J Digit Imaging (2011) 24:883–889
DOI 10.1007/s10278-010-9358-6

Virtual Machine Performance Benchmarking

Steve G. Langer · Todd French

Published online: 5 January 2011

Abstract The attractions of virtual computing are many: reduced costs, reduced resources and simplified maintenance. Any one of these would be compelling for a medical imaging professional attempting to support a complex practice on limited resources in an era of ever tightened reimbursement. In particular, the ability to run multiple operating systems optimized for different tasks (computational image processing on Linux versus office tasks on Microsoft operating systems) on a single physical machine is compelling. However, there are also potential drawbacks. High performance requirements need to be carefully considered if they are to be executed in an environment where the running software has to execute through multiple layers of device drivers before reaching the real disk or network interface. Our lab has attempted to gain insight into the impact of virtualization on performance by benchmarking the following metrics on both physical and virtual platforms: local memory and disk bandwidth, network bandwidth, and integer and floating point performance. The virtual performance metrics are compared to baseline performance on "bare metal." The results are complex, and indeed somewhat surprising.

S. G. Langer (✉) · T. French
Mayo Clinic,
Rochester, MN, USA
e-mail: langer.steve@mayo.edu

Keywords Computer hardware · Computer systems · Computers in medicine

Background

Virtual computing is a term that describes the concept of running one or more virtual computers (aka machines) on top of a single physical computer; the virtual machines (VMs) do not interface directly with any real hardware, but rather software mimics the real hardware that the virtual host provides [1, 2]. The attractions of virtual computing are many: reduced costs, reduced resources, and simplified maintenance. However, there are potential areas where virtual computers may not be advisable. High performance/speed requirements will have to be carefully considered if they are to be executed in an environment where the running software has to go through multiple layers of device drivers before reaching the real disk or network interface.

Why would the readers of this journal be interested in virtual computing? In the current economic environment, it can be challenging to obtain new physical resources. A department administrator may find it easy to deny a researcher a new physical server if the request competes with more clinically related funding requests. Perhaps an investigator has to choose between a desktop system that will be needed for office-related tasks (grant writing, reports, etc.) or a computer server on a different operating system to do the actual work. Alternatively, a given laboratory may face space and electric power constraints; in our case, the mission assigned to our lab includes maintaining test systems for change management of all our clinical viewing systems. This task alone translates to over 20 servers and does not address the research and development work we do. Our lab has neither the space nor power for 20 physical servers, but we did have the space for two 12 processor servers with 32 GB of memory each and separate redundant storage. However, the decision of how to use those resources is not at all obvious; certainly all VMs could be hosted on one platform, but will that one platform offer adequate performance for all the VMs?

To quantify the extent to which virtualization harms performance, it is useful to break down the constituents of performance. Basically a physical computer program in the midst of intense calculations makes use of at least local memory and processor resources; it may also use local disk and network resources. A VM has the same resources, except they are software "devices" that in turn may be layered on top of:

a) A "thin" hypervisor (VM host environment) that lies directly on the physical hardware (i.e., bare metal) or
b) A "thick" hypervisor that lies on top of a host operating system that lies on the physical hardware

The figure makes this clearer; the first shows a classic physical computer with the OS residing directly on the physical hardware (i.e., "bare metal"), the second shows a thin hypervisor which in turn hosts the user OS', and finally, shows a physical machine hosting a common OS which then hosts the hypervisor which then hosts the VM.

In this work, we endeavor to measure the following metrics across various combinations of virtual machine environments and host operating systems: local memory and disk bandwidth, network bandwidth, and integer and floating point performance.

Methods

The measurement hardware consisted of two identical Dell 690 workstations: 1 Gbit network interface, 8 GB of RAM, 15,000 RPM SCSI disks, and a 2.3-GHz quad-core processor (Dell Corporation, Round Rock TX). For the file server, we chose FreeNAS Version 7 (http://freenas.org), which is an optimized open source file server appliance based on 64-bit FreeBSD (http://freebsd.org). The two computers shared a private Gbit switch (Cisco Systems, San Jose CA). The computer in the client role ran various host operating system configurations as follows:

a) Windows7, 32 bit (Microsoft Corporation, Redmond, WA)
b) Windows7, 64 bit (Microsoft *op cit*)
c) Windows 2008 Server, 64 bit (Microsoft *op cit*)
d) Redhat Enterprise Linux V5.4, 64 bit (Red Hat Corporation, Raleigh, NC)
e) OpenSolaris V2009.06 (Sun Microsystems, Santa Clara, CA).
f) Fedora Core Linux V13.0, 64 bit (http://fedoraproject.org/)

The virtualization products trialed included:

a) VMWare Player 7 (VM Ware Inc., Palo Alto CA)
b) VMWare ESXi Server V 4.0 (VM Ware *op cit*)
c) Sun Virtual Box V3.1.2 (Sun Microsystems, *op cit*).
d) Red Hat KVM V5.4 (Red Hat *op cit*)
e) Xen (Citrix Systems, Fort Lauderdale, FL).

To standardize the measurement procedure, we built a suite of measurement tools on top of a minimalist instantiation of RedHat V5.5 32 bit. A 32-bit VM was chosen as the benchmark platform for portability, a 32-bit VM can run on either a 32- or 64-bit host OS while the converse is not true. This is important because a cost-sensitive user running a virtual environment on Microsoft tools may not be able to afford the additional charges that are incurred for that company's 64-bit high performance products. Using this base "appliance," we crafted a suite of tests that measures:

a) Local memory bandwidth
b) Local disk bandwidth
c) Network disk bandwidth over the commonly used (in Microsoft Windows) Common Internet File System protocol
d) Network web interface bandwidth over the Hypertext Transfer Protocol
e) Local central processing unit (CPU) integer performance
f) Local CPU floating point performance

Figure 2 in the Appendix shows the script that automated all the tests and reported the results to a file. File Input/Output performance was measured using the "dd" command that is standard in Linux. Integer performance was measured using the Dhrystone-2 benchmark compiled with the following "sh dry.c" [3, 4]. Floating point performance was measured using the Whetstone benchmark compiled with the following switches "cc whets.c –o whets -02 –fomit-frame-pointer of –ffast-math –fforce-addr –lm -DUNIX" and activating the setting for double precision [5, 6]. More modern benchmarks exist; the reader may be familiar with "SPECInt," the newer "SPEC CPU," or other offerings from Standard Performance Evaluation Corporation (Warrenton, VA) [7]. However, these tools are not free, while the source code for the older Dhrystone and Whetstone metrics is.

Having built the appliance, we installed it on a flash drive to measure "bare metal" performance; the client computer booted from the flash device and ran the test suite completely in system random access memory (RAM). The resulting performance figures represent the baseline performance possible in the "bare metal" configuration of an operating system running directly on top of the client computer physical hardware. We then reconfigured the client computer with various host operating systems, which in turn hosted various vended virtual computer environments. The base flash drive image was then used to create VMs in each of the virtual computing products. In all cases, the VM implementations consisted of:

- Single 32-bit CPU
- 2 GB virtual SATA disk
- Virtual network card interface with 1 Gbit bandwidth to the physical router
- 1,024 MB of RAM

The physical image was similar with the exception that the total system RAM was available to the 32 bit appliance kernel running on the native processor. The results are compiled in the next section.

Results

As suggested by Fig. 1, the results can be broken out into three groups based on whether the test appliance was operated on bare metal, a thin hypervisor, or a thick hypervisor residing on a host OS. The accumulated results are tabulated in Table 1.

The following discussion summarizes key points. For example, read/write (R/W) performance in RAM, local disk, and network disk comprise three distinct areas, and the winner is not consistent.

RAM Performance

Write winner: Virtual Box on Windows7, 64 bit (127% of bare metal performance)
Read winner: Virtual Box on Windows7, 32 bit (86% of bare metal)

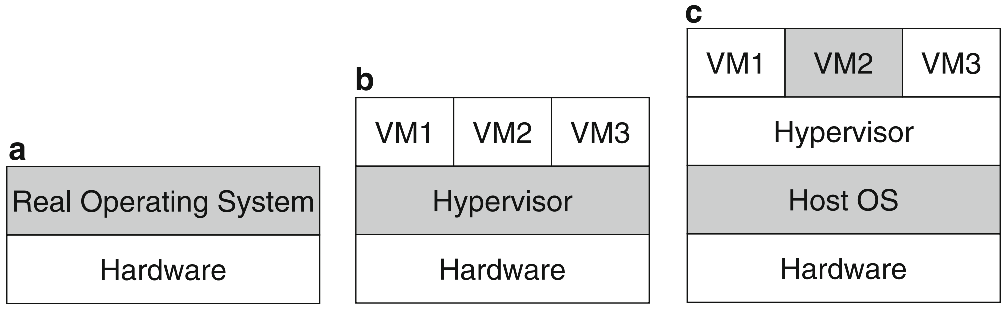

Fig. 1 **a** Conceptual view of a real physical computer, showing the real operating system directly on top of the physical computer hardware. Nominally, this should provide the best possible performance because the Operating System software is in direct control of the hardware without any software intermediaries. **b** In this figure, the virtual machine environment (hypervisor) lies directly on the physical hardware. The purpose of the hypervisor is to provide virtual resources to the virtual machines (built on familiar operating systems such as Windows). Because it is not meant to be used by humans directly, the hypervisor can be very lean and ignore aspects like a graphical user interface. However, the hosted virtual machines (VM1-3) now have an intermediate software layer between themselves and the physical hardware. **c** Finally, some hypervisors (i.e., VMWare Workstation or Sun Virtual Box) are meant to be used on top of other popular operating systems. This is obviously the most complex arrangement and may challenge performance in the VM that lives on top of the stack, as it has to traverse several layers of device drivers to reach physical hardware

Table 1 The results are grouped by coupling a single virtual environment (e.g. Sun Virtual Box) with a cluster of host OS environments

VM environment	Host OS	RAM W Mb/s	RAM R Mb/s	Local W Mb/s	Local R Mb/s	Net W Mb/s	Net R Mb/s	Web R Mb/s	Dhrystone 2 (Billions of operations/s)	Whetstone (Millions of operations/s)
Bare metal	rhel-32	641	532	519	536	10.3	54	80.2	6.0	1,085
Thin hypervisor	Xen	541	370	231	385	5.8	7.3	10.8	5.7	537
	ESXi	148	217	134	210	8.3	5.4	6.8	5.2	518
	kvm_redhat64	282	368	272	346	11.1	5.7	15.2	5.7	530
VMWare player	Win7-32	135	194	134	195	8.2	7	10.7	5.3	535
	Win7-64	152	180	144	175	9.1	7.8	15	5.7	518
	Win08 64	149	201	164	198	7.1	4.9	**17.5**	5.6	518
	redhat-64	60	265	76.4	278	9.5	**9**	13	5.3	524
	fedora13-64	169	203	150	188	9.3	8.1	15.8	5.6	510
Virtual box	Win7-32	634	**461**	**1000**	295	8.9	5.1	5.4	**10.6**	535
	Win7-64	**814**	438	524	350	9.7	7.9	19	8.8	518
	Win08 64	595	321	699	227	7.5	5.3	16.7	8.7	518
	redhat-64	629	416	232	**496**	**11.6**	3.7	4	9.1	821
	Solaris 64 bit	346	319	326	206	9	7	7.9	8.5	**852**
	fedora13-64	542	329	716	206	9.9	8.2	15.8	10.1	518

The remaining columns specify a particular performance aspect (e.g., read and write performance in MB/s, etc.). The Drhystone 2 benchmark is a measure of how many billion integer operations can be performed per second, the Whetstone is a similar metric for floating point performance in millions of operations per second. The values in *bold* are the peak performance value among the virtual environments for that metric

Local Disk Performance

Write winner: Virtual Box on Windows7, 32 bit (192% of bare metal)
Read winner: Virtual Box on Red Hat Linux (93% of bare metal)

Network Performance

Write winner: Virtual Box on Red Hat (112% bare metal)
Read winner: VM Ware Player on Red Hat Linux (17% of bare metal)

Web Read Performance:

Read winner: VM Ware Player on Windows Server 2008 64 bit (22% of bare metal)

CPU Integer Performance

Winner is Virtual Box on Windows7, 32 bit (175% of bare metal)

CPU Float Performance

Virtual Box on Solaris 64 bit (79% of bare metal)

Discussion

The experimental outline pursued herein is aligned with the needs of our lab and the various customers we serve. It is often the case that the lab serves as an "incubator" for departmental projects, and those that prove themselves are promoted to clinical applications that move to the official hospital data center. Because the data center has standardized on VMWare ESXi, we have found it most efficacious to perform our base development in that arena. However, one can also see that VMWare is not often the performance winner. Fortunately, free tools from VMWare (i.e., Convertor

Standalone Client) make it trivial to convert VMWare machines to Open Virtual Format which can be read by Virtual Box and Xen.

It is also a frequent requirement of our work to share our results with outside labs which are vey cost sensitive. For this reason, we chose to perform this analysis with products that may not be Free Open Source Software (FOSS), but are at least available without cost. Since we share the resulting VMs with third parties, it is also axiomatic that we must create them on platforms that are based on FOSS licenses; hence, the benchmark VM used here was based on Linux. Others could obviously replicate the current work on using a Windows VM benchmark platform; indeed, it would be interesting to see if the noted trends are reproduced.

One would expect, and indeed we certainly did, that the thin hypervisor group would be closest to bare metal results. However, the results are more complex than that, and as one can see from the preceding data, one can see that selecting the "best" VM environment depends on the target application's behavior; is it compute limited, R/W limited, or a combination of both? It was also somewhat puzzling that sometimes the Write performance (be it on RAM, local disk, or network disk) was sometimes faster on a VM then on bare metal (note the performance of Virtual Box in this regard). In retrospect, however, this should not have been so surprising. In an OS on bare metal, the Write performance is totally gated by the input/output (I/O) performance of the real OS, whereas in a VM the VM memory manager may employ newer and more efficient buffering algorithms then the real OS can when writing to a slower physical I/O system. However, this cannot be done in the case of reads; the entire path to the physical layer has to be traversed and one notes in no case does VM read performance beat that of bare metal.

Another surprising result is the Integer and Floating point performance of the Virtual Box VM verses bare metal. One may expect that a virtual environment could largely expose the CPU directly to the VM client (without the overhead of virtual device drivers inherent in disk and other I/O operations), and thus that client could approach bare metal speeds. But it is difficult to comprehend how the VM could actually best the bare metal Dhrystone 2 results—clearly there is some very clever engineering in play in the Virtual Box.

One final observation is the relatively poor across the board performance of the VMWare ESXi server compared to the other thin platforms (Xen and KVM). This may be due to ignorance of tuning on our part; but as all platforms were used "out of the box," we believe this experience would be observed by others. Another possible explanation is the difference in VM architecture. Both Xen and KVM rely on and use dedicated features in both the physical CPU and the guest OS being virtualized. This is called "para-virtualization" meaning that the VM environment performs some, but not all of the work, some of it is relegated to the physical CPU [8–12]. Obviously hardware runs faster than software, but the downside is that only newer hardware and modified OS' can be used. On the other hand, the full virtualization approach used by ESXi can run older hardware and support an unmodified OS (i.e., Windows NT and 2000), but apparently at a performance cost.

Based on the preceding one can deduce the following recommendations:

a) For applications that are highly integer compute sensitive, the best choice is Virtual Box on Windows7, 32 bit (unless longer 64 bit math is required in which case Fedora 64 bit is the winner).
b) For floating point sensitive applications, Virtual Box on either Solaris or Redhat 64 bit OS offer similar performance at about 80% of bare metal speed. The 20% penalty may be considered worthwhile, however, given the maintenance advantages that virtual machines have. Given the type of operations most often encountered in medical imaging processing (image registration, segmentation, etc.) this is the most common scenario [13].
c) For high speed network file or web serving needs, no VM result is better than about 25% of bare metal performance. Hence, VM methods cannot be recommended as a competitive replacement for physical network file servers at this time.

Conclusions

For various reasons we have found it very productive to adopt virtualization in our practice, but this direction is not without its drawbacks. In particular, read performance on local and network disk is negatively impacted as is floating point performance. Applications that are very sensitive to these requirements may not provide satisfactory performance in a network environment. Also, in contrast to expectations the best performance was often seen from a thick virtualization tool (Virtual Box) rather than the thin hypervisor environment.

Appendix

Fig. 2 The program “benchmark.pl” coordinates the tests and reporting of our Linux testing appliance. The “dd” command is used to measure Input/Output performance of files write to memory, local or remotely network disks. The Dhrystone2 and Whetstone metrics measure integer (billions of operations per second) and floating point performance (millions of operations per second), respectively

```
#!/usr/bin/perl

# benchmark.pl
###############################################
# Purpose: for benchmarking HW I/O performance
# Author:
# Usage:
#     mount a local disk in /mnt/local
#     mount a network CIFS share in /mnt/netshare
#     mount a local RAMDISK in /mnt/ram1
# Then run it as
#     benchmark.pl /path/resultfile
#
# Note: to automate the making of a RAMDISK and populate it,
#           include the below in /etc/rc.d/rc.local
#
#     /sbin/mke2fs -q -m 0 /dev/ramdisk
#     /bin/mount /dev/ramdisk /mnt/ram1
#     /bin/cp     /mnt/local/test_write2 /mnt/ram1
#

# clear out previous runs
qx {mkdir /mnt/ram1};
qx {mkdir /mnt/local};
qx {mkdir /mnt/netshare };
qx {rm /root/*ppt*};
qx {rm /mnt/ram1/test_write};
qx {rm /mnt/local/test_write};
qx {rm /mnt/netshare/test_write};
system (clear) ;

# Init for this run
# qx {mount -t cifs //strider-m/physics /mnt/netshare -o username=physics -o
password=*******};
$resultFile = @ARGV[0];
open (OUTPUT, ">$resultFile") || die "Can't make result file";

print OUTPUT "***** Local RAM write \n";
# dd writes to stderr, need to redirect to stdout
$a = qx {(dd if=/dev/zero  of=/mnt/ram1/test_write bs=1024k count=1) 2>&1};
print OUTPUT "$a\n";

print OUTPUT  "***** Local RAM read \n" ;
$a = qx {(dd if=/mnt/ram1/test_write2 of=/dev/null) 2>&1};
print OUTPUT  "$a\n";

print OUTPUT  "***** Local DIsk CIFS write \n";
$a = qx {(dd if=/dev/zero of=/mnt/local/test_write bs=1024k count=1) 2>&1};
print OUTPUT  "$a\n";

print OUTPUT  "***** Local DIsk CIFS read \n";
$a = qx {(dd if=/mnt/local/test_write2 of=/dev/null) 2>&1};
print OUTPUT  "$a\n";

print OUTPUT  "***** Network CIFS write \n" ;
$a = qx{(dd if=/dev/zero of=/mnt/netshare/test_write bs=1024k count=1) 2>&1};

print OUTPUT  "$a\n";

print OUTPUT  "***** Network CIFS read\n";
$a = qx {(dd if=/mnt/netshare/test_write2 of=/dev/null) 2>&1};
print OUTPUT  "$a\n";

print OUTPUT   "***** remote Web Read\n";
$a = qx {(wget http://rilcloud1:82/mayo-talk2.ppt) 2>&1};
print OUTPUT  "$a\n";

print OUTPUT  "****** Dhrystone ******\n";
print "****** Dhrystone ******\n";
$a = qx {(dry2-wor) 2>&1};
print OUTPUT  "$a\n";

print OUTPUT  " ***** Whetstone ******";
print " ***** Whetstone ******\n";
qx {whets};
$a = qx {(cat /root/whets.res) 2>&1};
print OUTPUT  "$a\n";

print " ***** DONE \n";
close (OUTPUT);
# qx {umount /mnt/netshare};
exit (0);
```

References

1. Langer S, Charboneau N, French T: DCMTB: a virtual appliance DICOM toolbox. J Digit Imaging 2009 Aug 25. [Epub ahead of print] PMID:19705204. doi:10.1007/s10278-009-9230-8
2. Smith JE, Nair R: The architecture of virtual machines. Comput IEEE Comput Soc 38(5):32–38, 2005. doi:10.1109/MC.2005.173
3. Dhrystone 2. http://www.netlib.org/benchmark/dhry-c. Last viewed February 2010
4. Weiker R: Dhrystone: a synthetic systems programming benchmark. Commun ACM 27(10):1013–1030, 1984
5. Whetstone. http://se.aminet.net/pub/benchmark/aburto/whetstone. Last viewed February 2010
6. Curnow HJ, Wichman BA: A synthetic benchmark. Comput J 19 (1):43–49, 1976
7. Standard Performance Evaluation Corporation: The SPEC BenchmarkSuite. Technical report. http://www.spechbench.org. Last viewed August 2010
8. VMWare: http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf Last viewed May 2010
9. Xen Wiki: http://en.wikipedia.org/wiki/Xen. Last viewed May 2010
10. Chen W, Lu H, Shen L, Wang Z, Xiao N, Chen D: A novel hardware assisted full virtualization technique. Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for Young Computer Scientists, pp.1292–1297, 18–21, Nov. 2008 doi:10.1109/ICYCS.2008.218
11. Whitaker A, Cox RS, Shaw M, Gribble SD: Rethinking the design of virtual machine monitors. Computer 38(5):57–62, 2005. doi:10.1109/MC.2005.169
12. Chaudhary V, Minsuk C, Walters JP, Guercio S, Gallo S: A comparison of virtualization technologies for HPC. Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on Advanced Information Networking and Applications, pp. 861–868, 25–28, 2008 doi:10.1109/AINA.2008.45
13. Yoo TS, Ackerman MJ, Lorensen WE, Schroeder W, Chalana V, Aylward S, Metaxas D, Whitaker R: Engineering and algorithm design for an image processing API: a technical report on ITK—the insight toolkit. Stud Health Technol Inform 85:586–92, 2002