Red Hat Reference Architecture Series

Performance & Scalability of Virtualized SAP Servers on Red Hat® Enterprise Linux®

SAP LinuxLab Certification Suite (SLCS) 2.3 (ERP2005 SR1)
MaxDB 7.6
Red Hat Enterprise Linux 5.2
IBM x3850 M2 Server
IBM DS3400 Storage

Version 1

March 2008

redhat.

Performance & Scaling of Virtualized SAP Servers on Red Hat® Enterprise Linux® (RHEL®)

Table of Contents

1. Executive Summary

In January / February 2008, SAP conducted a Linux Virtualization Certification Workshop to demonstrate the readiness of different technology partners to run SAP instances in virtual servers. As a result of this workshop SAP has affirmed that Red Hat Enterprise Linux (RHEL) Virtualization is ready for production use with SAP:

- Scales very well (close to linear) up to 100% host allocation (= 16 vCPUs on a 16 core system)
- Handles 100% over-subscription (= 32 vCPUs on a 16 core system) very well with no collapse in performance
- Uses the available I/O bandwidth extremely well

Red Hat Virtualization Technology displayed exceptional scalability and delivered between 25% and >100% higher aggregate throughput than competing virtualization technology.

2. Goals

The SAP Linux Virtualization Certification Workshop used the SAP LinuxLab Certification Suite (SLCS) for:

- Technical certification of a processor model. The focus was on x86-64 architecture based servers, i.e., Intel Tigerton.

- Technical certification of RHEL virtualization technology on Red Hat Enterprise Linux 5.

- Show the capabilities of SAP's enterprise ready database, MaxDB, in virtualized environments.

- Examine the performance and scaling impact at the SAP application level of having up to 100% CPU over-subscription (= 32 vCPUs on a 16 core system).

- Work on SAP recommendations for sizing and configuration of virtual machines when using SAP.

3. SLCS Architecture & Configuration

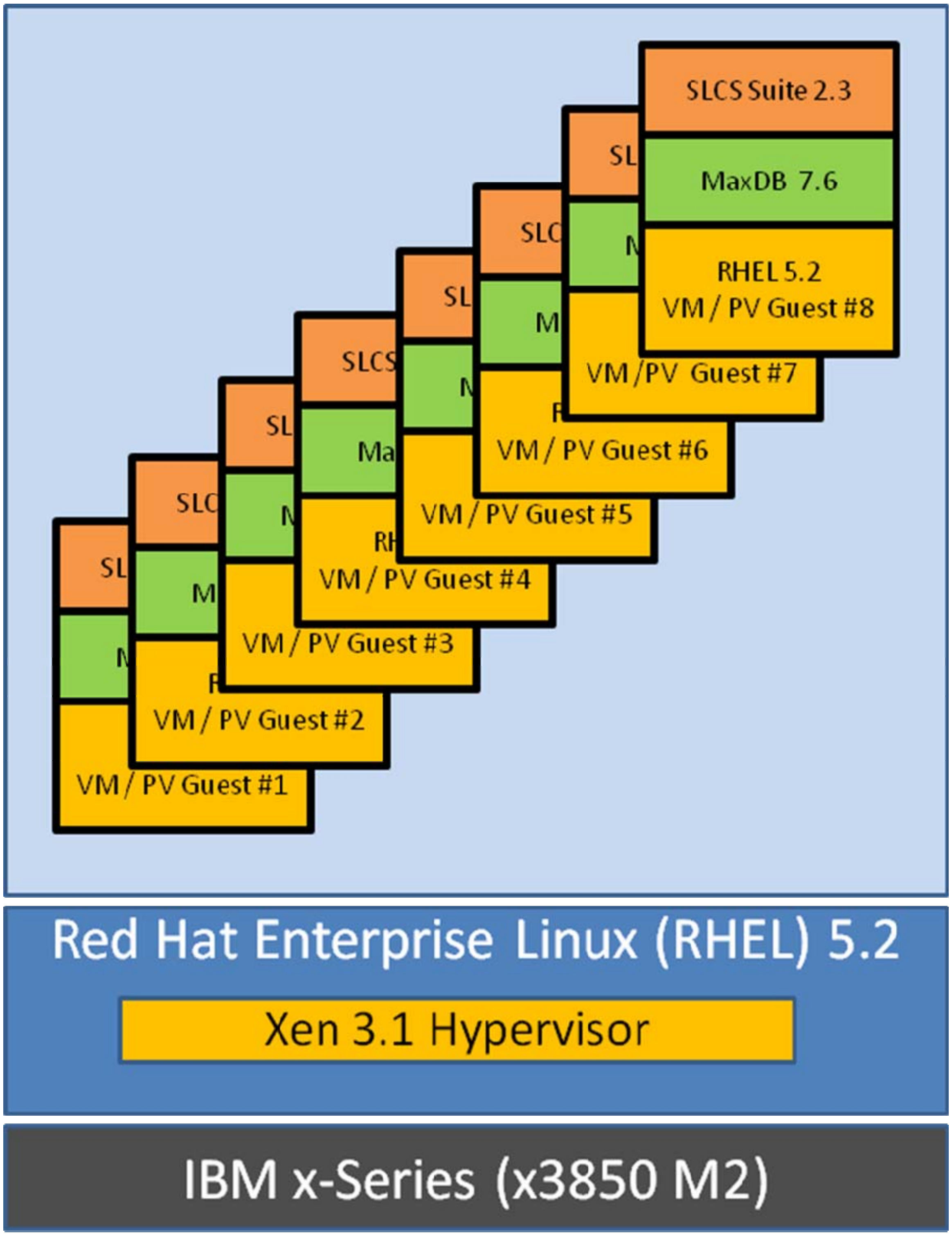

Figure 1: SLCS on RHEL 5.2 Virtual Servers

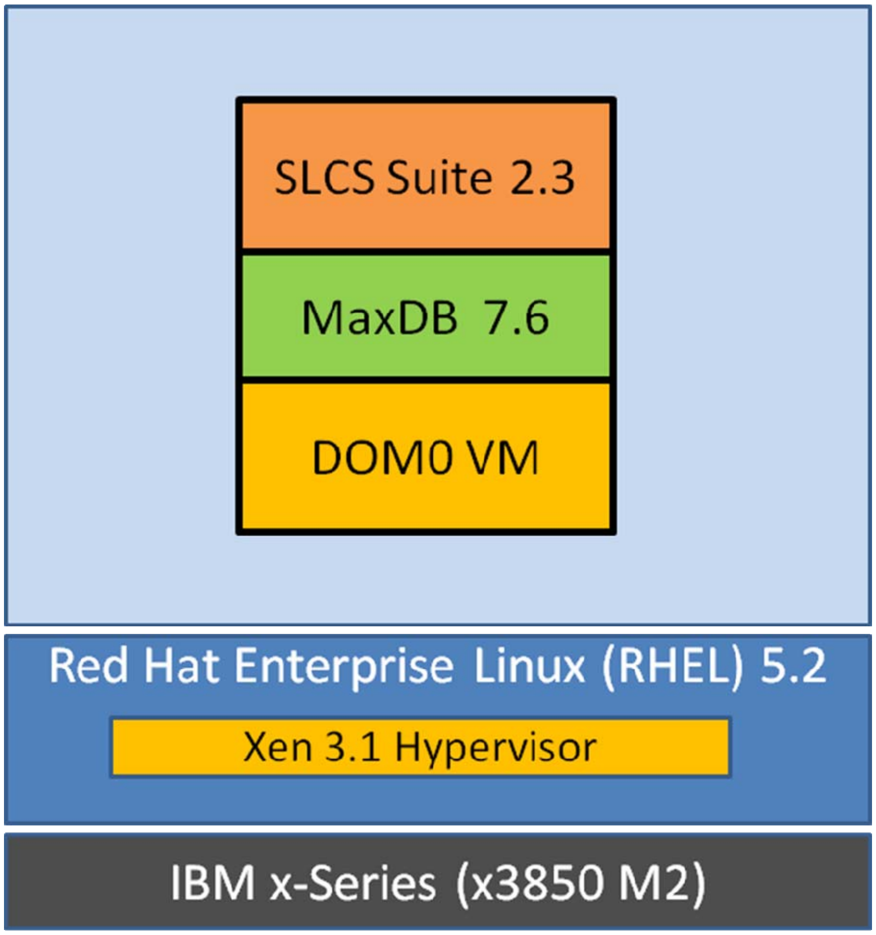

Figure 2: SLCS on RHEL 5.2 / Xen 3.1 DOM(0)

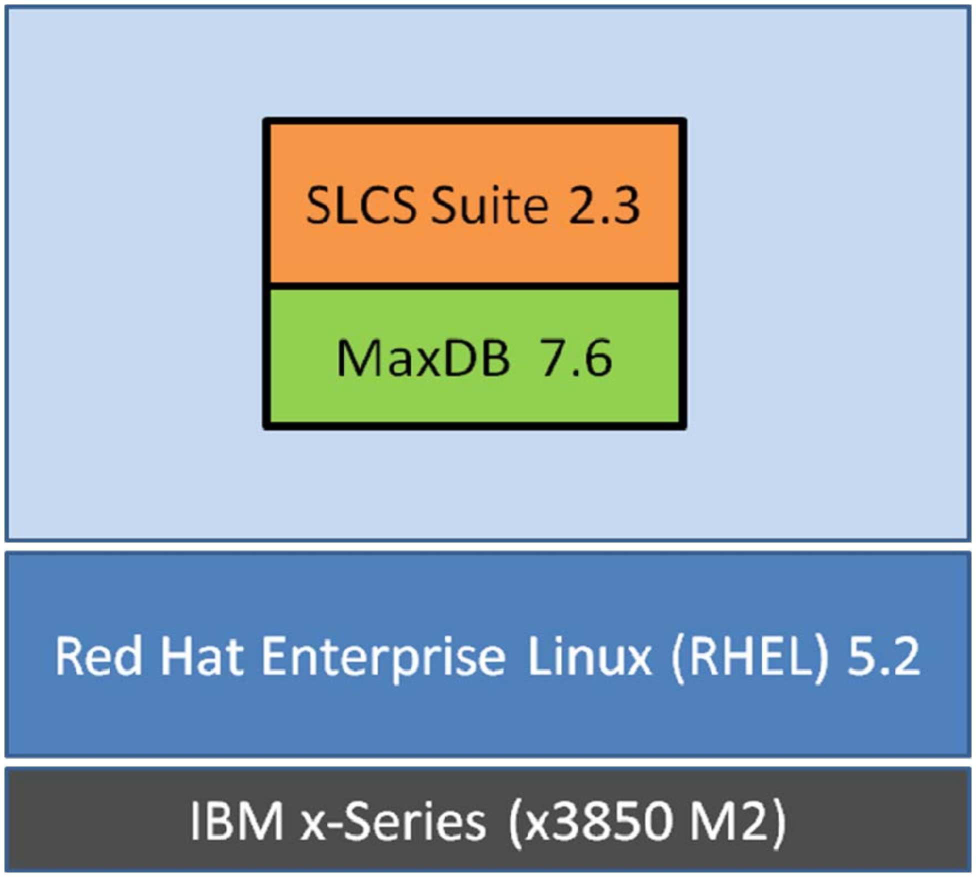

Figure 3: SLCS on Bare Metal

4. SAP LinuxLab Virtualization Certification Methodology

4.1 Test procedure

The last SAP LinuxLab Virtualization Workshop focused on the difference between the native (= bare metal) and the virtual performance of the same server. The main goal was to examine the performance overhead of the virtualization layer using different (non Unicode) SAP workloads. The virtual machines used to have a resource configuration of 2 vCPU's and 4GB of memory. The lessons learned from all participating parties led to performance improvement work on the virtualization platforms and SAP sizing recommendations, see SAP Note 962334.

This workshop had a slightly different focus. Having the latest hardware technology in place and using the performance improvements in the virtualization platforms allowed us to change two things. The resource configuration of a virtual machine to be tested was increased to 4 vCPU's and 16GB of memory and a comparison between native (= bare metal) and virtual systems was dropped. However, as shown later in this report, even for this workshop, Red Hat did run experiments with SLCS instances running (i) on bare metal, (ii) in dom0, and (iii) in RHEL virtual machines. Having only one virtual machine running (which uses ~25% of the available physical resources) may provide good performance when compared with the throughput for a native (= bare metal) machine with the same resources. But what happens to the performance when additional virtual machines with additional SAP instances are deployed? What does the scaling look like? How many virtual machines of the mentioned size of 4 vCPU's and 16GB memory can be deployed on a host without loosing too much performance? To answer these questions, especially when allocating up to 200% (= 8 VMs) of the CPU resources (and using them concurrently) was the main goal of this certification workshop.

There are three tests which were performed to measure performance and scalability:

- **Database load scenario** - The I/O scenario is realized by importing an SAP ERP2005SR1 system into a database. This phase is divided into three minor phases which cover 1) write only (Database Create) 2) read/write (Database Load) and 3) read only (Database Update/Initialization) workloads.
- **ABAP memory allocation scenario** - The memory scenario is realized by allocating and working on huge memory areas. In particular, this is mostly done by excessive use of mmap() and mprotect() functions.
- **Sales and Distribution (SD) ABAP scenario** - The CPU scenario is realized by running a Sales and Distribution business scenario.

4.2 Database load scenario

[Note: The SLCS database load scenario described below describes a database migration necessitated by increasing load over time. One of the goals of RHEL virtualization, in response to increasing load, is to offer applications & databases flexibility and longevity by enabling them to be hosted on larger VMs, on larger servers using the latest hardware and software.]

The database load scenario is a task which every customer has to perform at least once - the installation of the SAP system. After the initial installation the system may run for years without any change to the system infrastructure. But after several years of running the database on one server, as new users are added to the system the hardware could reach its capacity. In order to avoid running out of capacity customers have to change the hardware of the database server either by migrating to a faster system or by upgrading the hardware directly. Upgrading the hardware may be difficult after several years, because the CPU sockets or the layout of the memory modules may have changed, thus a migration is the only way to speed up the database server.

Migration of an existing SAP system is done by exporting the current database content to a database and operating system independent format called 'database export'. After creating a new database on a new system, this database export is loaded into the new database. During this time the SAP system has to be offline since the database is unavailable because of exporting and importing the content. Such export and import scenarios are normally performed during a weekend where the SAP system may not be needed. The important and essential part of the migration is that it has to be finished as fast as possible since by Monday the system's users will want to connect to the SAP system. Furthermore, the SAP consultants doing the migration are very expensive. Every hour which is saved due to a fast import is worth a lot of money.

An export consists of different files for each SAP content (e.g. application content is in a different file than documentation content). Furthermore each content file is separated into four different types of files. These four types are TOC-, STR-, EXT- and data files. The first three types are command files with no real application data content. The data files hold the compressed content of the database.

The MaxDB database used in this workshop was version 7.6.03_15. This version has two IO threads for each volume. Please note that the focus in this phase is the I/O throughput and not the CPU usage. Although, the CPU usage in this phase may also be interesting, the current setup does not allow specific I/O and CPU monitoring for every VM.

4.2.1 Create database (Write-Only)

The first part is creating the log volume having 2GB and afterwards, creating the three data volumes concurrently with each having 30GB. This first phase is plain I/O write performance. The command to create the database is:

```
(
cat <<EOF
db_create $DB_SID CONTROL,CONTROL
param_startsession
param_init
param_put MAXCPU              <anzahl CPUs>
param_put MAXLOCKS            300000
param_put JOIN_MAXTAB_LEVEL4  64
param_put _UNICODE            YES
param_put DEFAULT_CODE        ASCII
param_put _PACKET_SIZE        131072
param_put MCOD                NO
param_put SHAREDSQL           NO
EOF
param_addvolume 1 LOG /usr/sap/<SID>/devspaces/DISKL001 F 524288
param_addvolume 1 DATA /usr/sap/<SID>/devspaces/DISKD0001 F 2621440
param_addvolume 2 DATA /usr/sap/<SID>/devspaces/DISKD0002 F 2621440
param_addvolume 3 DATA /usr/sap/<SID>/devspaces/DISKD0003 F 2621440
param_addvolume 4 DATA /usr/sap/<SID>/devspaces/DISKD0004 F 2621440
param_addvolume 5 DATA /usr/sap/<SID>/devspaces/DISKD0005 F 2621440
param_directput ALLOW_MULTIPLE_SERVERTASK_UKTS NO
param_directput AUTO_RECREATE_BAD_INDEXES NO
param_directput BACKUP_BLOCK_CNT 8
param_directput CACHE_SIZE `expr '(' $DBCACHE '*' 128 - 3000 ')' / 10
'*' 8`
param_directput CAT_CACHE_SUPPLY 64064
param_directput CONVERTER_REGIONS 8
param_directput DATA_VOLUME_GROUPS 1
param_directput DATE_TIME_FORMAT INTERNAL
param_directput LOG_QUEUE_COUNT 11
param_checkall
db_admin
db_connect
util_connect
db_activate SUPERDBA,ADMIN
util_release
db_online
load_systab
util_connect
util_execute SET LOG AUTO OVERWRITE ON
util_release
db_offline
) | dbmcli -t $LOGFILE -s -R $DBROOT 2>&1
```

4.2.2 Load database (Read/Write)

The second part is the database load phase (the main import) with lots of writes for the data and also reads needed for index creation. The import of the data is as follows. During the script based installation, a script (import.sh) is executed. This file checks if all export files are available. If they are, it generates the R3load command files which contain the instructions for the import. Afterwards it starts a R3load process for each command file, e.g., documentation content, application content). The R3load process imports the data into the database. The number of concurrent R3load processes is determined by the number of available CPUs. We use the double amount of available CPUs as number of concurrent R3load processes.

The first job, SAPSDIC, is started without any other job running. Having a Unicode system, the first job has to be SAPSDIC. After the job finished, the other

R3load processes are started. After a job is imported, the script starts a new R3load process which imports the next one. The last job to be imported is SAPVIEW. The import of the job starts after all other jobs have finished successfully.

During the import a few computing resources are used to unpack the content data and to modify this unpacked data to fit into the database. Besides the small overhead of computing the I/O throughput is still the main factor that determines the time of the load phase.

The list of jobs is:

```
Job             Tables  Size/MB   Vardata/MB
--------------- ------ --------- ----------
SAP0000              1       0.8        0.0
SAPAPP10             1      90.9        0.0
SAPAPP11          3321      50.0        8.2
SAPAPP12          1726      50.0        0.0
SAPAPP13             8      50.0        0.0
SAPAPP14             1      54.7        0.0
SAPAPP15           128      50.0        0.0
SAPAPP16             1     193.4        0.0
SAPAPP17           581      50.0        0.7
SAPAPP18           827      50.0        0.0
SAPAPP19           620      50.0        0.0
SAPAPP20           167      50.0        0.0
SAPAPP21            23      50.0        0.0
SAPAPP22             1      61.7        0.0
SAPAPP23           235      50.0        0.0
SAPAPP24           610      50.0        0.0
SAPAPP25            32      50.0        0.0
SAPAPP26          2556      50.0        0.0
SAPAPP27           903      50.0        0.0
SAPAPP28          2752      50.0        0.0
SAPAPP29          1004      50.0        0.0
SAPAPP30           235      50.0        0.0
SAPAPP31           663      50.0        0.0
SAPAPP32          2995      50.0        0.0
SAPAPP33          3176      50.0        0.0
SAPAPP34          4065      50.0        0.0
SAPAPP35          1192      50.0        0.0
SAPAPP36          1293      50.0        1.1
SAPAPP37           111      50.0        0.0
SAPAPP38            47      50.0        0.0
SAPAPP39           317      44.0        1.2
SAPAPP40             1     302.6        0.0
SAPAPP41             1      17.0       17.0
SAPAPPL0         17390    2754.8        2.2
SAPAPPL1         11727    1176.0       49.9
SAPAPPL2          2915      50.0        0.1
SAPAPPL3          1458      50.0        0.0
SAPAPPL4             1     128.7        0.0
SAPAPPL5           908      50.0        0.2
SAPAPPL6             1      67.8       67.8
SAPAPPL7          2163      50.0        4.8
SAPAPPL8             1      79.1        0.0
SAPAPPL9             1      99.2        0.0
SAPCLUST            44      40.7       40.7
SAPDDIM              1       0.0        0.0
SAPDFACT             2       0.0        0.0
SAPDODS              5       0.0        0.0
SAPPOOL            122     102.0      102.0
```

```
SAPSDIC               359      924.8         60.9
SAPSDOCU              100       30.0          0.0
SAPSLEXC               10       19.0          0.1
SAPSLOAD               17        0.3          0.0
SAPSPROT              222       32.0          0.3
SAPSSE10                5      499.9          0.0
SAPSSE11               26      459.5          0.0
SAPSSE12                1       43.2          0.0
SAPSSEX0                1      741.9          0.0
SAPSSEX1              132      500.0        221.8
SAPSSEX2               18      500.0          0.0
SAPSSEX3               37      500.0          0.0
SAPSSEX4               21      499.9          1.3
SAPSSEX5                1     1212.2          0.0
SAPSSEX6                3      499.8          0.0
SAPSSEX7                9      500.0          0.0
SAPSSEX8                1      670.7          0.0
SAPSSEX9                1     2856.3          0.0
SAPSSEXC               51      500.0          0.0
SAPSSRC               175      549.1          2.7
SAPUSER                13        0.1          0.0
SAPUSER1                1        0.0          0.0
SAPVIEW                 0        0.0          0.0
```

The time stamp of this phase starts with the call of R3load of SAPSDIC and finishes with the successful end of SAPVIEW. The other tasks, like granting access rights or the call of dipgntab is not included in the time measurement.

4.2.3 Update database statistics (Read-Only)

Updating the database statistics which are database reads only is the last of the three sub measurements. The MaxDB command used is

```
sql_updatestat ${DB_SCHEMA}.* ESTIMATE SAMPLE 1000 ROWS
```

This command updates the statistics for the SQL optimizer. Directly before and after this command the time measurement is started and stopped. In this phase the first one thousand rows of each table are read to generate the statistics. We cannot determine how many bytes of data are read. Furthermore, the access of the database is sequential and only one CPU is used. Also, no database cache can be used, because every table is accessed only once and caching it doesn't speed up anything. This means, the elapsed time of this phase is not as useful as the values of the create phase and load phase, but since it is generated by default, it is included in the results.

4.3 ABAP memory allocation scenario

The ABAP memory allocation report does the following. It generates an internal ABAP table of a defined size by writing random data into this table. It then loops over this table for a defined time. During each loop it reads one dataset randomly and discards the result afterwards. The overall number of loops per second is displayed on the output screen after approximately 18 minutes.

The most important part are the defined size, defined time and number of reports started for each available CPU. The defined size of each internal table is 300MB. The defined runtime is 900 seconds, which means, that exactly after 900 seconds (clock time of the machine) the execution is stopped and the overall loops are written to a file. For every available CPU in the system, the script configures three dialog work processes. For every available dialog work process one report is started. For example, having a four (v)CPU machine, the script will configure a total of twelve work processes, restart the system and launch twelve cpu_load reports. Allocating 900MB of memory per CPU also sets the minimum amount of available memory to approximately 1GB per CPU.

4.4 Sales and Distribution ABAP scenario

To get a CPU related measurement in place, a Sales and Distribution scenario is used. It starts by creating an order, creating invoices, etc. until the complete order process is finished. The performance metric which is gathered with the test is the number of Sales and Distribution workers (or terminals) that can use the system concurrently while still maintaining a reasonable GUI response time.

Due to time constraints, this test was not run. These tests will be run as part of a second phase of this work which has already been initiated.

5. Hardware / Software Configuration

5.1 Hardware Configuration

- IBM System x3850 M2, equipped with:
    - 4x Intel quad core Xeon MP X7350 CPU @ 2.93GHz, 8MB L2 Cache, 16 cores in total
    - 128GB DDR2 synchronous RAM
    - 4x 73GB internal SAS Drives
    - 1 Emulex FibreChannel PCIE HBA, 4 Gbps, dual ported
- Storage array IBM DS3400, Model 1726-42X
    - 2x Controllers with 2 FC Host Ports per Controller
    - 12x 300 GB, 15K RPM SAS Drives

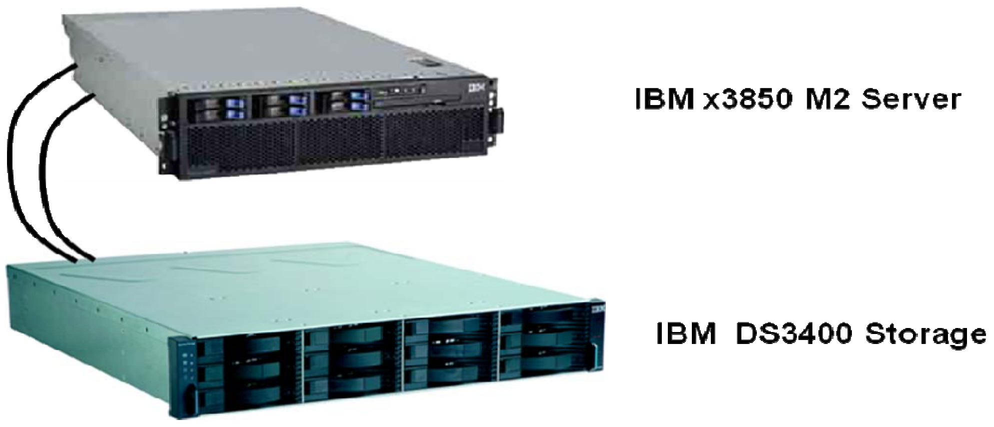

Figure 4

5.2 Storage configuration

We connected the FC HBA ports directly to the controllers of the storage array, no switches were used. We configured the storage to serve one LUN per controller, where the LUNs were setup identically as:

- LUN0 = 6x300 GB RAID 0+1 = 836 GB
- LUN1 = 6x300 GB RAID 0+1 = 836 GB

We created a LVM2 based volume group per LUN where we stored the logical volumes containing images of the guests evenly distributed. So guests 01,03,05,07 where put on the volume group on LUN0, guests 02,04,06,08 were put on the volume group on LUN1.

The logical volume per guest were set at 150GB each.

On the local storage we created:

- 2 Local Drives – RAID 1 = System Volume Group
- 2 Local Drives – RAID 1 = Volume Group with 2 logical volumes:
  - 20 GB containing the install tree of RHEL 5.1, Kickstarts, test kernel
  - 12 GB containig the SLCS 2.3 Kit

5.3 Hypervisor Setup

On the machine we installed a stock Red Hat Enterprise Linux 5.1 x86_64, base install with the following software added:

- kernel-xen-2.6.18-75.el5.75_bz412731.1.x86_64 – this kernel includes the mprotect patches which will be released with Red Hat Enterprise Linux 5.2.
- IBM Storage controller client software for storage configuration
- sysstat package
- createrepo package to create yum repositories

To ease the install and setup of the guests, we did the following:

- Create install tree at /var/nfs/tree which was exported Read Only via NFS.
- Create yum repository at /var/nfs/saprepo that contains the patched kernel, served via HTTP using the apache web server contained in Red Hat Enterprise Linux 5.1.
- Create kickstart files at /var/nfs/ks that automate the setup of the guests.
- Create /sapresults to collect the log files of the runs.

We limited domain-0 to one VCPU and for the final 8 guest run to 2GB RAM. During the runs 1-7 dom0 had 16GB of RAM.

As we faced some problems with the internal network card, we used eth2 for networking. So we put all guests on xenbr2. These are the only changes we made to the hypervisor setup.

5.4 Guest Setup

The guests were configured according to the agreed setup explained during the workshop:

- 64bit Operating system Red Hat Enterprise Linux 5.1.
- 4 VCPUs
- 16GB of RAM
- 150 GB of storage
- mprotect patched kernel (will be released as part of Red Hat Enterprise Linux 5.2)

In order to guarantee identical setup, we used the following kickstart file to create the guests:

```
# SLCS kickstart file
install
nfs --server=10.12.7.148 --dir=/var/nfs/tree
lang en_US.UTF-8
network --device eth0 --bootproto dhcp --hostname slcs01
rootpw --iscrypted $1$wahc3n.e$lLmBGCKbKIi8hR6ZJnFgZ1
firewall --disabled
authconfig --enableshadow --enablemd5
selinux --disabled
key --skip
timezone --utc Europe/Berlin
bootloader --location=mbr --driveorder=xvda --append="console=xvc0"
clearpart --all --drives=xvda --initlabel
part /boot --fstype ext3 --size=100 --ondisk=xvda
part swap --size=4096 --asprimary
part / --fstype ext3 --size=1 --grow --maxsize=9999999 --asprimary
repo --name=saprepo --baseurl=http://10.12.7.148/
reboot
%packages
@base
@core
@java
compat-libstdc++-33-3.2.3-61.x86_64
kernel-xen-2.6.18-75.el5.75_bz412731.1.x86_64
sysstat
%post
# create some dirs
mkdir /work
mkdir /sapresults
# add testsuite to fstab
echo "/dev/xvdb /mnt ext3 ro 0 0" >> /etc/fstab
# Disable services
chkconfig anacron off
chkconfig bluetooth off
chkconfig cpuspeed off
chkconfig crond off
chkconfig cups off
chkconfig iptables off
chkconfig ip6tables off
chkconfig microcode_ctl off
chkconfig pcscd off
chkconfig restorecond off
chkconfig sendmail off
chkconfig rhnsd off
chkconfig yum-updatesd off
# Setting IP
ADDR=`ip addr | grep eth0 | grep inet | cut -d " " -f 6 | cut -d "/" -f 1`
echo "# Kickstart generated hosts file for SAP SLCS" > /etc/hosts
echo "127.0.0.1		localhost.localdomain localhost" >> /etc/hosts
echo "$ADDR		slcs01" >> /etc/hosts
```

As you can see, we disabled some services not needed, we added the yum repos to install the patched kernel during kickstart and mangled the /etc/hosts file to meet the needs of the SLCS 2.3 setup.

We created all guests simultaneously using the following script:

```
#!/bin/bash
#
# create slcs
echo "destroying guest and xen config file"
xm destroy slcs$1
rm -f /etc/xen/slcs$1
echo "Creating slcs$1"
virt-install -n slcs$1
	-r 16384
```

```
        --vcpus=4
        --nographics
        -l nfs:10.12.7.148:/var/nfs/tree
        --file /dev/mapper/VslcsVG1-slcs$1
        -p
        -b xenbr2
        -x "ksdevice=eth0 ks=nfs:10.12.7.148:/var/nfs/ks/slcs$1.ks noipv6 ip=dhcp"
sed 's/,w"/,w", "phy:\/dev\/mapper\/Vkits-LSAPkit,xvdb,r"/g' /etc/xen/slcs$1 > /tmp/slcs$1
cp -f /tmp/slcs$1 /etc/xen/slcs$1
```

As you can see, we added the logical volume containing the SLCS 2.3 test kit as a read-only volume to the guests and mount this under /mnt.

With this setup we can recreate the guests without the need of any manual intervention.

This installation method creates the follwoing XEN config file in /etc/xen:

```
name = "slcs01"
uuid = "9ff2fdbf-e849-3402-6b31-4ea5ec919870"
maxmem = 16384
memory = 16384
vcpus = 4
bootloader = "/usr/bin/pygrub"
on_poweroff = "destroy"
on_reboot = "restart"
on_crash = "restart"
vfb = [  ]
disk = [ "phy:/dev/mapper/VslcsVG1-slcs01,xvda,w",
"phy:/dev/mapper/Vkits-LSAPkit,xvdb,r" ]
vif = [ "mac=00:16:3e:22:93:60,bridge=xenbr2" ]
```

6. Test Results

As already mentioned, the results on which the LinuxLab focused most were not the plain comparison between native (= bare metal) and virtual performance or even comparable throughput figures between the different virtualization platforms, but the development of the throughput when utilizing a machine up to its limits, i.e., scalability.

The x-axis of all graphs show 8 different measurement intervals. These eight intervals represent the number concurrently running virtual machines and the resources of host system which are used by them. With one virtual machine running, 25% of the host's physical CPU resources are allocated. With two virtual machines, 50% of the host's physical CPU resources are used by the virtual machines. With three virtual machines, 75% of the physical CPU resources are allocated. And with four virtual machines, the virtual machines are running using 100% of the physical CPU resources. This reflects the first interesting situation, when no physical CPU is available for the underlying hypervisor. The same holds true for five, six, seven and eight virtual machines are running consuming 125%, 150%, 175% and 150% percent respectively of the physical CPU resources.

When it comes to I/O, going from the one virtual machine to two virtual machines the amount of data which is written to the disk is doubled. With three virtual machines, three times more data is written to the disk compared to the one virtual machine case. With four virtual machines, four times more data is written to the disk compared to the one virtual machine case. Similarly, with five through eight virtual machines, there is 5 through 8 times the amount of data being written compared to the one virtual machine case.

# VMs / Guests (Host Utilization)	Database run times in seconds			STD: Memory Throughput Scaling	MAP: Memory Throughput Scaling
	Create	Load	Update / Initialization		
1 (25%)	100%	100%	100%	100%	100%
2 (50%)	123%	112%	112%	192%	194%
3 (75%)	198%	147%	120%	257%	272%
4 (100%)	251%	154%	125%	307%	308%
5 (125%)	333%	212%	131%	315%	338%
6 (150%)	409%	230%	131%	322%	344%
7 (175%)	481%	259%	139%	326%	346%
8 (200%)	549%	291%	152%	327%	342%

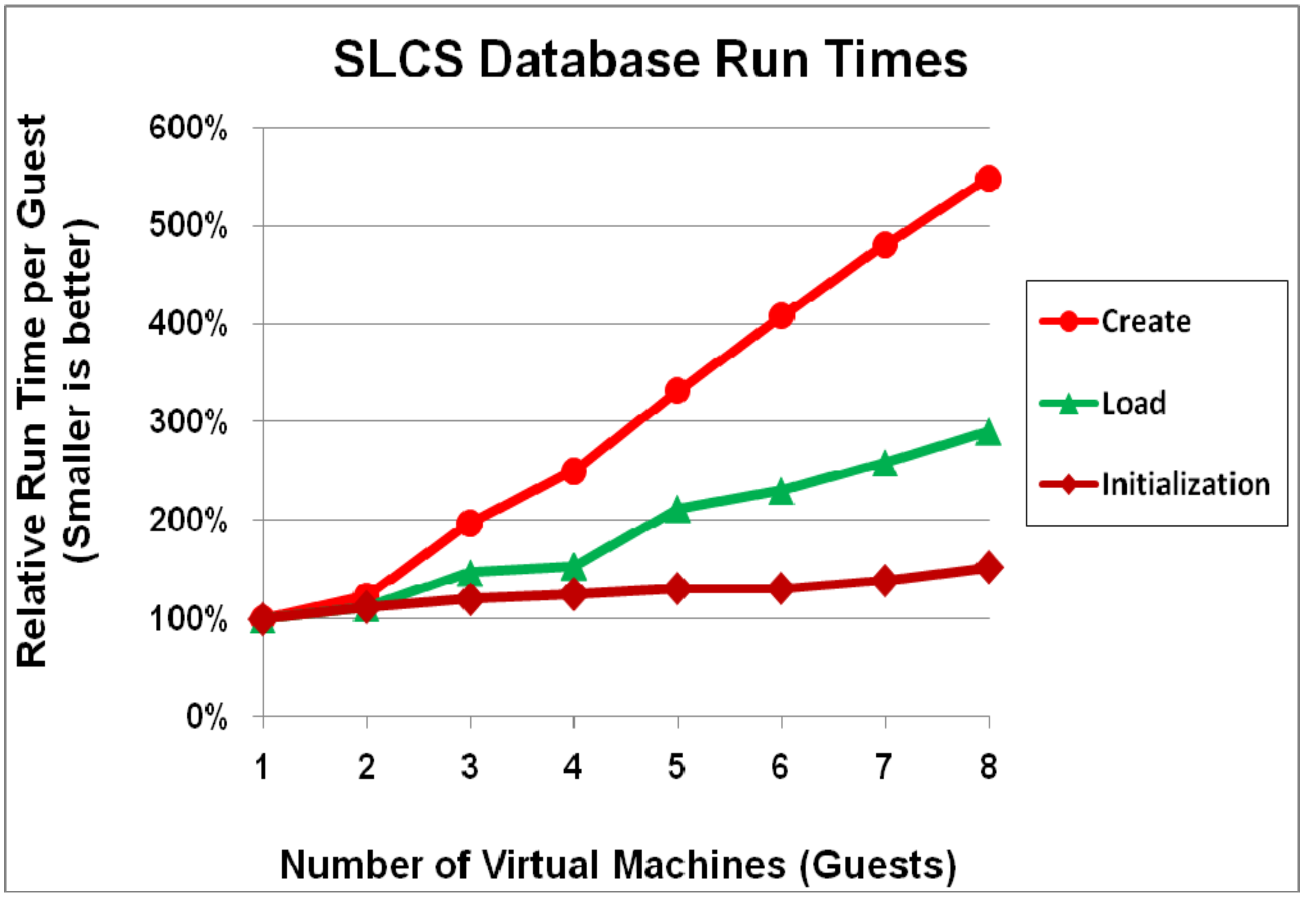

Figure 5

6.1 SLCS Database I/O Throughput

Figure 5 shows the relative performance of the three phases of DB I/O of SLCS. Note that several factors can influence these numbers including the number and type of disks and RAID settings, the type of storage subsystem, type of controllers, type of storage adapters and their respective speeds.

The results of read-only phase (= DB update/initialization) phase were ignored because of the low system utilization during this phase even when the system was 200% committed with 8 guests.

The write-only (= DB create) phase, e.g., creation of a new database volume, happens only on rare occasions. So, its scalability is not as critical.

In general, the read/write (= DB load) phase is the most commonly used scenario at a customer site. As shown in Figure 5, the read/write phase scales much better then the write-only phase and its performance and scalability was deemed production-ready.

6.2 SLCS Memory Throughput

On Linux you have two different kind of shared memory allocation technologies for SAP. You can either:

- use **mapped** memory through /dev/shm or
- use **standard** SystemV shared memory

Which implementation is used is set in the profile of the SAP instance. The parameter to use is es/implementation, which can have two values, map and std. The default for SAP kernel 7.00 is 'map', for SAP kernel 7.10 the default is 'std'.

The difference between these implementations is:

- When using virtualization, there is an additional memory translation layer between the memory of the virtual machines and the real physical memory. Using 'map', the memory to be used is mapped into the work process's memory window, whereas
- Using 'std', the whole memory can be accessed by the work process. To protect one SAP work process from writing into the memory context or area of an other user, we need mprotect() to deny any access to a memory region which does not belong to the user's context.

Figures 6 and 7 show the impact of running the 'SLCS memory throughput' performance tests when using both the es/implementation=std setting and the es/implementation=map setting. From these graphs it is clear that for SLCS, there is a very small performance difference between the 'map' and 'std' settings for RHEL 5.2 virtualization.

Figure 6 shows the absolute 'memory throughput' performance using the 'std' and 'map' modes. The numbers have been normalized using 1 VM 'std' performance as the base. The two modes give very comparable performance with the 'std' mode delivering slightly better performance than the 'map' mode at each data point.

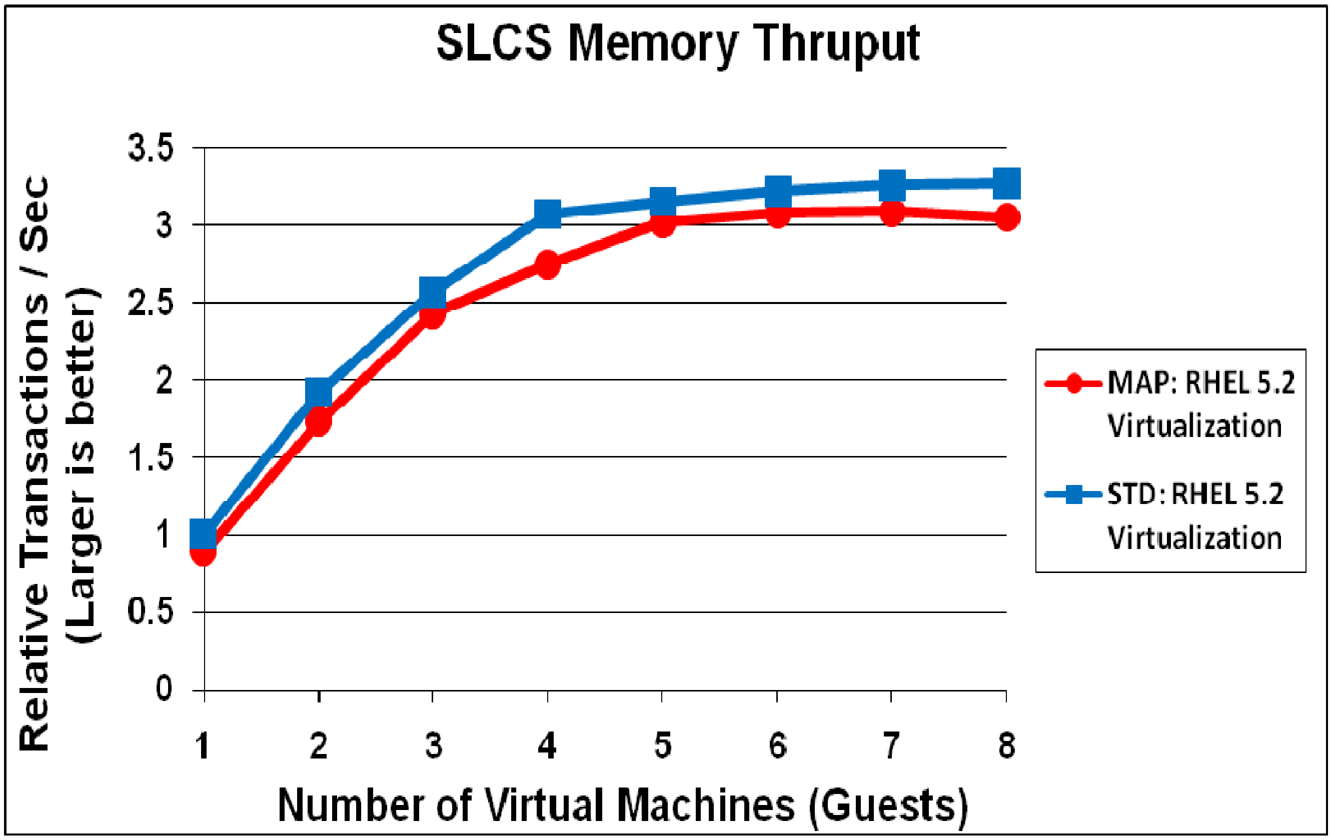

Figure 6

Figure 7 shows the scaling of 'memory throughput' performance using the 'std' and 'map' modes. As seen in Figure 6, the two modes give very comparable performance. However, it is important to note that since the 'map' mode has slightly lower single VM performance, its scaling (relative to its single VM performance) shows up as slightly higher than the scaling for the 'std' mode.

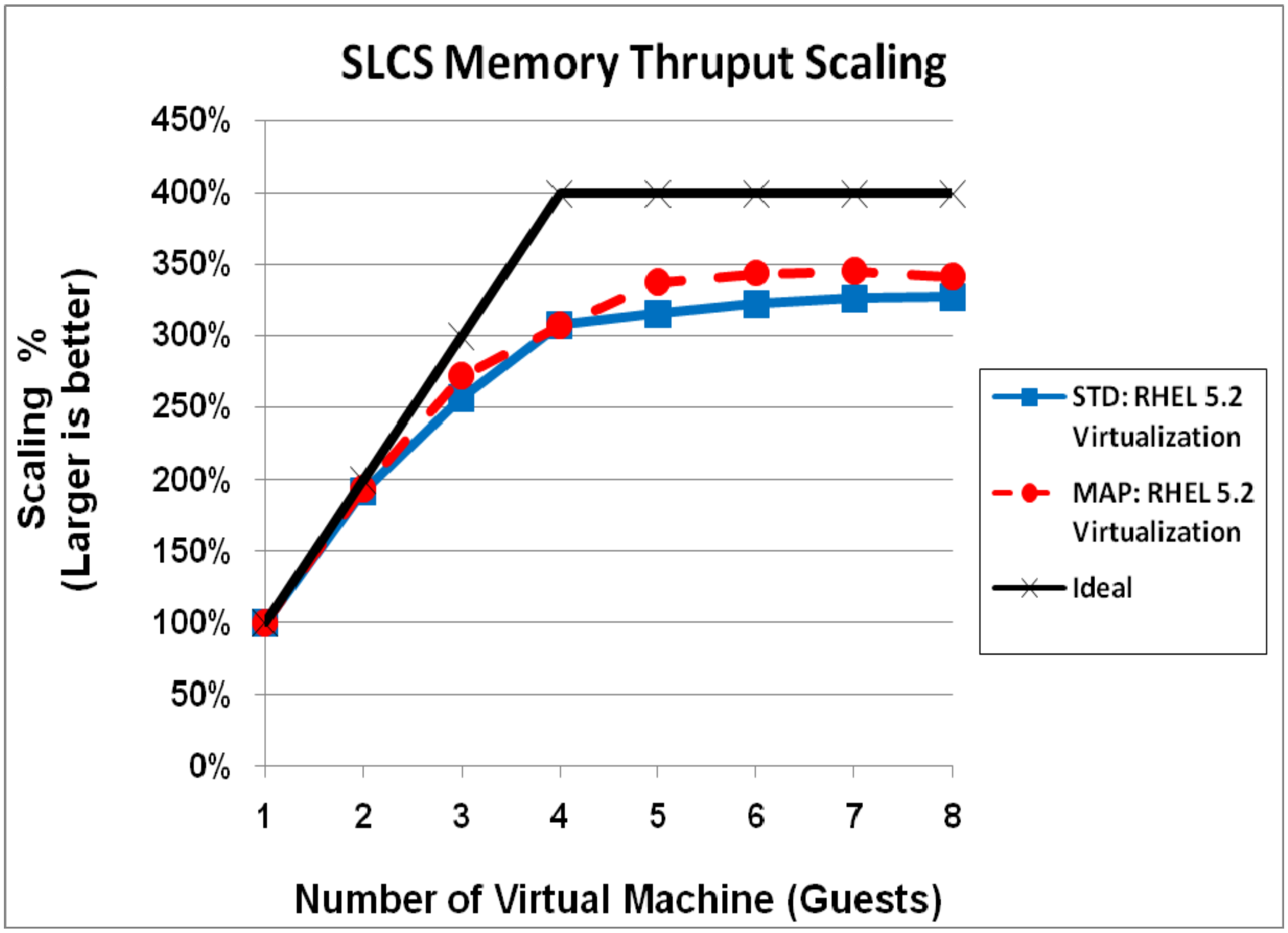

Figure 7

Figure 8 compares the'memory throughput' performance of an single SLCS instance running on:

1. Bare metal
2. Dom0
3. RHEL 5,3 PV Guest

On the 16-core server used, the performance of Dom0 is somewhat lower than expected because Dom0 is bound to 4 fixed processor cores. While the PV Guest is not bound to 4 fixed processor cores.

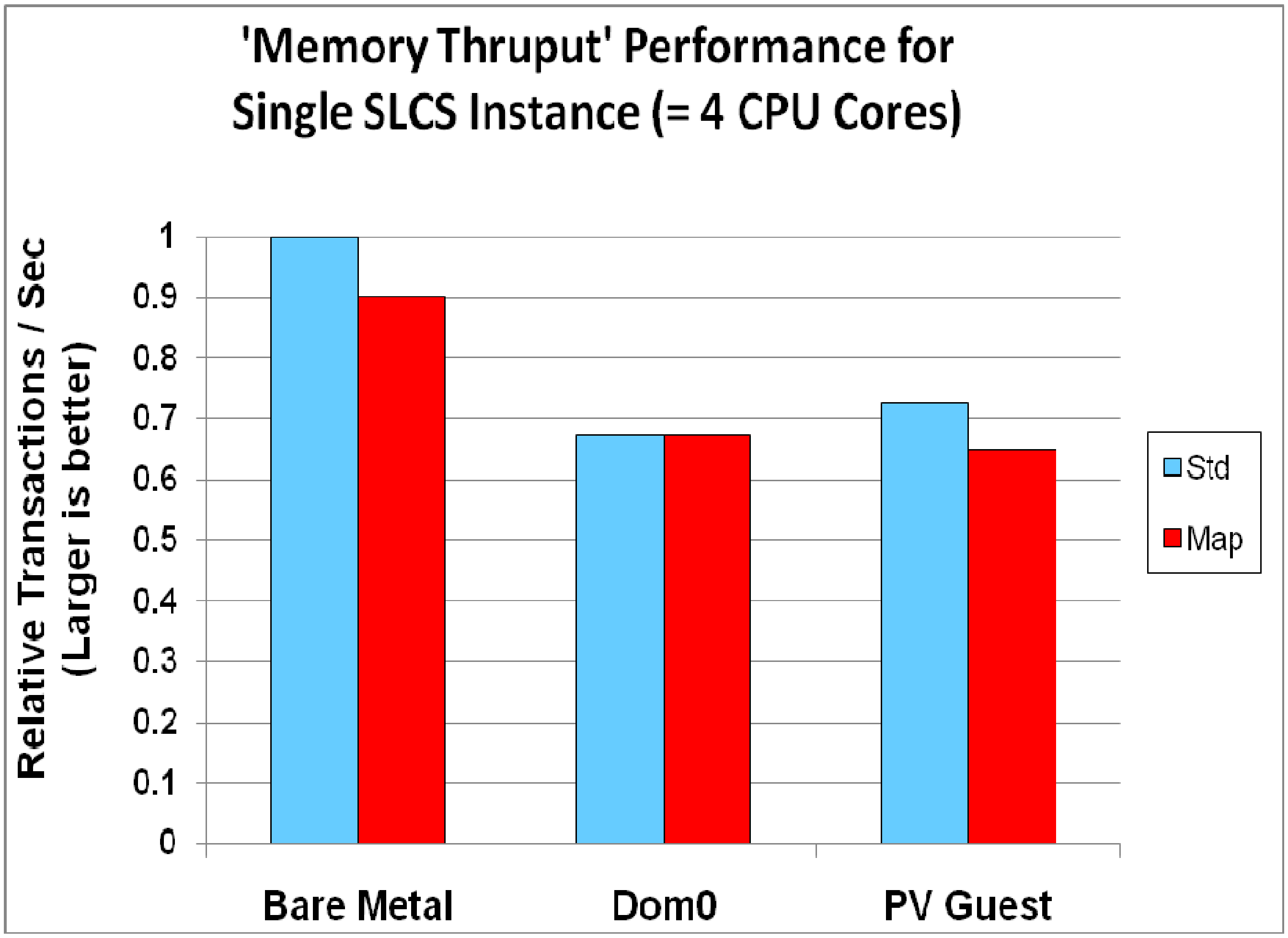

Figure 8

7. Conclusions & Next Steps

In January / February 2008, SAP conducted a Linux Virtualization Certification Workshop to demonstrate the readiness of different technology partners to run SAP instances in virtual servers. As a result of this workshop SAP has affirmed that Red Hat Enterprise Linux (RHEL) Virtualization is ready for production use with SAP:

- Scales very well (close to linear) up to 100% host allocation (= 16 vCPUs on a 16 core system)
- Handles 100% over-subscription (= 32 vCPUs on a 16 core system) very well with no collapse in performance
- Uses the available I/O bandwidth extremely well

Red Hat Enterprise Linux Virtualization (RHEL 5.2 / Xen 3.1) displayed exceptional scalability. The numbers here represent out-of-box performance with minimal tuning. The tests were run in an impressively short period of time – 1 day to set up and 2 days to run the tests. Most of the latter time was in the elapsed time it took to run the tests.

SAP instances running in virtual servers, 4 vCPUs each, were added until the physical server was 100% over-committed, i.e., 32 vCPUs running on 16 physical cores. The throughput started leveling off once the physical server was 100% committed. But it continued to scale gracefully without ever dropping in performance as shown in Figure 6 and Figure 7.

RHEL virtualization delivered between 25% and >100% higher aggregate throughput than competing virtualization technology.

Since the CPU related measurements of SLCS were not run, the goal of follow-on work will include experiments to run instances of the SAP SD (Sales and Distribution) benchmark in well tuned virtual servers using RHEL 5.2 virtualization on an IBM System x™ server and have the results certified / validated by SAP and then publish the results.

Other areas that merit testing in the future include:

- Impact of new memory related CPU technologies like AMD's Nested Page Tables (NPT) and Intel's Extended Page Tables (EPT), which could further improve the speed of memory operations.
- Relative performance of guests using para-virtualization (used here) versus full-virtualization technologies.

8. References

1. SAP Linux Virtualization Workshop by Hannes Kuehnemund, SAP Linux Lab, March 7, 2008

2. Documentation for SAP LinuxLab Certification Suite (SLCS) 2.3 Release 0.2 by Hannes Kuehnemund, SAP Linux Lab, August 13, 2007

9. Appendix I: IBM System x3850 M2

Highlights

- Outstanding commercial application serving and database performance
- Balanced system design supports more applications and help clients manage expanding business needs
- Mission-critical availability to help prevent interruptions in day-to-day enterprise operations
- High-bandwidth, low-latency memory subsystem built on DDR II registered DIMM technology
- Go green and save through server memory technology using less power and new, high-efficiency power supplies

Built on the next generation of Enterprise X-Architecture® servers, the IBM System x3850 M2 takes performance, efficiency and reliability to the next level. Featuring an unmatched combination of x86 performance and scalability with a balanced design, the x3850 M2 delivers unrivaled reliability, providing confidence in your IT solution deployments. An easy upgrade path provides the necessary flexibility to deliver an optimized solution for scale-up database, enterprise applications and server consolidation through virtualization services.

Product features	**Hardware summary**
• Fourth-generation Enterprise X-Architecture chipset design enhancements deliver performance, balanced design and proven reliability that customers have come to expect from this class of server. • Active Memory™ with features such as Memory ProteXion™, hot-add memory and Chipkill™ memory provide a level of reliability and availability that helps reduce downtime and maintain data integrity • Two times the memory availability than previous generations with 32 DIMM slots, running DDR II PC2-5300 creates a more balanced total system design. • Ability to upgrade to scalable system with the ScaleXpander Option kit allowing customers the flexibility to scale as their business needs change.	• 4-processor, 4U rack-optimized enterprise server with Intel Xeon Series 7200 and 7300 processors. • Dual-core and quad-core processor options with up to 2.4 GHz (dual-core) and 2.93 GHz (quad-core) speeds with up to 8MB L2 cache. • Seven PCI-Express x8 high-performance I/O expansion slots, two support hot-swap capabilities. • ScaleXpander Option kit offers customers the flexibility to scale from 4 sockets all the way to 16 sockets.

ScaleXpander accommodates growth

The x3850 M2 is a traditional 4-socket server that provides organizations an uncomplicated, cost-effective option—yet easily accommodates growing businesses that require increased application performance. The ScaleXpander Option kit delivers the ability to scale from 4 sockets to 8 sockets and up to 16 sockets with the flexible XpandOnDemand design; it's a pay-as-you-grow offering that provides more flexibility than the previous x3850, so organizations have even more control over IT costs. And the system's balanced design enables organizations to operate efficiently and save money on initial server configurations without having to make up-front decisions about scalability and long-term protection of their IT investments. When clients add a second chassis, the x3850 M2 doubles the memory availability and I/O capability and nearly doubles processing power.

More power for additional applications

The enterprise server's balanced design offers more processing power with quad-core processors and double the number of memory slots, allowing it to handle an increased number of applications. In addition to keeping the ratio of processor core to memory addressability balanced, the x3850 M2 offers flexibility to scale—providing access to more processors, memory or I/O necessary for workloads.

Consistent reliability

With the x3850 M2, organizations can consolidate an increased number of applications on one server with a higher threshold for protecting data. And with enhanced memory subsystems, it can help eliminate bottlenecks to operate quickly, efficiently and seamlessly.

Designed with mission-critical availability in mind, the x3850 M2 offers advanced Active Memory™ features, including:

- Memory ProteXion™ to help prevent data loss. An advanced design provides a deeper level of diagnostics to help keep the memory up and running longer while enabling data integrity.
- IBM Chipkill™ memory to help correct multiple, single-bit errors using off-the-shelf DIMMs
- Memory Mirroring with hot-swap support to help protect data through the ability to write simultaneously to independent redundant memory cards
- Advanced Buffer eXecution (ABX) Provides buffer on board technology to help resist chip failure to improve availability and reliability while delivering 37% less power consumption and up to 18% reduced latency

Unparalleled efficiency

Today, organizations demand powerful IT architecture with attractive cost benefits. The x3850 M2 is not only efficient in terms of scalability and utilization, but it also leverages standard efficiency features with additional service enhancements.

Key advantages:

- Unparalleled utilization—organizations can take advantage of virtually every aspect of their system while optimizing for a cost-effective solution
- Improved power management, operating at a lower wattage and delivering 37% less power consumption using DDR II memory
- Enhanced energy efficiency to help reduce both power and business costs; operations help you go green and save
- Advanced diagnostics with Dynamic System Analysis for ease of use, allowing clients to find and fix problems faster and more easily than previous IBM System x™ generations

Additional features and enhancements:

- IBM PowerExecutive™ 2.0 helps monitor and cap power consumption to improve energy efficiency and lower costs

- Rear access power supply helps serviceability by allowing clients to replace the power supply without opening up the system
- Improved streamlined firmware makes the process of upgrading faster