.
8. If a warning for device initialization appears, select Yes, discard any data.
9. At the Name the Computer screen, type the host name, and click Configure Network.
10. At the Network Connections screen, select the server's main or management network interface, and click Edit.
11. At the Editing network interface screen, check Connect Automatically.
12. On the same screen, select the IPv4 Settings tab, change the Method to Manual, and click Add.
13. On the same screen, enter the IP address, Netmask, Gateway, and DNS server. Click Apply.
14. Click Close on the Network Connections screen, and click Next on the Name the Computer screen.
15. At the Time zone selection screen, select the appropriate time zone, and click Next.
16. Enter the root password in the Root Password and Confirm fields, and click Next.
17. At the Assign Storage Devices screen, from the list in the left column, select the Linux disk, and click the arrow to copy the device to the right column. Next to the Linux disk, click the Boot radio button, and click Next.
18. At the Partition selection screen, select Replace Existing Linux System(s), and click Next.
19. If a warning appears, click Write changes to disk.
20. At the default installation screen, click Next to begin the installation.
21. At the Congratulations screen, click Reboot.
22. After the system reboots, log in as root.

Installing Microsoft Windows Server 2012 Datacenter

We installed Microsoft Windows Server 2012 Datacenter on the second of two logical volumes, configured with two mirrored disks.

1. Insert and boot from the Windows Server 2012 Datacenter installation DVD.
2. At the first Window Setup screen, keep the defaults for installation language, time/currency format, and keyboard input method. Click Next.
3. At the second Windows Setup screen, click Install now.
4. At the third Windows Setup screen, enter the Windows activation key, and click Next.

5. At the fourth Windows Setup screen, select the Windows Server 2012 Datacenter (Server with a GUI), and click Next.
6. At the fifth Windows Setup screen, select the checkbox to accept the license term, and click Next.
7. At the sixth Windows Setup screen, click Custom: Install Windows only (advanced).
8. At the seventh Windows Setup screen, select Drive 2 as the Windows installation drive, and click Next to start installation.
9. The system will reboot. At the Settings screen, enter the password for the Administrator (twice), and click Finish.

Optimizing operating system configurations

Before running tests for the optimized Red Hat Enterprise Linux 6 configuration, run the following two bash scripts. Red Hat Enterprise Linux 6.4 uses the tuned utility with the throughput-performance profile to configure the filesystems for better performance and to run the CPUs at high performance while the CPU power profile is under OS control. Run the following command to activate this tuned profile.

```
tuned-adm profile throughput-performance
```

DisableSomeDefaultServices.sh

```
#!/bin/bash
## For the optimized-configuration tests, disable unneeded services
for i in abrt-ccpp abrt-oops abrtd acpid atd auditd autofs \
    avahi-daemon cgconfig crond cups haldaemon irqbalance kdump\
    libvirt-guests mcelogd mdmonitor messagebus portreserve\
    postfix rhnsd rhsmcertd rpcbind rpcgssd rpcidmapd certmonger\
    netfs sysstat; do
  service $i stop
done
service lvm2-monitor force-stop
## end of DisableSomeDefaultServices.sh
```

Before running tests for the optimized Windows Server 2012 configuration, run the following batch script. In particular, the OS power profile is set to High performance and the desktop GUI is configured for high performance.

CommandsNoPersonaManagement.bat

```
rem Note: script closely adapted from
rem http://mtellin.com/2010/09/13/creating-a-windows-7-template-for-vmware-
view/
rem Version dated 2012-02-05
reg load "hku\temp" "%USERPROFILE%\..\Default User\NTUSER.DAT"
reg ADD "hku\temp\Software\Policies\Microsoft\Windows\Control Panel\Desktop"
/v SCRNSAVE.EXE /d "%windir%\system32\scrnsave.scr" /f
reg ADD "hku\temp\Software\Policies\Microsoft\Windows\Control Panel\Desktop"
/v ScreenSaveTimeOut /d "600" /f
reg ADD "hku\temp\Software\Policies\Microsoft\Windows\Control Panel\Desktop"
/v ScreenSaverIsSecure /d "1" /f
reg ADD "hku\temp\Software\Microsoft\Windows\CurrentVersion\Policies\System"
/v Wallpaper /d " " /f
reg ADD "hku\temp\Software\Microsoft\Windows\CurrentVersion\Internet
Settings\Cache" /v Persistent /t REG_DWORD /d 0x0 /f
```

```
reg ADD "hku\temp\Software\Microsoft\Feeds" /v SyncStatus /t REG_DWORD /d
0x0 /f
reg ADD
"hku\temp\Software\Microsoft\WIndows\CurrentVersion\Policies\Explorer" /v
HideSCAHealth /t REG_DWORD /d 0x1 /f
reg unload "hku\temp"

reg ADD "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Internet
Explorer\Main" /v DisableFirstRunCustomize /t REG_DWORD /d 0x1 /f
reg ADD "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management\PrefetchParameters" /v EnableSuperfetch /t
REG_DWORD /d 0x0 /f
reg ADD
"HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU" /v
NoAutoUpdate /t REG_DWORD /d 0x1 /f
reg ADD "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows
NT\SystemRestore" /v DisableSR /t REG_DWORD /d 0x1 /f
reg ADD "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Disk" /v
TimeOutValue /t REG_DWORD /d 200 /f
reg ADD "HKEY_LOCAL_MACHINE\SOFTWARE\Image" /v Revision /t REG_SZ /d 1.0 /f

reg ADD "HKEY_LOCAL_MACHINE\SOFTWARE\Image" /v Virtual /t REG_SZ /d Yes /f
reg ADD
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\eventlog\Application"
/v MaxSize /t REG_DWORD /d 0x100000 /f
reg ADD
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\eventlog\Application"
/v Retention /t REG_DWORD /d 0x0 /f
reg ADD
"HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Network\NewNetworkWindo
wOff" /f
reg ADD
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\eventlog\System" /v
MaxSize /t

REG_DWORD /d 0x100000 /f
reg ADD
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\eventlog\System" /v
Retention /t REG_DWORD /d 0x0 /f
reg ADD
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\eventlog\Security" /v
MaxSize /t REG_DWORD /d 0x100000 /f
reg ADD
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\eventlog\Security" /v
Retention /t REG_DWORD /d 0x0 /f
reg ADD "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\CrashControl"
/v CrashDumpEnabled /t REG_DWORD /d 0x0 /f
reg ADD
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explo
rer" /v NoRecycleFiles /t REG_DWORD /d 0x1 /f
reg ADD "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server" /v fDenyTSConnections /t REG_DWORD /d 0x0 /f
```

```
reg ADD "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-Tcp" /v UserAuthentication /t REG_DWORD /d 0x0 /f
reg ADD
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\policies\syste
m" /v EnableLUA /t REG_DWORD /d 0x0 /f
reg Add "HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Sideshow" /v
Disabled /t REG_DWORD /d 0x1 /f

Powershell Set-Service 'BDESVC' -startuptype "disabled"
Powershell Set-Service 'wbengine' -startuptype "disabled"
Powershell Set-Service 'DPS' -startuptype "disabled"
Powershell Set-Service 'UxSms' -startuptype "disabled"
Powershell Set-Service 'Defragsvc' -startuptype "disabled"
Powershell Set-Service 'HomeGroupListener' -startuptype "disabled"
Powershell Set-Service 'HomeGroupProvider' -startuptype "disabled"
Powershell Set-Service 'iphlpsvc' -startuptype "disabled"
Powershell Set-Service 'MSiSCSI' -startuptype "disabled"
Powershell Set-Service 'swprv' -startuptype "disabled"
Powershell Set-Service 'CscService' -startuptype "disabled"
Powershell Set-Service 'SstpSvc' -startuptype "disabled"
Powershell Set-Service 'wscsvc' -startuptype "disabled"
Powershell Set-Service 'SSDPSRV' -startuptype "disabled"
Powershell Set-Service 'SysMain' -startuptype "disabled"
Powershell Set-Service 'TabletInputService' -startuptype "disabled"
Powershell Set-Service 'Themes' -startuptype "disabled"
Powershell Set-Service 'upnphost' -startuptype "disabled"
Powershell Set-Service 'VSS' -startuptype "disabled"
Powershell Set-Service 'SDRSVC' -startuptype "disabled"
Powershell Set-Service 'WinDefend' -startuptype "disabled"
Powershell Set-Service 'WerSvc' -startuptype "disabled"
Powershell Set-Service 'MpsSvc' -startuptype "disabled"
Powershell Set-Service 'ehRecvr' -startuptype "disabled"
Powershell Set-Service 'ehSched' -startuptype "disabled"
Powershell Set-Service 'WSearch' -startuptype "disabled"
Powershell Set-Service 'wuauserv' -startuptype "disabled"
Powershell Set-Service 'Wlansvc' -startuptype "disabled"
Powershell Set-Service 'WwanSvc' -startuptype "disabled"
bcdedit /set BOOTUX disabled
vssadmin delete shadows /All /Quiet
Powershell disable-computerrestore -drive c:\
netsh advfirewall set allprofiles state off
powercfg -H OFF
powercfg -setactive 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c
net stop "sysmain"
fsutil behavior set DisableLastAccess 1
schtasks /change /TN "\Microsoft\Windows\Defrag\ScheduledDefrag" /Disable
schtasks /change /TN "\Microsoft\Windows\SystemRestore\SR" /Disable
schtasks /change /TN "\Microsoft\Windows\Registry\RegIdleBackup" /Disable
schtasks /change /TN "\Microsoft\Windows Defender\MPIdleTask" /Disable
schtasks /change /TN "\Microsoft\Windows Defender\MP Scheduled Scan"
/Disable
schtasks /change /TN "\Microsoft\Windows\Maintenance\WinSAT" /Disable
```

```
rem End of CommandsNoPersonaManagement.bat
```

TESTING WITH SPEC CPU2006

Installing the SPEC CPU2006 software on the Red Hat Enterprise Linux 6.4 operating system

We used version 1.2 of the SPEC CPU2006 distribution with binaries, run-time libraries and run-time options taken from the Intel Linux 64-bix distribution (`cpu2006.1.2.ic13.linux64.18feb2013.binaries.tar.xz`), which uses the Intel ic13 compiler. Specifically, we used configuration for the x64 “AVX” Intel processor-type for the rate and speed SPEC CPU runs.

1. Log onto Red Hat Enterprise Linux 6.4.
2. Create a scratch partition for SPEC CPU’s intermediate results on an unused volume (here /dev/sdb):
   ```
   parted /dev/sdb mklabel gpt
   parted /dev/sdb mkpart primary "1 -1"
   parted /dev/sdb name 1 scratch
   mkfs.ext4 /dev/sdb1
   mkdir /scratch
   ```
3. Configure the system to mount the scratch partition at directory/scratch. Edit the file /etc/fstab and add the following lines to the end.
   ```
   ## Scratch space for SPEC CPU
   /dev/sdb1 /scratch ext4 defaults 0 0
   ```
4. Mount the scratch partition.
   ```
   mount /scratch
   ```
5. Install the GCC compiler as well as the run-time libraries for 32-bit programs by adding the following packages: gcc , libc.i686, libgcc.i686, libstdc++.i686, and glibc-devel.i686.
6. Create a local SPEC CPU directory.
   ```
   mkdir /opt/cpu2006
   ```
7. Mount the SPEC CPU2006 v1.2 installation CD or ISO image to `/mnt`
8. Run the SPEC CPU2006 installation script from the installation directory. Answer `Y` to both questions.
   ```
   cd /mnt
   sh install.sh /opt/cpu2006
   ```
9. Extract the Intel 18feb2013-linux64 binaries, libraries and scripts from the cpu2006.1.2.ic13.linux64.18feb2013.binaries.tar.xz archive into /opt/cpu2006.
   ```
   cd /opt/cpu2006
   tar Jxf cpu2006.1.2.ic13.linux64.18feb2013.binaries.tar.xz
   ```
10. Modify file permissions for the SPEC CPU executables and shared libraries.
    ```
    cd /opt/cpu2006
    find benchspec/CPU2006 -name "*cpu2006.1.2.ic13.linux64.*27dec2012" |\
         xargs chmod 755
    find sh libs -name "*.so" | xargs chmod 755
    ```
11. Modify the run-time configuration files so that SPEC CPU uses the scratch directory for intermediate results. Go to directory `/opt/cpu2006/config` and add the following line to both `cpu2006.1.2.ic13.linux64.avx.speed.27dec2012.cfg` and `cpu2006.1.2.ic13.linux64.ws.avx.rate.27dec2012.cfg` after the line `reportable=1`.
    ```
         output_root=/scratch
    ```

12. Copy the contents of the SPEC CPU executables directory to `/scratch`.

```
rm -r /scratch/benchspec/
cp -pr /opt/cpu2006/benchspec /scratch/benchspec
```

Running the SPEC CPU2006 tests on the Red Hat Enterprise Linux 6.4 operating system

We used Intel's batch script, to run the SPEC CPU2006 FP and INT rate and speed tests.

/opt/cpu2006/official-ws-avx-smt-on-rate.sh

```
. ./shrc
. ./numa-detection.sh
ulimit -s unlimited
a=`cat /proc/cpuinfo | grep processor | wc -l`
rm -rf topo.txt
specperl nhmtopology.pl
b=`cat topo.txt`
c=`expr $a / 2`
echo "*****************************************************************"
echo Running rate with $a copies on an SSE4.2 system with a topology of $b
echo "*****************************************************************"
if [ $NUMA == 0 ]
then
  if [ -e /sys/kernel/mm/redhat_transparent_hugepage/enabled ]
  then
    echo always > /sys/kernel/mm/redhat_transparent_hugepage/enabled
    runspec --define external --rate $a -c
cpu2006.1.2.ic13.linux64.ws.avx.rate.27dec2012 --define smt --define
cores=$c --define $b  --define no-numa --define THP_enabled  -o all int
    runspec --define external --rate $a -c
cpu2006.1.2.ic13.linux64.ws.avx.rate.27dec2012 --define smt --define
cores=$c --define $b  --define no-numa --define THP_enabled -o all fp
  else
    runspec --define external --rate $a -c
cpu2006.1.2.ic13.linux64.ws.avx.rate.27dec2012 --define smt --define
cores=$c --define $b  --define no-numa  -o all int
    runspec --define external --rate $a -c
cpu2006.1.2.ic13.linux64.ws.avx.rate.27dec2012 --define smt --define
cores=$c --define $b  --define no-numa  -o all fp
  fi
else
  echo 1 >       /proc/sys/vm/drop_caches
  if [ -e /sys/kernel/mm/redhat_transparent_hugepage/enabled ]
  then
    echo always > /sys/kernel/mm/redhat_transparent_hugepage/enabled
    numactl --interleave=all runspec --define external --rate $a -c
cpu2006.1.2.ic13.linux64.ws.avx.rate.27dec2012 --define smt --define
cores=$c --define $b --define invoke_with_interleave --define drop_caches --
define THP_enabled   -o all int
    numactl --interleave=all runspec --define external --rate $a -c
cpu2006.1.2.ic13.linux64.ws.avx.rate.27dec2012 --define smt --define
cores=$c --define $b --define invoke_with_interleave --define drop_caches --
define THP_enabled  -o all fp
  else
```

```
    numactl --interleave=all runspec --define external --rate $a -c
cpu2006.1.2.ic13.linux64.ws.avx.rate.27dec2012 --define smt --define
cores=$c --define $b --define invoke_with_interleave --define drop_caches -o
all int
    numactl --interleave=all runspec --define external --rate $a -c
cpu2006.1.2.ic13.linux64.ws.avx.rate.27dec2012 --define smt --define
cores=$c --define $b --define invoke_with_interleave --define drop_caches -o
all fp
  fi
fi
```

Installing the SPEC CPU2006 software on the Windows Server 2012 operating system

We used version 1.2 of the SPEC CPU2006 distribution with binaries, run-time libraries and run-time options taken from the Intel Windows 64-bit distribution (`cpu2006.1.2.ic13.sep2012.winx64.binaries.zip`), which uses the Intel ic13 compiler. Specifically, we used configuration for the x64 “AVX” Intel processor-type for the rate and speed SPEC CPU runs.

1. Log in as administrator to create a scratch partition for SPEC CPU’s intermediate results.
2. Open the Server Manager.
3. Select File and Storage Services.
4. Select Disks.
5. From the list of disks, right-click on your Scratch device, select Reset Disk, and click Yes to erase the data.
6. From the list of disks, right-click on your Scratch device, and select New Volume…
7. On the Server and Disk screen, select the disk, and click Next.
8. Click OK to initialize the disk with a GPT label.
9. On the Size screen, keep the defaults, and click Next.
10. On the Drive Letter or Folder screen, Select The following folder, and enter c:\scratch. Click OK to create this folder.
11. On the File System Settings screen, select NTFS.
12. On the same screen, enter a Volume label of Scratch, and click Next.
13. On the Confirmation screen, click Create.
14. Close the Server Manager.
15. Restart the server.
16. Create a local SPEC CPU directory.
```
mkdir c:\cpu2006
```
17. Mount the SPEC CPU2006 v1.2 installation CD or ISO image as drive `d:`
18. Run the SPEC CPU2006 installation script from the installation directory. Answer `Y` to both questions.
```
cd d:\
install.bat c:\cpu2006
```
19. Extract the Intel sep2102-winx64 binaries, libraries and scripts from the `cpu2006.1.2.ic13.sep2012.winx64.binaries.zip archive into c:\cpu2006.`
20. Modify the run-time configuration files so that SPEC CPU uses the scratch directory for intermediate results. Go to directory `c:\cpu2006\config` and add the following line to both `cpu2006.1.2.ic13.0.winx64.avx.speed.29sep2012.cfg and`

`cpu2006.1.2.ic13.0.winx64.ws.avx.rate.29sep2012.cfg` after the line `reportable=1.`

```
output_root=/scratch
```

21. Copy the contents of directory `c:\cpu2006\benchspec` to `d:\scratch\benchspec`.

Running the SPEC CPU2006 tests on the Windows Server 2012 operating system

We used Intel's batch script, to run the SPEC CPU2006 FP and INT rate and speed tests.

c:\cpu2006\official-ic13.0-avx-workstation-winx64-smt-on.bat

```
call shrc.bat
set /a numcores=(%NUMBER_OF_PROCESSORS%)/2
set OMP_NUM_THREADS=%numcores%
set KMP_AFFINITY=granularity=fine,scatter
call runspec -c cpu2006.1.2.ic13.0.winx64.avx.speed.29sep2012.cfg -T all -o
asc --flagsurl=Intel-ic13.0-official-windows.xml int
call runspec -c cpu2006.1.2.ic13.0.winx64.avx.speed.29sep2012.cfg -T all -o
asc --flagsurl=Intel-ic13.0-official-windows.xml fp

set OMP_NUM_THREADS=
set KMP_AFFINITY=
call runspec --rate %NUMBER_OF_PROCESSORS% --define smt-on --define
cores=%numcores% -c cpu2006.1.2.ic13.0.winx64.ws.avx.rate.29sep2012.cfg -T
all -o asc --flagsurl=Intel-ic13.0-official-windows.xml int
call runspec --rate %NUMBER_OF_PROCESSORS% --define smt-on --define
cores=%numcores% -c cpu2006.1.2.ic13.0.winx64.ws.avx.rate.29sep2012.cfg -T
all -o asc --flagsurl=Intel-ic13.0-official-windows.xml fp

call runspec --loose -n 1 --rate 1 --define smt-on --define cores=%numcores%
-c cpu2006.1.2.ic13.0.winx64.ws.avx.rate.29sep2012.cfg -T base -o asc int
call runspec --loose -n 1 --rate 1 --define smt-on --define cores=%numcores%
-c cpu2006.1.2.ic13.0.winx64.ws.avx.rate.29sep2012.cfg -T base -o asc fp
```

TESTING WITH LINPACK

Installing LINPACK benchmark on the Red Hat Enterprise Linux 6.4 operating system

We used Intel's binary, version 11.0.3 for Linux x64, for the LINPACK benchmark.

1. Log onto Red Hat Enterprise Linux 6.4.
2. Uncompress the archive.
   ```
   tar zxf l_lpk_p_11.0.3.008.tgz
   ```
3. The benchmark code is in directory `linpack_11.0.3/benchmarks/linpack`.
   ```
   cd linpack_11.0.3/benchmarks/linpack
   ```

Running the LINPACK benchmark on the Red Hat Enterprise Linux 6.4 operating system

We adapted Intel's sample batch script and used its unmodified input data to run the LINPACK benchmark.

run_linpack.sh

```
#!/bin/bash
# Adapted from Intel's sample run script
#
# Run parameters are read from lininput_xeon64

#    Setting up affinity for better threading performance
```

```
MY_KMP_AFFINITY=nowarnings,granularity=fine,verbose,explicit,proclist=
p[1]='0'
p[2]='0,1'
p[4]='0,2,1,3'
p[8]='0,2,4,6,1,3,5,7'
p[16]='0,2,4,6,8,10,12,14,1,3,5,7,9,11,13,15'
p[32]='0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,2
6,27,28,29,30,31'

arch=xeon64

for i in 1 2 4 8 16 32; do
  echo 3 > /proc/sys/vm/drop_caches
  echo Threads = $i
  export OMP_NUM_THREADS=$i
  export KMP_AFFINITY="${MY_KMP_AFFINITY}[${p[$i]}]"
  echo "Affinity = $KMP_AFFINITY"
  date
  ./xlinpack_$arch lininput_$arch
  echo -n "Done: "
  date
done | tee -a lin_$arch-$1.txt
## end of shell script run_linpack.sh
```

lininput_xeon64

```
Sample Intel(R) Optimized LINPACK Benchmark data file (lininput_xeon64)
Intel(R) Optimized LINPACK Benchmark data
15                    # number of tests
1000 2000 5000 10000 15000 18000 20000 22000 25000 26000 27000 30000 35000
40000 45000 # problem sizes
1000 2000 5008 10000 15000 18008 20016 22008 25000 26000 27000 30000 35000
40000 45000 # leading dimensions
4 2 2 2 2 2 2 2 2 2 1 1 1 1 1 # times to run a test
4 4 4 4 4 4 4 4 4 4 4 1 1 1 1 # alignment values (in KBytes)
```

Installing the LINPACK benchmark on the Windows Server 2012 operating system

To run the LINPACK benchmark, we adapted Intel's sample batch script and modified the Windows input data to match that used in the Red Hat Linux tests.

1. Log onto Windows Server 2012.
2. Uncompress the archive.
   `unzip w_lpk_p_11.0.3.008.zip`
3. The benchmark code is in directory `linpack_11.0.3/benchmarks/linpack`.
   `cd w_lpk_p_11.0.3.008\linpack_11.0.3\benchmarks\linpack`

Running the LINPACK benchmark on the Windows Server 2012 operating system

We used the following batch script and input data to run the LINPACK benchmark.

runme_xeon64.bat

```
@echo off

:: echo Adapted from Intel's sample run script
```

```
SETLOCAL EnableDelayedExpansion

rem Argument 1 is the tag for the run's output file
set fff="LINPACK_%1%.txt"

rem Setting up BASE affinity for better threading performance
set MY_KMP_AFFINITY=nowarnings,granularity=fine,verbose,explicit,proclist=
rem Setting path to OpenMP library
set PATH=..\..\..\redist\intel64\compiler;%PATH%

rem My processor affinities
set p1=0
set p2=0,16
set p4=0,2,16,18
set p8=0,2,4,6,16,18,20,22
set p16=0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30
set
p32=0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,2
7,28,29,30,31
date /t
time /t

echo Running linpack_xeon64.exe. Output could be found in %fff%.
for %%i in (1 2 4 8 16 32) do (
  echo %%i
  :: echo !p%%i!
  set OMP_NUM_THREADS=%%i
  set KMP_AFFINITY=%MY_KMP_AFFINITY%[!p%%i!]
  echo !KMP_AFFINITY!
  linpack_xeon64.exe new-lininput_xeon64 >> %fff%
  date /t >> %fff%
  time /t >> %fff% )

echo    Done:
date /t
time /t

ENDLOCAL
@echo on
rem End of batch script runme_xeon64.bat
```

new-lininput_xeon64

```
Modified Intel(R) Optimized LINPACK Benchmark data file (lininput_xeon64)
Intel(R) Optimized LINPACK Benchmark data
15   # number of tests
1000 2000 5000 10000 15000 18000 20000 22000 25000 26000 27000 30000 35000
40000
 45000 # problem sizes
1000 2000 5008 10000 15000 18008 20016 22008 25000 26000 27000 30000 35000
40000
 45000 # leading dimensions
4 2 2 2 2 2 2 2 2 2 1 1 1 1 1 # times to run a test
4 4 4 4 4 4 4 4 4 4 4 1 1 1 1 # alignment values (in KBytes)
```

TESTING WITH STREAM

Installing the Stream software on the Red Hat Enterprise Linux 6.4 operating system

We used version 5.10 of the Stream distribution from http://www.streambench.org/ and compiled it with the Intel ic13 compiler for Linux.

1. Log onto Red Hat Enterprise Linux 6.4.
2. Uncompress the source-code archive into the directory stream-5.10.
3. Create s_f, the foreground version of Stream (10 iterations), with the Intel C compiler:
   `icl -O3 -xAVX _DNTIMESS=10  stream.c -o s_f`
4. Create s_b, the foreground version of Stream (100 iterations), with the Intel C compiler:
5. `icl -O3 -xAVX _DNTIMESS=100 stream.c -o s_b`

Running the Stream tests on the Red Hat Enterprise Linux 6.4 operating system

In order to perform the multi-core tests, we followed the guidance from the Stream documentation, and used multiple copies

We used Intel's batch script to run the Stream FP and INT rate and speed tests.

run_stream.sh

```
#!/bin/bash


exec 2> /dev/null
date
pkill -9 s_f > /dev/null 2>&1
pkill -9 s_b > /dev/null 2>&1
sleep 3
for size in 1 2 4 8 16 32; do
   echo 3 > /proc/sys/vm/drop_caches
   printf '\n============================================\n'

   printf "Doing $size\n"

   printf "Launching "
   if  [ $size -ne 1 ] ; then
      for i in $(seq $(($size-1)) ); do
         printf "$i($(($i % 2))) "

        numactl --membind $(($i % 2)) --physcpubind $i \
            nice -n -10 ./s_b > /dev/null 2>& 1 &
      done
   fi

   sleep 3
   printf "0(0)\n"
   numactl --membind 0 --physcpubind 0 \
        nice -n -10 ./s_f

   pkill -9 s_f > /dev/null 2>&1
   pkill -9 s_b > /dev/null 2>&1
   wait
```

```
done

printf '==============================================\n\n'
date
exit 0
## end of shell script run_stream.sh
```

Installing the Stream software on the Windows Server 2012 operating system

We used version 5.10 of the Stream distribution from http://www.cs.virginia.edu/stream/FTP/Code/ and compiled it with the Intel ic13 compiler for Windows under the Microsoft Studio environment. We also installed the psutil tools from http://sysinternals.com. We replaced the Unix/Linux-style timer with one appropriate for Windows (see the patch-file, below, for details).

1. Create a local Stream directory.
   ```
   mkdir stream-5.10
   ```
2. Copy the Stream 5.10 source files into this directory.
3. We modified the source code for stream.c,version 5.10, to use a Windows-style routine to measure time. The patch file, stream.c-diff, for the modifications follows. You can apply the modification with the Unix/Linux command:
   ```
   patch < stream.c-diff
   ```

stream.c-diff

```
44c44
< //# include <unistd.h>
---
> # include <unistd.h>
48,73c48
< //# include <sys/time.h>
< #include <windows.h>
< typedef int ssize_t;
<
< //////////////////
< double PCFreq = 0.0;
< __int64 CounterStart = 0;
<
< void StartCounter()
< {
<     LARGE_INTEGER li;
<     if(!QueryPerformanceFrequency(&li))
<         printf("QueryPerformanceFrequency failed!\n");
<
<     PCFreq = (double)(li.QuadPart);
<
<     QueryPerformanceCounter(&li);
<     CounterStart = li.QuadPart;
< }
< double mysecond()
< {
```

```
<     LARGE_INTEGER li;
<     QueryPerformanceCounter(&li);
<     return (double)(li.QuadPart-CounterStart)/PCFreq;
< }
< //////////////////
---
> # include <sys/time.h>
133c108
< #   define NTIMES     100
---
> #   define NTIMES     10
242d216
<         StartCounter();
441,442d414
< #if 0
<
457d428
< #endif
```

4. Create s_f.exe, the foreground version of Stream (10 iterations), with the Intel C compiler.
   ```
   icl /O3 /QxAVX -DNTIMES=10 stream.c -o s_f.exe
   ```
5. Create s_b.exe, the foreground version of Stream (100 iterations), with the Intel C compiler.
   ```
   icl /O3 /QxAVX -DNTIMES=100 stream.c -o s_f.exe
   ```

Running the Stream tests on the Windows Server 2012 operating system

We used Intel's batch script to run the Stream FP and INT rate and speed tests.

stream_start.bat

```
@echo off
set fff="NUMA_%1%.txt"
echo %fff%
del %fff% > NUL 2>&1

rem ############################################
rem 1
pskill s_b.exe > NUL 2>&1  &pskill s_f.exe > NUL 2>&1
choice /t 5 /c yn /d y > NUL 2>&1
echo ""  >> %fff%
echo "1" >> %fff%
echo 1

call fg-stream >> %fff%

rem #############################################
rem 2
pskill s_b.exe > NUL 2>&1  & pskill s_f.exe > NUL 2>&1
choice /t 5 /c yn /d y > NUL 2>&1

echo ""  >> %fff%
echo "2" >> %fff%
```

```
echo 2

start /high /node 1 /affinity 0x0001 /b s_b.exe > NUL
choice /t 20 /c yn /d y > NUL 2>&1
call fg-stream >> %fff%

rem ############################################
rem 4
pskill s_b.exe > NUL 2>&1  & pskill s_f.exe > NUL 2>&1
choice /t 5 /c yn /d y > NUL 2>&1

echo ""  >> %fff%
echo "4" >> %fff%
echo 4

start /high /node 1 /affinity 0x0001 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0002 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0002 /b s_b.exe > NUL
choice /t 20 /c yn /d y > NUL 2>&1
call fg-stream >> %fff%

rem ############################################
rem 8
pskill s_b.exe > NUL 2>&1  & pskill s_f.exe > NUL 2>&1
choice /t 5 /c yn /d y > NUL 2>&1

echo ""  >> %fff%
echo "8" >> %fff%
echo 8

start /high /node 1 /affinity 0x0001 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0002 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0002 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0004 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0004 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0008 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0008 /b s_b.exe > NUL
choice /t 20 /c yn /d y > NUL 2>&1
call fg-stream >> %fff%

rem ############################################
rem 16
pskill s_b.exe > NUL 2>&1  & pskill s_f.exe > NUL 2>&1
choice /t 5 /c yn /d y > NUL 2>&1

echo ""  >> %fff%
echo "16" >> %fff%
echo 16

start /high /node 1 /affinity 0x0001 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0002 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0002 /b s_b.exe > NUL
```

```
start /high /node 0 /affinity 0x0004 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0004 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0008 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0008 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0010 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0010 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0020 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0020 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0040 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0040 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0080 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0080 /b s_b.exe > NUL
choice /t 20 /c yn /d y > NUL 2>&1
call fg-stream >> %fff%

rem ###############################################
rem 32
pskill s_b.exe > NUL 2>&1  & pskill s_f.exe > NUL 2>&1
choice /t 5 /c yn /d y > NUL 2>&1

echo ""  >> %fff%
echo "32" >> %fff%
echo 32

start /high /node 1 /affinity 0x0001 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0002 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0002 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0004 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0004 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0008 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0008 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0010 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0010 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0020 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0020 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0040 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0040 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0080 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0080 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0100 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0100 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0200 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0200 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0400 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0400 /b s_b.exe > NUL
start /high /node 0 /affinity 0x0800 /b s_b.exe > NUL
start /high /node 1 /affinity 0x0800 /b s_b.exe > NUL
start /high /node 0 /affinity 0x1000 /b s_b.exe > NUL
start /high /node 1 /affinity 0x1000 /b s_b.exe > NUL
start /high /node 0 /affinity 0x2000 /b s_b.exe > NUL
start /high /node 1 /affinity 0x2000 /b s_b.exe > NUL
start /high /node 0 /affinity 0x4000 /b s_b.exe > NUL
```

```
start /high /node 1 /affinity 0x4000 /b s_b.exe > NUL
start /high /node 0 /affinity 0x8000 /b s_b.exe > NUL
start /high /node 1 /affinity 0x8000 /b s_b.exe > NUL
choice /t 20 /c yn /d y > NUL 2>&1
call fg-stream >> %fff%

rem ############################################
pskill s_b.exe > NUL 2>&1  & pskill s_f.exe > NUL 2>&1

echo DONE
rem End of batch script stream_start.bat
```

fg_stream.bat

```
@echo off
start /high /node 0 /affinity 0x0001 /b /w s_f.exe
rem End of batch script fg_stream.bat
```

APPENDIX C – DETAILED TEST RESULTS

Figure 7 presents the SPEC CPU2006 results in detail.

	Red Hat Enterprise Linux 6		Microsoft Windows Server 2012		Red Hat advantage	
Score	**Out-of-box**	**Optimized**	**Out-of-box**	**Optimized**	**Out-of-box**	**Optimized**
SPECint_rate2006	639	640	618	619	3.4%	3.4%
SPECint_rate_base2006	623	621	593	591	5.1%	5.1%
SPECfp_rate2006	422	422	406	405	3.9%	4.2%
SPECfp_rate_base2006	408	404	399	397	2.3%	1.8%

Figure 7: SPEC CPU2006 floating point and integer performance results for the two solutions.

Figures 8 and 9 present the detailed results for our LINPACK tests, taking the geometric mean at various thread counts.

LINPACK results, in GFlops, for out-of-box configurations			
Number of threads	**Red Hat Enterprise Linux 6**	**Microsoft Windows Server 2012**	**Red Hat advantage**
1	25.5	24.8	2.8%
2	49.9	47.7	4.6%
4	86.0	87.8	-2.1%
8	147.3	153.6	-4.1%
16	275.0	249.8	10.1%

Figure 8: LINPACK results, in GFlops (10^9 floating-point operations per second), for out-of-box-configurations.

LINPACK results, in GFlops, for optimized configurations			
Number of threads	**Red Hat Enterprise Linux 6**	**Microsoft Windows Server 2012**	**Red Hat advantage**
1	27.2	26.6	2.3%
2	50.9	49.7	2.4%
4	97.4	93.9	3.7%
8	173.3	166.0	4.4%
16	291.9	253.6	15.1%

Figure 9: LINPACK results, in GFlops (10^9 floating-point operations per second), for optimized-configurations.

Figures 10 and 11 present the detailed results for our STREAM tests.

STREAM results for out-of-box configurations: Memory bandwidth, in MB/s				
Number of threads	**STREAM sub-test**	**Red Hat Enterprise Linux 6**	**Microsoft Windows Server 2012**	**Red Hat advantage**
1	Copy	7,705.1	8,968.7	-14.1%
	Scale	8,230.0	8,149.8	1.0%
	Add	10,596.2	10,426.7	1.6%
	Triad	10,665.1	10,481.5	1.8%
2	Copy	17,476.6	20,058.6	-12.9%
	Scale	18,442.4	18,171.8	1.5%
	Add	23,530.6	22,858.2	2.9%
	Triad	23,687.6	22,994.0	3.0%
4	Copy	32,366.4	24,843.6	30.3%
	Scale	34,544.8	22,663.2	52.4%
	Add	44,124.4	28,963.6	52.3%
	Triad	44,431.2	28,710.4	54.8%
8	Copy	57,396.8	44,924.0	27.8%
	Scale	61,013.6	43,011.2	41.9%
	Add	67,405.6	54,555.2	23.6%
	Triad	68,437.6	54,390.4	25.8%
16	Copy	59,382.4	69,147.2	-14.1%
	Scale	69,662.4	61,038.4	14.1%
	Add	75,806.4	74,046.4	2.4%
	Triad	66,822.4	68,928.0	-3.1%

Figure 10: STREAM memory-performance results for the out-of-box configurations.

STREAM results for optimized configurations: Memory bandwidth, in MB/s				
Number of threads	**STREAM sub-test**	**Red Hat Enterprise Linux 6**	**Microsoft Windows Server 2012**	**Red Hat advantage**
1	Copy	7,662.4	8,989.6	-14.8%
	Scale	8,848.1	8,155.1	8.5%
	Add	10,790.4	10,440.6	3.4%
	Triad	10,773.3	10,499.4	2.6%
2	Copy	15,299.8	20,138.8	-24.0%
	Scale	17,685.8	18,247.0	-3.1%

STREAM results for optimized configurations: Memory bandwidth, in MB/s				
Number of threads	STREAM sub-test	Red Hat Enterprise Linux 6	Microsoft Windows Server 2012	Red Hat advantage
	Add	21,571.6	23,015.2	-6.3%
	Triad	21,532.8	23,123.4	-6.9%
4	Copy	28,837.6	24,936.8	15.6%
	Scale	34,387.6	22,372.4	53.7%
	Add	41,919.2	29,046.4	44.3%
	Triad	41,649.2	29,260.0	42.3%
8	Copy	50,116.8	44,916.8	11.6%
	Scale	56,096.0	43,095.2	30.2%
	Add	70,109.6	55,236.0	26.9%
	Triad	75,199.2	55,215.2	36.2%
16	Copy	64,456.0	69,929.6	-7.8%
	Scale	68,908.8	63,760.0	8.1%
	Add	69,712.0	73,545.6	-5.2%
	Triad	73,276.8	69,324.8	5.7%

Figure 11: STREAM memory-performance results for the optimized configurations.

ABOUT PRINCIPLED TECHNOLOGIES

Principled Technologies, Inc.
1007 Slater Road, Suite 300
Durham, NC, 27703
www.principledtechnologies.com

We provide industry-leading technology assessment and fact-based marketing services. We bring to every assignment extensive experience with and expertise in all aspects of technology testing and analysis, from researching new technologies, to developing new methodologies, to testing with existing and new tools.

When the assessment is complete, we know how to present the results to a broad range of target audiences. We provide our clients with the materials they need, from market-focused data to use in their own collateral to custom sales aids, such as test reports, performance assessments, and white papers. Every document reflects the results of our trusted independent analysis.

We provide customized services that focus on our clients' individual requirements. Whether the technology involves hardware, software, Web sites, or services, we offer the experience, expertise, and tools to help our clients assess how it will fare against its competition, its performance, its market readiness, and its quality and reliability.

Our founders, Mark L. Van Name and Bill Catchings, have worked together in technology assessment for over 20 years. As journalists, they published over a thousand articles on a wide array of technology subjects. They created and led the Ziff-Davis Benchmark Operation, which developed such industry-standard benchmarks as Ziff Davis Media's Winstone and WebBench. They founded and led eTesting Labs, and after the acquisition of that company by Lionbridge Technologies were the head and CTO of VeriTest.