Paper 375-2012

Utilize Red Hat Kernel Virtual Machine to enable your Development and Test Environment to Peacefully Co-Exist with your Production Environment

Bob Augustine, Hewlett-Packard, Downers Grove, IL, USA

Barry Marson, Red Hat, Westford, MA, USA

ABSTRACT

As x86–64-based servers, such as the HP ProLiant, have become less expensive, increases in performance have enabled businesses to seek ways to more effectively utilize **THEM WITHIN THEIR ENTERPRISES.**

Another industry trend has created the need for businesses to be able to further subdivide larger proprietary servers that have utilized using hard and **VIRTUAL PARTITION TECHNOLOGY.**

In order for a migration from these proprietary architectures to be viable, there is a need to offer a migration methodology for virtualization technologies to be made a**VAILABLE ON COMMODITY HARDWARE.**

Because of SAS® software's unique I/O footprint methods of virtualization utilizing shared resources, typically enabled via a **HYPERVISOR HAVE BEEN AVOIDED.**

INTRODUCTION

HP and Red Hat decided to collaborate on this proof of concept (POC) because we had a number of customers asking if running a SAS® production environment on bare metal along with the development and test environments running on top of RHEL KVM was a viable alternative to having multiple servers, each isolated and unable to share resources between environments.

As customers are seeking to migrate SAS from older, proprietary environments to less expensive servers running operating systems, such as Red Hat Enterprise Linux, one of the limiting factors has been the ability to replace hard partitioning technologies available on those older environments with viable alternatives.

Since today, there are no partitioning

[1] technologies available for Intel® based x86-64 servers, the only alternative is virtualization[2] technologies.

Traditionally a virtualization environment for SAS software in a production environment has been discouraged; rather the use of virtualized environments was only considered for development, test and to a lesser degree QA environments.

SAS's software places a high demand on I/O (input/output). In order to effectively utilize virtualization technologies, those technologies need to be sufficiently light weight with respect to how they impact the ability to deliver I/O from the storage devices through the operating system's to the user's data space.

The purpose of this POC was to recreate a use case which would have the SAS production instance running on the host operating system at the same time the test and development instances were running on guests [3]. We wanted to demonstrate that having multiple guests running simultaneously would have minimal impact on the host environment.

The results confirm that running SAS production instance on bare metal along with the test and development environments on top of RH KVM is a viable alternative to hard and soft partitioning technologies.

IMPORTANT

This POC provides a reference for comparing hardware and software products. It is not intended to be used as a sizing guideline. In the real world server performance is highly dependent upon the application design and workload profiling.

While the tests simulated realistic SAS production workloads, as with any laboratory testing, the performance metrics quoted in this paper are idealized. In a production environment, these metrics may be impacted by a variety of factors.

As a matter of best practice for all deployments, HP recommends implementing a proof-of-concept using a test environment that matches as closely as possible the planned production environment. In this way appropriate performance and scalability characterizations can be obtained. For help with a proof-of-concept contact an HP SAS technical specialist by sending e-mail to SASTech@hp.com or your HP partner.

TEST TOPOLOGY

- 1 x HP ProLiant DL580 G7 with:
  - 4P/32C 2.26GHz (Intel Xeon® X7560 processors)
  - 256GB RAM
  - 8 x 146GB 15K SAS SFF internal disks
- 4 x HP P2000 G3 MSA Array System:
  - Dual Controller
  - 2GB Cache memory per Dual Controller
  - 8Gb Fibre Channel
  - 49 X 146GB 15K RPM SAS SFF Drives
- Red Hat Enterprise Linux (RHEL) 6.1 Server
- Host4 specific
  - Device Mapper Multipath I/O
  - LVM Striping using a 64KB stripe size
- Guest specific[4]
  - 8 VCPU (Virtual CPU)
  - 64GB of memory
  - KVM virtIO io-native, cache-none for all disks
- For each of the 3 environments, Production, Development and Test
  - XFS file system
  - All file systems are created on the host
  - Two file systems created: saswork and sasdata
  - Sasdata/input populated with 119GB of files for every 10 users

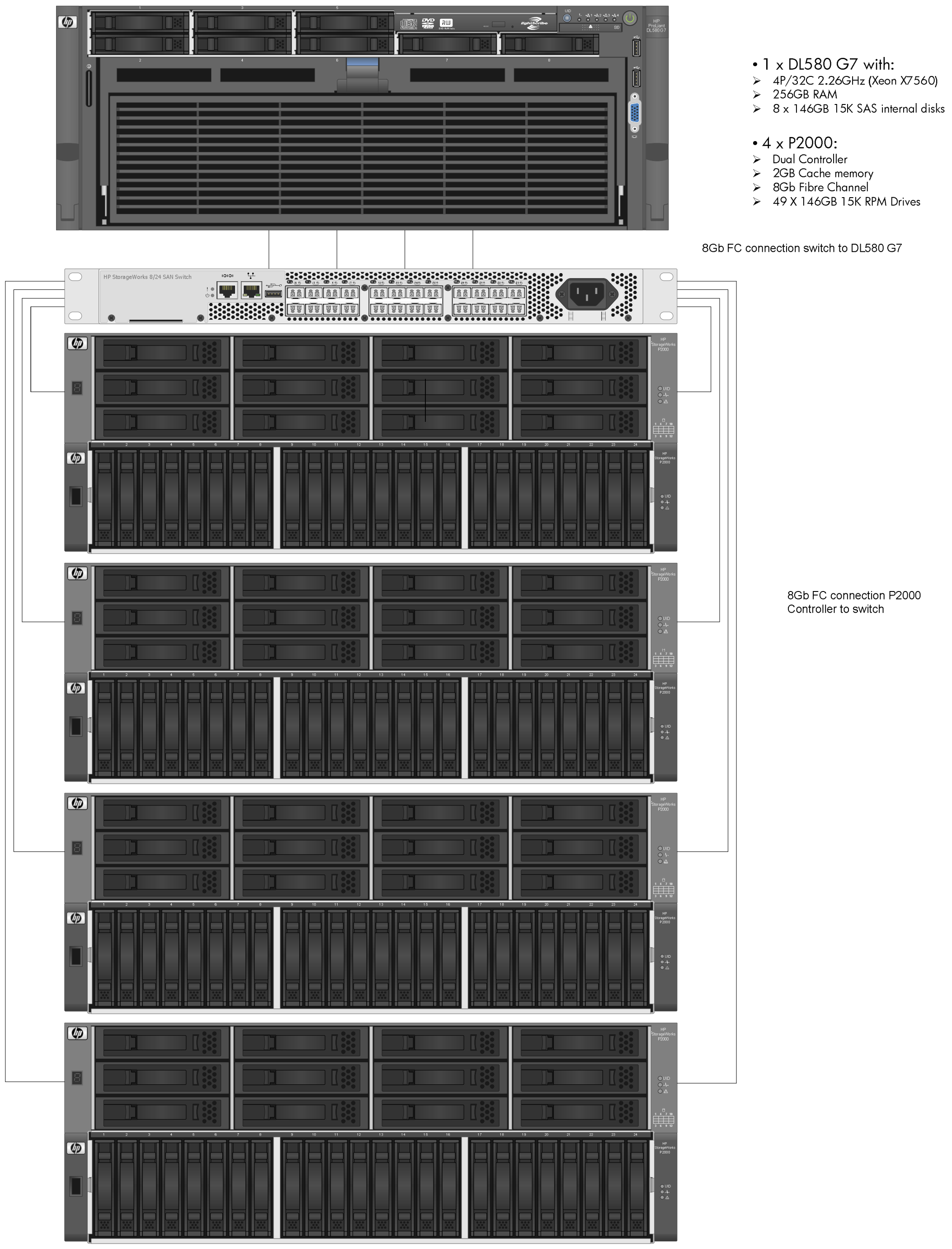
• 1 x DL580 G7 with:
➢ 4P/32C 2.26GHz (Xeon X7560)
➢ 256GB RAM
➢ 8 x 146GB 15K SAS internal disks
• 4 x P2000:
➢ Dual Controller
➢ 2GB Cache memory
➢ 8Gb Fibre Channel
➢ 49 X 146GB 15K RPM Drives
8Gb FC connection switch to DL580 G7
8Gb FC connection P2000 Controller to switch
HP StorageWorks 8/24 SAN Switch

Figure 1 – Architectural diagram of test bed

TEST METHODOLOGY

For our testing, we used the SAS Mixed Analytics workload. This is a workload that is evenly divided between I/O intensive jobs and compute intensive jobs. While no user's environment is the same, this workload is a good approximation of most SAS implementations.

The workload is made up of fictitious users. A10-user test submits 36 simultaneous jobs, a 20-user test submits 72 jobs, a 40-user test submits 144 jobs, and a 60-user test submits 216 SAS jobs.

Metrics monitored during the tests are

- Script wall time – the amount of time it takes for the test script to complete
- Cumulative wall time for each SAS job – the total amount of wall time it takes for all jobs to complete
- User CPU consumed
- System CPU consumed
- Longest running job – how long did the longest running job take to complete
- I/O throughput for both the guests and the host – the MB/sec or GB/sec of peak and sustained I/O observed during the test run.

RHEL 6 ADJUSTMENTS/TUNING APPLIED

To both host and guests

- Use command “tuned-adm” (pronounced Tune-D) to set profile for “enterprise-storage”
- Read Ahead set to 16K with blockdev command

Host (specific)

- Transparent Huge Pages enabled (for guests)[55]
- vm.swappiness = 10 (prevents swapping of guests)

Guests

- irqbalance service disabled (after boot up)5
- Transparent Huge Pages (TPH) disabled5

EXECUTION PROCEDURE

Note: Scripts available in the Appendices

1. Host – Scripts in Appendix B
   - Set read ahead to 16K using the blockdev command
   - Flush (sync) and drop file system page cache
   - Interrupt Request Queue (IRQ) is stopped
   - Boot up Development guest

2. Guest – Development Scripts for the development environment in Appendix C
   - Fully allocate guest memory
   - Stop irqbalance and disable THP
   - Flush (sync) and drop file system page cache
3. Host
   - Wait for host to convert guest dev process pages to hugepages (takes a few minutes)
   - Boot up test guest
4. Guest – Test Scripts for the test environment in Appendix D
   - Fully allocate guest memory
   - Stop irqbalance and disable THP
   - Flush (sync) and drop file system page cache
1. Host
   - Wait for host to convert guest dev process pages to hugepages (takes a few minutes)
   - Use the taskset command to pin the SAS processes on the host to specific host processors and cores

TEST RESULTS

The following are the test results. We ran each test multiple times and averaged the results.

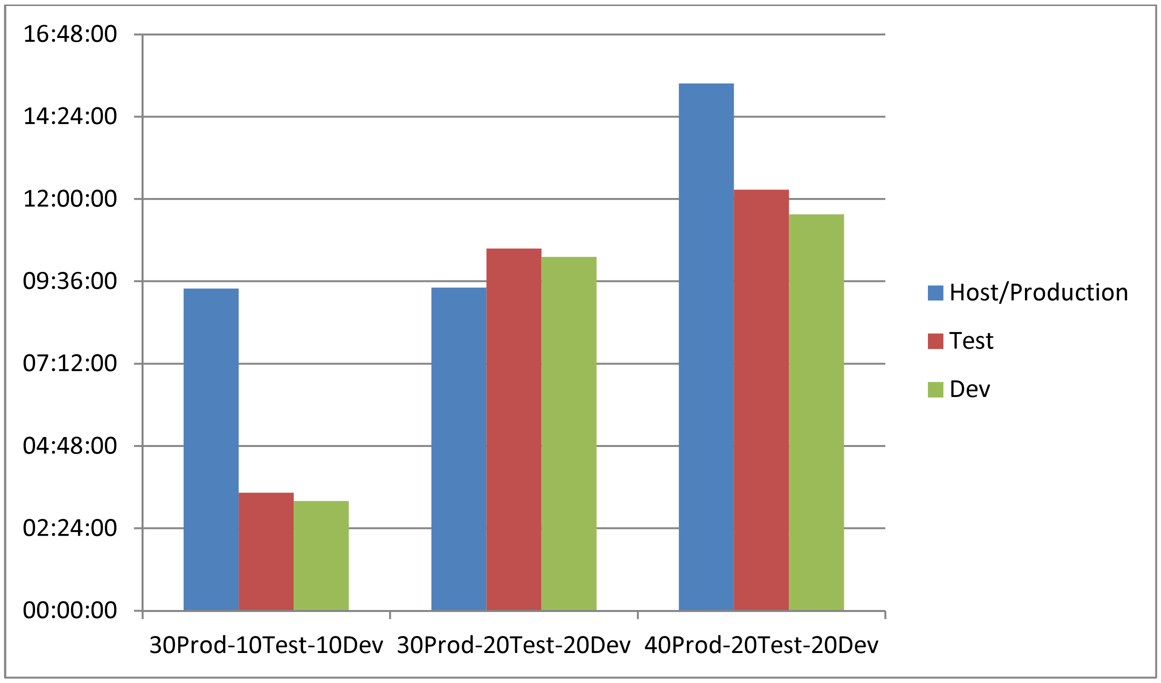

Figure 2 - Graph of Cumulative Jobs wall times for all production/development/test runs

	Host/Production	Test	Development
Script Wall	42:54	NA	NA
Cumulative Jobs Wall	12:38:53	NA	NA
User CPU	8:29:19	NA	NA
System CPU	50:08	NA	NA
Longest Running	35:34	NA	NA

Table 1. Production only 40-user run

	Host/Production	Test	Development
Script Wall	43:04	NA	NA
Cumulative Jobs Wall	8:55:47	NA	NA
User CPU	6:16:00	NA	NA
System CPU	35:25	NA	NA
Longest Running	34:13	NA	NA

Table 2. Production only 30-user run

	Host/Production	Test	Development
Script Wall	43:04	43:02	43:00
Cumulative Jobs Wall	9:23:31	3:26:13	3:12:00
User CPU	6:37:16	2:25:58	2:24:59
System CPU	36:14	15:28	16:07
Longest Running	36:14	40:26	40:22

Table 3. Production 30-user run, test 10-user run, development 10-user run

	Host/Production	Test	Development
Script Wall	42:56	51:09	51:38
Cumulative Jobs Wall	9:25:21	10:33:36	10:18:33
User CPU	6:44:51	4:44:01	4:44:50
System CPU	37:42	41:30	31:15
Longest Running	37:42	51:08	51:37

Table 4. Production 30-user run, test 20-user run, development 20-user run

	Host/Production	Test	Development
Script Wall	42:57	51:36	52:55
Cumulative Jobs Wall	15:22:20	12:16:05	11:33:11
User CPU	8:58:09	4:44:51	4:44:39
System CPU	51:39	27:18	28:44
Longest Running	40:21	52:35	52:54

Table 5. Production 40-user run, test 20-user run, development 20-user run

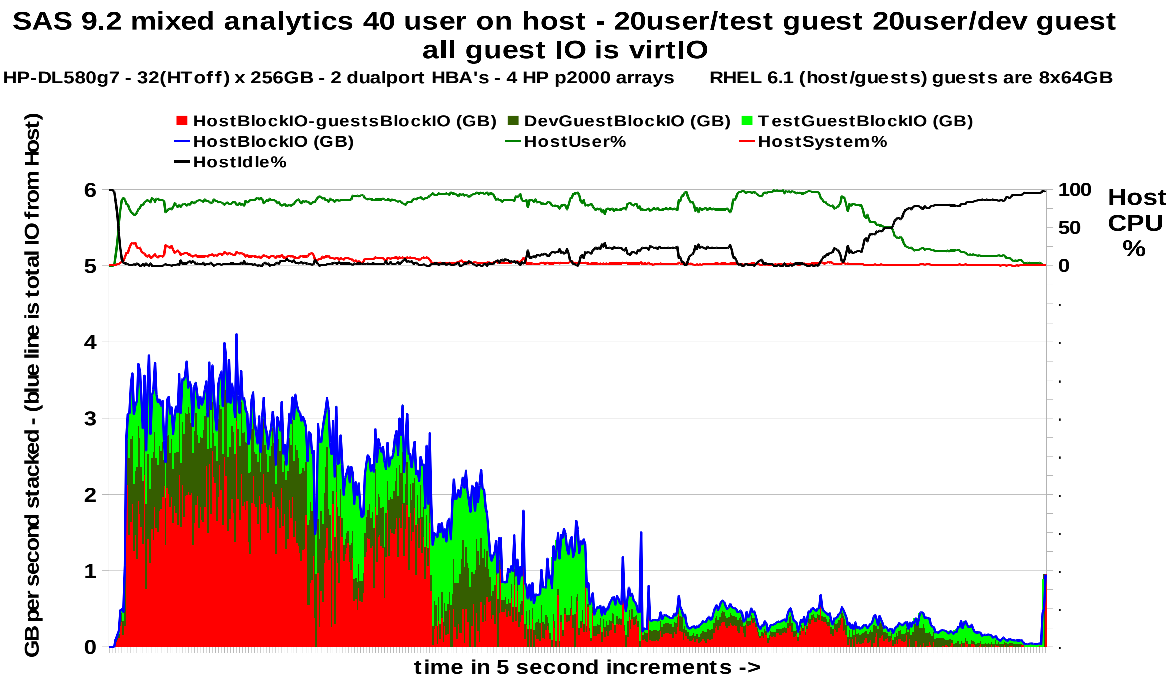

Figure 3. Graph of 40-user run, test of 20-user run, development of 20-user run

TEST ANALYSIS SUMMARY

The results are:

- Total I/O varied little between a 30-user and 40-user host-only run
  - Total I/O 2.9GB/sec sustained 30-user versus 3.2GB/sec sustained 40-user
- Total I/O remained relatively constant between 30-10-10, 30-20-20 and 40-20-20 runs
  - Total I/O 3.5GB/sec sustained for all scenarios
- Isolation of guests to host
  - Varying the number of users within the guests had little impact on the host runtimes
- I/O results were achieved using Virtual I/O
  - The expectation is that pass through or Single Root I/O Virtualization (SR-IOV) would provide even greater I/O throughput for the guests
- Red Hat KVM is very efficient
  - The additional processing and I/O requirements for KVM are light.
  - Increasing guest workload (users) reasonably has little effect on the host

Increasing host workload (users) reasonably has little effect on the guests,

CONCLUSION

The combination of HP ProLiant servers, in this case the DL580 G7, and RHEL 6.1 makes the utilization of KVM a very viable alternative for SAS environments to proprietary UNIX partitioning. The I/O overhead induced by virtualization is low relative to what has been experienced in the past.

Running Red Hat Enterprise Linux 6 on HP ProLiant DL580 G7 is a recommended configuration for SAS deployments.

GLOSSARY OF TERMS

Hard Partitions – A hard partition physically divides the computer into groups of cell boards, which operate independently of other groups of cell boards. Hard partitions isolate application environments from single points of failure, meaning applications running within hard partitions are not affected by hardware or software events occurring in other partitions. Each hard partition executes a single operating system image, providing software isolation and enabling alternate partitions to execute different versions of the operating system.

Features

- Closely corresponds to a standalone system
- Provides electrical isolation and isolates hardware failures to the specific hardware partition
- Granularity down to 1 cell
- Enables multiple operating systems and applications to be run on the same physical system
- Each partition has independent processor, memory, and I/O resources allocated
- Can increase/decrease processing power by adding/deleting cells to a partition

Soft Partitions – also referred to as virtual partitions – are provided by a software based partitioning product that carves up an individual server into several smaller virtual servers, each with their own operating system, resources, and applications. Any application or operating system related failures can only impact or bring down the virtual partition in which it is executing without affecting other virtual partitions executing on the same system. The amount of overhead incurred utilizing virtual partitions is negligible, because virtual partitions, like hard partitions have a shared nothing environment. There is no need for an underlying hypervisor to intercept and dispense requests for service.

Features

- Provides operating system, application, and resource isolation within a server or hardware partition
- High performance software partitioning product, due to:
  - Separate processor core, memory, and I/O hardware resources per partition
  - Cell local resource support
- Resource granularities: processor core, I/O slot
- Applications run the same within a virtual partition as they do in a standalone OS

Virtualization – also referred to as virtual machines – is a hypervisor based product that can be used to carve an individual hard partition or server into several smaller virtual servers, each with their own operating system, shares of resources, and applications. Any application or operating system related failure can impact the virtual machine (VM) in which it is executing and does not affect other VMs executing on the same system.

Host – The server or partition on which the virtual machine software is running.

Guest – Any virtualized machine running its own operating system is referred to as a guest.

APPENDIX A – CREATE GUESTS XML

DEVELOPMENT GUEST

```
<domain type='kvm'>
  <name>dev</name>
  <uuid>cd6a6b2e-34f4-fbad-01d7-8f7d20816ff0</uuid>
  <memory>52428800</memory>
  <currentMemory>52428800</currentMemory>
  <vcpu>8</vcpu>
  <os>
    <type arch='x86_64' machine='rhel6.1.0'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source file='/boot/dev/dev'/>
      <target dev='vda' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x15' function='0x0'/>
    </disk>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/mapper/vgdevstorage-asuite'/>
      <target dev='vdb' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x17' function='0x0'/>
    </disk>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/mapper/vgdevstorage-input'/>
      <target dev='vdc' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x18' function='0x0'/>
    </disk>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/mapper/vgdevstorage-output'/>
      <target dev='vdd' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x19' function='0x0'/>
    </disk>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/mapper/vgdevstorage-sas'/>
      <target dev='vde' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1a' function='0x0'/>
    </disk>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/mapper/vgdevstorage-saswork'/>
      <target dev='vdf' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x1b' function='0x0'/>
    </disk>
    <interface type='bridge'>
      <mac address='52:54:00:dc:2a:0d'/>
      <source bridge='br0'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x13' function='0x0'/>
    </interface>
    <serial type='pty'>
      <target port='0'/>
    </serial>
    <console type='pty'>
      <target type='serial' port='0'/>
    </console>
    <input type='tablet' bus='usb'/>
    <input type='mouse' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes'/>
    <sound model='ich6'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x14' function='0x0'/>
    </sound>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
    </video>
    <memballoon model='virtio'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x16' function='0x0'/>
    </memballoon>
  </devices>
</domain>
```

TEST GUEST

```
<domain type='kvm' id='12'>
  <name>test</name>
  <uuid>cd6a6b2e-34f4-fbad-01d7-8f7d20816fef</uuid>
  <memory>52428800</memory>
  <currentMemory>52428800</currentMemory>
  <vcpu>8</vcpu>
  <os>
    <type arch='x86_64' machine='rhel6.1.0'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source file='/home/test1'/>
      <target dev='vda' bus='virtio'/>
      <alias name='virtio-disk0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
    </disk>
    <disk type='block' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='native'/>
      <source dev='/dev/mapper/vgsas-asuite'/>
      <target dev='vdb' bus='virtio'/>
      <alias name='virtio-disk1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
```

```
</disk>
<disk type='block' device='disk'>
  <driver name='qemu' type='raw' cache='none' io='native'/>
  <source dev='/dev/mapper/vgsasdata-input'/>
  <target dev='vdc' bus='virtio'/>
  <alias name='virtio-disk2'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x0'/>
</disk>
<disk type='block' device='disk'>
  <driver name='qemu' type='raw' cache='none' io='native'/>
  <source dev='/dev/mapper/vgsasdata-output'/>
  <target dev='vdd' bus='virtio'/>
  <alias name='virtio-disk3'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x09' function='0x0'/>
</disk>
<disk type='block' device='disk'>
  <driver name='qemu' type='raw' cache='none' io='native'/>
  <source dev='/dev/mapper/vgsasdata-sas'/>
  <target dev='vde' bus='virtio'/>
  <alias name='virtio-disk4'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x0'/>
</disk>
<disk type='block' device='disk'>
  <driver name='qemu' type='raw' cache='none' io='native'/>
  <source dev='/dev/mapper/vgsas-saswork'/>
  <target dev='vdf' bus='virtio'/>
  <alias name='virtio-disk5'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0b' function='0x0'/>
</disk>
<interface type='bridge'>
  <mac address='52:54:00:dc:2a:0e'/>
  <source bridge='br0'/>
  <target dev='vnet1'/>
  <model type='virtio'/>
  <alias name='net0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
<serial type='pty'>
  <source path='/dev/pts/6'/>
  <target port='0'/>
  <alias name='serial0'/>
</serial>
<console type='pty' tty='/dev/pts/6'>
  <source path='/dev/pts/6'/>
  <target type='serial' port='0'/>
  <alias name='serial0'/>
</console>
<input type='tablet' bus='usb'>
  <alias name='input0'/>
</input>
<input type='mouse' bus='ps2'/>
<graphics type='vnc' port='5901' autoport='yes'/>
<sound model='ich6'>
  <alias name='sound0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
</sound>
<video>
  <model type='cirrus' vram='9216' heads='1'/>
  <alias name='video0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
</video>
<memballoon model='virtio'>
  <alias name='balloon0'/>
```

```
      <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
    </memballoon>
  </devices>
</domain>
```

APPENDIX B – HOST SCRIPTS

SETUP SCRIPT – SETUP THE ENVIRONMENT – ONLY USED AFTER A REBOOT

```
#!/bin/bash

sync
# set read ahead for each file system mounted and utilized
blockdev --setra 16384 /dev/mapper/vgsasdata-output
blockdev --setra 16384  /dev/mapper/vgsasdata-sas
blockdev --setra 16384 /dev/mapper/vgsasdata-input
blockdev --setra 16384 /dev/mapper/vgsas-saswork
blockdev --setra 16384 /dev/mapper/vgsas-asuite
blockdev --setra 16384 /dev/mapper/vg--dev--sas-n--output
blockdev --setra 16384  /dev/mapper/vg--dev--sas-n--sas
blockdev --setra 16384 /dev/mapper/vg--dev--sas-n--input
blockdev --setra 16384 /dev/mapper/vg--dev--sas-n--saswork
blockdev --setra 16384 /dev/mapper/vgdevstorage-asuite
blockdev --setra 16384 /dev/mapper/vgdevstorage-output
blockdev --setra 16384  /dev/mapper/vgdevstorage-sas
blockdev --setra 16384 /dev/mapper/vgdevstorage-input
blockdev --setra 16384 /dev/mapper/vgdevstorage-saswork
blockdev --setra 16384 /dev/mapper/vgdevstorage-asuite
```

RUN SCRIPT – COMPLETE THE RUNTIME ENVIRONMENT – USED EACH TIME A RUN IS EXECUTED

```
#!/bin/bash

echo never >  /sys/kernel/mm/redhat_transparent_hugepage/enabled
sync
echo 3 > /proc/sys/vm/drop_caches
swapoff -a
swapon -a
blockdev --setra 16384 /dev/mapper/vgsasdata-output
blockdev --setra 16384  /dev/mapper/vgsasdata-sas
blockdev --setra 16384 /dev/mapper/vgsasdata-input
blockdev --setra 16384 /dev/mapper/vgsas-saswork
blockdev --setra 16384 /dev/mapper/vgsas-asuite
blockdev --setra 16384 /dev/mapper/vg--dev--sas-n--output
blockdev --setra 16384  /dev/mapper/vg--dev--sas-n--sas
blockdev --setra 16384 /dev/mapper/vg--dev--sas-n--input
blockdev --setra 16384 /dev/mapper/vg--dev--sas-n--saswork
blockdev --setra 16384 /dev/mapper/vgdevstorage-asuite
blockdev --setra 16384 /dev/mapper/vgdevstorage-output
blockdev --setra 16384  /dev/mapper/vgdevstorage-sas
blockdev --setra 16384 /dev/mapper/vgdevstorage-input
blockdev --setra 16384 /dev/mapper/vgdevstorage-saswork
blockdev --setra 16384 /dev/mapper/vgdevstorage-asuite
./vmstat 5 > /tmp/vmstat &
tail -f /tmp/vmstat
```

APPENDIX C – DEVELOPMENT ENVIRONMENT SCRIPTS

SETUP SCRIPT – SETUP THE ENVIRONMENT – ONLY USED AFTER A REBOOT

```
#!/bin/bash

service irqbalance stop
echo 3 > /proc/sys/vm/drop_caches
sync
echo never >  /sys/kernel/mm/redhat_transparent_hugepage/enabled
blockdev --setra 16384 /dev/vdd
blockdev --setra 16384  /dev/vde
blockdev --setra 16384 /dev/vdc
blockdev --setra 16384 /dev/vdf
blockdev --setra 16384 /dev/vdbcd /asuite/input
for i in *
do
dd if=/asuite/input/${i} of=/dev/null bs=1M &
done
```

RUN SCRIPT – COMPLETE THE RUNTIME ENVIRONMENT – USED EACH TIME A RUN IS EXECUTED

```
#!/bin/bash

echo 3 > /proc/sys/vm/drop_caches
sync
echo never >  /sys/kernel/mm/redhat_transparent_hugepage/enabled
swapoff -a
swapon -a
blockdev --setra 16384 /dev/vdd
blockdev --setra 16384  /dev/vde
blockdev --setra 16384 /dev/vdc
blockdev --setra 16384 /dev/vdf
./vmstat 5  > /tmp/vmstat &
tail -f /tmp/vmstat
```

APPENDIX D – TEST ENVIRONMENT SCRIPTS

SETUP SCRIPT – SETUP THE ENVIRONMENT – ONLY USED AFTER A REBOOT

```
#!/bin/bash

service irqbalance stop
echo 3 > /proc/sys/vm/drop_caches
sync
echo never >  /sys/kernel/mm/redhat_transparent_hugepage/enabled
blockdev --setra 16384 /dev/vdd
blockdev --setra 16384  /dev/vde
blockdev --setra 16384 /dev/vdc
blockdev --setra 16384 /dev/vdf
blockdev --setra 16384 /dev/vdb
cd /asuite/input
for i in *
do
dd if=/asuite/input/${i} of=/dev/null bs=1M &
done
```

RUN SCRIPT – COMPLETE THE RUNTIME ENVIRONMENT – USED EACH TIME A RUN IS EXECUTED

```
#!/bin/bash

echo 3 > /proc/sys/vm/drop_caches
sync
echo never >  /sys/kernel/mm/redhat_transparent_hugepage/enabled
swapoff -a
swapon -a
blockdev --setra 16384 /dev/vdd
blockdev --setra 16384  /dev/vde
blockdev --setra 16384 /dev/vdc
blockdev --setra 16384 /dev/vdf
blockdev --setra 16384 /dev/vdb
./vmstat 5  > /tmp/vmstat &
tail -f /tmp/vmstat
```

FOR MORE INFORMATION

HP and SAS, www.hp.com/go/sas

HP ProLiant servers, www.hp.com/go/ProLiant

HP ProLiant DL580 server, www.hp.com/servers/dl580

HP P2000 G3 MSA Array Systems, www.hp.com/go/p2000

HP and Red Hat, http://h71028.www7.hp.com/enterprise/cache/321114-0-0-0-121.html

Red Hat and SAS, www.redhat.com/sas

If you would like a site specific sizing or assistance with the definition of a POC, please send e-mail to SASTech@hp.com

To help us improve our documents, please provide feedback at http://h71019.www7.hp.com/ActiveAnswers/us/en/solutions/technical_tools_feedback.html.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Name:	Bob Augustine	Name:	Barry Marson
Enterprise:	Hewlett-Packard Development Company, LP	Enterprise:	Red Hat
Address:	2001 Butterfield Road	Address:	314 Littleton Road
City, State ZIP:	Downers Grove, Il. 60515	City, State ZIP:	Westford, Ma. 01886
Work Phone:	630.561.2956	Work Phone:	978.392.3155
E-mail:	bob.augustine@hp.com	E-mail:	bmarson@rh.com

END NOTES

---

[1] Please see the glossary for the definition of hard partitioning and soft partitioning technology.

[2] Please see the glossary for the definition of virtualization technologies.

[3] Please see the glossary for the definition of host and guest.

[4] Please see the glossary for the definition of host and guest.

[5] These are adjusted during the execution procedure.