Reprinted from the

Proceedings of the Linux Symposium

July 23rd–26th, 2008
Ottawa, Ontario
Canada

A Survey of Virtualization Workloads

Andrew Theurer
IBM Linux Technology Center
habanero@us.ibm.com

Karl Rister
IBM Linux Technology Center
kmr@us.ibm.com

Steve Dobbelstein
IBM Linux Technology Center
steved@us.ibm.com

Abstract

We survey several virtualization benchmarks, including benchmarks from different hardware and software vendors, comparing their strengths and weaknesses. We also cover the development (in progress) of a new virtualization benchmark by a well known performance evaluation group. We evaluate all the benchmarks' ease of use, accuracy, and methods to quantify virtualization performance. For each benchmark, we also detail the areas of a virtualization solution they stress. In this study, we use Linux where applicable, but also use other operating systems when necessary.

1 Introduction

Although the concept of virtualization is not new [1], there is a recent surge of interest in exploiting it. Virtualization can help with several challenges in computing today, from host and guest management, energy consumption reduction, reliability, and serviceability. There are now several virtualization offerings, such as VMware® ESX [2], IBM PowerVM™, Xen [3][4] technology from Citrix, Virtual Iron, RedHat, and SUSE, and Microsoft® Windows Server® 2008 Hyper-V [5]. As the competition heats up, we are observing a growth of performance competitiveness across these vendors, yielding "marketing collateral" in the form of benchmark publications.

1.1 Why are Virtualization Benchmarks Different?

A key difference in benchmarking a virtualization-based solution is that a hypervisor is included. The hypervisor is responsible for sharing the hardware resources for one or more guests in a safe way. The use of a hypervisor and the sharing of hardware can introduce overhead. One of the goals of benchmarking virtualization is to quantify this overhead and ideally show that virtualization solution X has lower overhead than virtualization solution Y. Another difference in benchmarking virtualization is that the benchmark scenarios can be very different than one without virtualization. Server consolidation is such a scenario. Server consolidation may not typically be benchmarked without the use of a hypervisor (but not out of the realm of possibility; for example, containers may be used). Server consolidation benchmarks strive to show how effective a virtualization solution can host many guests. Since many guests can be involved in this scenario, it may require the use of several benchmarks running concurrently. This concept is not common on traditional benchmarks.

2 Recently Published Benchmarks

The following are virtualization benchmarks with published specifications and run rules that users can replicate in their own environment. These benchmarks strive to set a standard for virtualization benchmarking. In this section, we discuss the strengths and weaknesses of these benchmarks.

2.1 vConsolidate

The vConsolidate benchmark [6] was developed by Intel® to measure the performance of a system running consolidated workloads. As one of the earlier proposals for a virtualization benchmark, vConsolidate was written to prompt the industry to discuss how the performance of a system running with virtualization should be measured.

vConsolidate runs a benchmark for a web server, a mail server, a database server, and a Java™ server, each in a separate guest. There is also a guest that runs no benchmark, which is meant to simulate an idle server. These five guests make up a *consolidation stack unit*, or CSU, as illustrated in Figure 1.

The tester starts with running 1 CSU, obtaining the benchmark score and the processor utilization of the

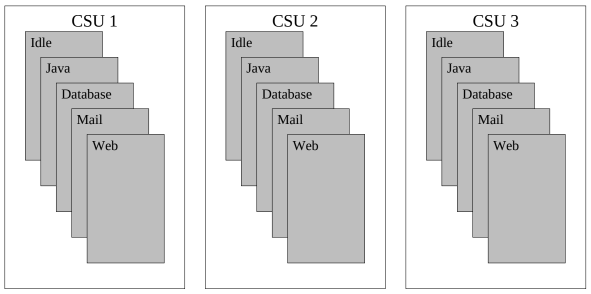

Figure 1: Consolidation stack units

system. The tester does three iterations and then uses the median score and its processor utilization. The tester incrementally adds additional CSUs, recording the benchmark score and processor utilization, until the benchmark score for the set of N CSUs is less than the score for N-1 CSUs, or until all the system processors are fully utilized. The final benchmark score is the maximum of the scores reported along with the number of CSUs and the processor utilization for that score.

The vConsolidate benchmark score is calculated by first summing each of the component benchmark scores across the individual CSUs. The sums are then normalized against the score of the benchmark running on a reference system, giving a ratio of the sum compared to the score of the reference platform. The reference system scores can be obtained from a 1 CSU run on any system. It is Intel's desire to define a "golden" reference system for each profile. The vConsolidate score for the test run is the geometric mean of the ratios for each of the benchmarks. Figure 2 shows sample results from a vConsolidate test run. In this example, the maximum score was achieved at 4 CSUs with a processor utilization of 78.3%.

The reporting of the processor utilization along with the score is not common. Most standard benchmarks simply report the benchmark score and are not concerned with the processor utilization. The processor utilization, however, is a useful metric in characterizing the performance of the system running the consolidated workloads. It can also be useful in spotting performance issues in other areas of the system (for example, disk I/O, network), for example, when the score starts dropping off before the processors get fully utilized, as seen in Figure 2.

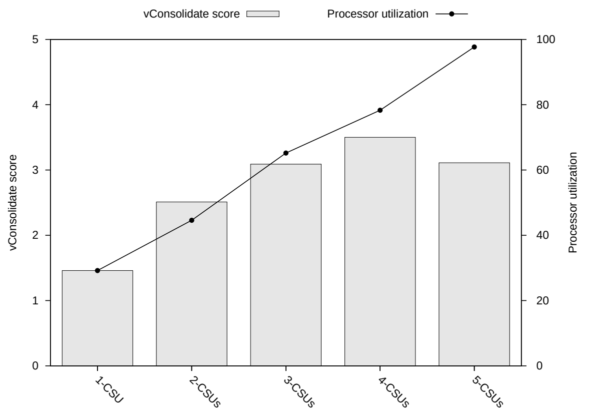

Figure 2: Sample results for vConsolidate

2.1.1 Benchmark Implementation

vConsolidate specifies which benchmarks are run for each of the workloads: WebBench™ [7] from PC Magazine for the web server, Exchange Server Load Simulator (LoadSim) [8] from Microsoft for the mail server, SysBench [9] for the database server, and a slightly modified version of SPECjbb2005 [10] for the Java server.

WebBench is implemented with two programs—a controller and a client. The controller coordinates the running of the WebBench client(s). vConsolidate uses only one client program, which runs eight engine threads. The client and the controller can be run on the same system because neither is processor intensive. The only interface to WebBench is through its GUI, making it difficult to automate.

vConsolidate indirectly specifies which mail server to run. Microsoft's LoadSim only works against a Microsoft Exchange Server, therefore, the mail server must be Exchange Server running on the Windows operating system. Although it runs in a GUI, LoadSim can be easily automated because it can be started from the command line.

vConsolidate runs a version of SysBench that has been modified so that it prints out a period after a certain number of database transactions. The output is then redirected to a file, which is processed by vConsolidate to calculate the throughput score.

vConsolidate has instructions for modifying the SPECjbb2005 source to add a think time, so that the test doesn't run full bore, and to have it print out some statistics at a periodic interval. The output is then redirected to a file that is processed by vConsolidate to calculate the throughput score. The benchmark specifies the version of Java to run: BEA® JRockit® 5.0.

One observation of the benchmark implementation is that most of the workloads are modified or configured such that the benchmark does not run all out, but simulates a server running with a given load. LoadSim is configured to run with 500 users. SysBench is configured to run only four threads. SPECjbb2005 is modified to add a delay. However, WebBench is not modified to limit its load. The delay time and think time are both zero. It may be that this is an oversight of the benchmark configuration. Or it may be that even with no delay and no think time that WebBench does not generate enough load to consume the network and/or processor utilization on the server. That is, WebBench may generate a moderate load even with the delay and think time set to zero.

Certain components of vConsolidate are portable to other applications and other operating systems. WebBench makes standard HTTP requests, consequently it doesn't matter which web server software is run nor the OS on which it runs. The POSIX version of SysBench has support for MySQL, PostgreSQL, and Oracle® database, making it conceivable that vConsolidate could be easily modified to use those databases. SPECjbb will run on any OS that has a Java Virtual Machine. SPECjbb will also run on any Java Virtual Machine (JVM), so although vConsolidate specifies JRockit 5.0, it is conceivable that it could run with any other JVM. Other components of vConsolidate are not portable, e.g., the mail benchmark is LoadSim which requires Microsoft Exchange Server. vConsolidate could be made more portable by not requiring specific software, e.g., JRockit 5.0, and by using more portable, industry standard bench marks, such as SPECweb2005 for the web server and SPECmail2008 for the mail server.

However, vConsolidate has achieved its purpose of getting the discussion started on benchmarking virtualization. Intel is not concerned with trying to make vConsolidate and industry standard benchmark. Comments on how to improve the benchmark are now being fed to the Standard Performance Evaluation Corporation (SPEC) and they are developing an industry standard benchmark for virtualization. In fact, the portability issue is still up for debate among the SPEC members. Some are of the mind that the benchmark should specify the software and benchmarks used so that fair comparisons can be made between different hardware platforms. Others see that specifying the software and benchmarks favors the software selected, does not allow for comparisons to be made between software stacks, and reduces the concept of the benchmark being open.

2.1.2 Running the Benchmark

As mentioned above, the test setup requires client machines to run LoadSim and to run the WebBench controller and client. In the current version of vConsolidate (1.1), each client runs one instance of LoadSim and one instance of the WebBench controller and client. Essentially, one machine runs all of the client drivers for one CSU. This is an improvement over the previous version of vConsolidate that specified separate clients for LoadSim and WebBench. Having one client machine per CSU makes for an easier test setup, not to mention the savings in hardware, energy, and rack space.

The test setup also requires one machine to run the vConsolidate controller. The controller coordinates the start and stop of each test run so that all benchmarks start at the same time via a daemon that runs on the LoadSim and WebBench clients, and on the database and Java servers. The controller kicks off the LoadSim clients first and then delays some time to let the LoadSim clients finish logging into the mail servers before starting the other benchmarks. The controller collects the benchmarks' results when the run is complete.

The vConsolidate controller, LoadSim client, and WebBench controller and client are all Windows applications. As with the servers, this prevents a person from running a single OS other than Windows for the test bed. When testing a Linux environment one must still deal with Windows clients.

vConsolidate has nice automation of the benchmarks. The Linux daemons kickoff and kill a `run.sh` script to start and stop the test, respectively. The `run.sh` script can can have any commands in it. We were able to add commands to `run.sh` to kickoff and kill profilers. The Windows daemon is capable of starting and stopping LoadSim and is able to press the "Yes" button on the WebBench GUI to start WebBench.

On the other hand, vConsolidate itself is hard to automate. Much of this arises from a common mentality when writing a Windows application, which is to assume that everything is controlled by a user sitting in front of the screen. Many Windows benchmarks are written with GUIs that are nice for user interaction but terrible for automating the benchmark. We have a sophisticated, well-developed automation framework for running benchmarks that allows us to, for example, kick off a batch of benchmarks to be run over night and come back in the morning and look at the results. It is difficult to run GUI-based benchmarks from the scripted automation framework.

The vConsolidate controller uses a GUI interface. Thus, you cannot automate a test run suite of 1, 2, and 3 CSUs, for example. Each test run must be started by a user pressing the "Start!" button on the vConsolidate GUI. It would be nice if vConsolidate could be started from a command line, thus enabling it to be scripted.

Our test runs of vConsolidate used a version prior to 1.1 which required the user manually to press a "Stop!" button when the WebBench clients finished. The tester had to keep an eye on the WebBench tests and when they finish, go to the vConsolidate controller and push the "Stop!" button. If the tester was not alert, he could miss the end of the test run and end up with bad results because the other benchmarks would have continued to run beyond the end of the WebBench test and would have logged throughput numbers including the time when WebBench was not running. We managed to automate the test termination by making use of Eventcorder™ [11], a Windows program for automating keyboard and mouse input. vConsolidate version 1.1 addressed this issue and will stop the tests by itself.

As mentioned above, vConsolidate uses WebBench to test the web server. WebBench is also a GUI-based Windows benchmark. For each test run, and for each client, the tester must manually setup the test on the WebBench controller up to the point of clicking the "Yes" button to start the test. The vConsolidate controller does coordinate the pushing of the Yes button on all the WebBench controllers at the same time so the tests start at the same time. However, the tester must manually setup each WebBench controller before each test run. This could be avoided by using a more recent benchmark (WebBench is six years old) that is not GUI-based, such as SPECweb2005.

vConsolidate is still a work in progress. Some issues, such as test termination, have been addressed. Other issues have been deferred to the SPEC virtualization work group. We hope to see many of the issues raised here addressed in future releases of the benchmark.

2.2 VMmark

We must note up front that we have not had any hands-on experience with VMmark™. The following analysis is based on the paper *VMmark: A Scalable Benchmark for Virtualized Systems* [12].

VMmark was developed by VMware®. VMmark was the first performance benchmark for virtualization. Like vConsolidate, VMmark is a benchmark to measure the performance of a system running consolidated workloads. And like vConsolidate, VMmark runs a benchmark for a web server, a mail server, a database server, and a Java server, and has an idle server. VMmark adds another benchmark for a file server. In VMmark's terminology, the combination of the six guests is called a *tile*.

VMmark constrains each of its benchmarks so that it runs at less than full capacity so that it emulates a server that is usually running at less than full capacity. A given tile should then generate a certain amount of load on the system. The tester starts with a test run on a single tile and then incrementally adds tiles until the system is fully utilized.

VMmark specifies which benchmarks to use: Exchange Server Load Simulator (LoadSim) from Microsoft for the mail server, a slightly modified version of SPECjbb2005 for the Java server, SPECweb2005 [13] for the web server, SysBench for the database server, and a slightly modified version of dbench [15] for the file server. The paper mentions using Oracle database and Oracle's SwingBench benchmark, but the latest version of VMmark specifies MySQL and SysBench.

A normal run of LoadSim increases the number of users until a maximum is reached. The benchmark score is the maximum number of users. VMmark is concerned with maintaining a specific load, therefore it keeps the number of users set at 1000 and instead measures the number of transactions executed by the server.

VMmark uses a modified version of SPECjbb2005. SPECjbb2005 is designed to do short runs over an increasing number of warehouses. The VMmark version of SPECjbb2005 is set to run eight warehouses for a long period of time. It is also modified to report periodic status instead of a final score at the end of a run. As with vConsolidate, VMmark requires BEA® JRockit® 5.0.

VMmark uses a modified version of SPECweb2005. The VMmark version of SPECweb changes the think time from ten seconds to two seconds to generate the desired load. The run rules for SPECweb benchmark specify three separate runs, each with a warm up and a cool down period. VMmark, however, does one long run to keep a consistent load on the system. VMmark makes use of the internal polling feature of SPECweb to get periodic measurements of the number of pages accessed. VMmark runs both the SPECweb2005 backend simulator and the web server in the same guest "to simplify administration and keep the overall workload tile size down."

VMmark does not need to modify SysBench. The desired workload can be obtained by setting the number of threads. (The same is true of SwingBench. In the paper, SwingBench is configured for 100 users.)

VMmark uses a modified version of dbench. The benchmark is modified to run repeatedly so that it keeps running during the full VMmark run. The benchmark is also modified to connect to an external program that keeps track of the benchmark's progress and controls the benchmark's resource use so that it generates a predictable load. VMmark also runs a small program that allocates and mlocks a large block of memory to keep the page cache small and force most of the dbench I/O to go to the physical disk.

As with vConsolidate, VMmark requires one client machine per tile. The client machines run Microsoft Windows Server 2003 Release 2 Enterprise Edition (32-bit).

A VMmark test runs at least three hours. Periodic measurements are taken every minute. After the system has achieved a steady state, the benchmark is run for two hours. The two hours are split into three 40 minute runs. The median score of the three runs is used to calculate the score for the tile. This method has an advantage over vConsolidate, since the tester only has to start the test once instead of having to start three separate runs.

The overall benchmark score is determined by the number of tiles run and the individual scores of the benchmarks. Similar to vConsolidate, the individual benchmark scores are first normalized with respect to a reference system. The score for a tile is the geometric mean of the normalized scores for each benchmark in the tile. The overall VMmark score is the sum of the scores for each tile. This is different from vConsolidate, which sums the normalized scores across the CSUs for each benchmark and then takes the geometric mean of each of the benchmark sums for the overall score.

VMmark is less portable than vConsolidate. It specifies which operating systems, server software, and benchmarks are to be used. And, of course, it only runs on VMware® ESX Server. Apparently VMmark is only concerned with comparing hardware platforms.

It is not clear how much effort VMware will spend on updating VMmark. VMware is on the SPEC Virtualization subcommittee and is spending effort there to help build an industry standard virtualization benchmark.

3 Other Benchmark Studies

These studies focus on virtualization benchmarks that do not implement a reference standard, and did not attempt to be adopted as such. They can be considered "ad-hoc," but certainly could be adapted or replicated.

3.1 Server Consolidation on POWER5

In 2006, we conducted a study [17] to help understand the effectiveness of server consolidation using an IBM System p5™ 550 server and Linux guests. The goal was to see how many servers tasked with common services such as web, file and e-mail could be consolidated. New methods were constructed to capture representative loads for such servers and to measure the server consolidation capacity of a virtualization solution.

3.1.1 Workload Definition

A goal of this project was to devise a metric that was tailored to a virtualization solution, not just an application solution. Typically, a single benchmark is designed to be ramped-up to achieve peak throughput while maintaining a specified quality-of-service level, and the metric is a measure of throughput. This study instead used many benchmarks, each configured to inject a fixed load with no ramp-up. Similar to some server consolidation benchmarks, aggregate load for the host was increased by adding more guests and benchmarks. This study achieved this with three benchmark types, targeting a mail, web, and file server. Initially, one instance of the three benchmark types was used (concurrently), injecting load to three guests. Host load was increased iteratively by adding another instance of the three benchmark types along with three more guests. This was repeated until either 50% of host processor utilization was reached or quality of service from any benchmark instance could not be maintained.

In this study, there was a desire to have each server that was consolidated represent a load that might be typical on a stand-alone server. This was achieved by taking a common x86 server, and for each benchmark type, injecting a low load, then ramping up the load until the server reached 15% processor utilization. The load level to achieve 15% processor was our reference load. This reference load was then used to target guests on the POWER5™ system.

3.1.2 Resource Characterization

This workload exhibited a significant amount of disk and network I/O due to both file server and web server benchmarks involved. This flows logically to/from guest and its client. However, there is significant traffic between the guests and the guest designated as the Virtual I/O Server (VIOS). Unless a guest has an I/O adapter dedicated to its exclusive use, the VIOS must handle I/O requests for the guests. This requires an efficient method to move this data as well as adequate processor resources for the VIOS.

3.1.3 Issues and Challenges

This study required an effective way to ensure that the host did not exceed 50% processor utilization. This limitation was placed on this study to ensure ample headroom, should a guest or multiple guests need to accommodate spikes in load. One could have monitored system utilization tools, but there was a concern that all processor cycles may not be accounted for. For example, processor cycles used by the hypervisor, which may not be attributed to one particular guest, may not be accounted for in a host utilization tool. To avoid this problem, half the processors were disabled, ensuring no chance of exceeding 50% host processor utilization

When dealing with benchmarks that have different strategies to begin injecting load, warming up, entering a measurement period, ramping down, and stopping, a method was required that allowed one to accurately measure the load of all three benchmark types. Ideally, one would have individual benchmarks that have their measurement period coincide at the exact same time. In absence of this, one must record the results of each benchmark type individually, while ensuring the load of the other two benchmarks is the load desired and is generated throughout the entire measurement period of the first benchmark. For example, to get results for the web server benchmark, one must ensure the file and mail server benchmarks' steady state began before, and ended after, the web server benchmark's measurement period. This technique should be followed for mail and file server benchmark measurements as well. Because each of these benchmarks can align their measurement period with benchmarks of the same type, all instances of that server type can be measured concurrently. For example, if one is testing 10 sets of consolidated servers (10 web, 10 file, 10 e-mail), three passes of measurements are made. The first pass has all web benchmarks execute in unison, while mail and file benchmarks' steady state begin before, and end, after the web benchmarks' measurement period occurs. The second pass has measurements from file benchmarks, and the third pass has measurements from mail benchmarks.

3.2 LAMP Server Consolidation

One of the primary workloads that we have targeted for consolidation has been the underutilised LAMP (Linux Apache MySQL PHP/Perl/Python/etc.) servers, similar to what a web-hosting company might have. Traditionally, these types of servers are some of the lowest utilized and therefore have a high capacity for consolidation. Additionally, as detailed further when discussing the workload characteristics, these workloads stress many parts of the system stack and are therefore more representative as a generic workload than something that stresses one area.

In order to compare the consolidation capabilities of various virtualization solutions (Xen, VMware, PowerVM, and KVM) on various hardware platforms (x86, x86_64 and POWER), a consolidation benchmark was developed using a LAMP stack application and additional open-source tools. In order to ensure cross platform availability, all stack components were built from source (with the exception of the Linux kernel and the distribution being tested). The benchmark consists of two data collection steps that combine to form a consolidation metric which is the number of underutilized servers that can be consolidated as guests on a single virtualized system. The first data collection step is to run the target workload on a system that is representative of the historically underutilized servers that are to be virtualized. Existing surveys and reports by industry consulting services provide metrics such as the average processor utilization of the underutilized servers; these processor utilization metrics are used to determine the injection rate (the amount of work for the benchmark drivers to "inject" into the test system) that the client drivers should use to drive the baseline system. This injection rate combined with the workload itself defines the baseline workload that is then run simultaneously on each of the virtualized guests on the test system in the second data collection step. The more guests that a test system can host while maintaining similar performance and quality-of-service to the baseline system, the higher the consolidation metric it achieves.

In order to understand the workload characteristics that the virtualized system must be capable of handling, the characteristics of the workload when running on the baseline system must first be understood. A LAMP stack application consists of the Apache web server, the MySQL database, and an interpreted language (PHP in this particular example) all running on top of the Linux operating system. These applications inherently have properties that need to be understood.

The Apache web server accepts connections from client systems and responds appropriately depending on the request. Servicing a simple request with non-dynamic code will usually consist of reading the requested object from disk or fetching it from cache, and returning it to the client. Servicing a more complicated request, such as PHP code, will involve fetching the script from disk or cache, invoking the interpreter, and then returning the generated content to the client. All requests are logged, which consists of a sequential I/O write that will eventually be forced to flush to disk.

When the dynamic code interpreter is invoked, the execution possibilities are quite expansive, but there are some basic concepts that can be summarized: first, the dynamic code may have never been requested before, which means that it will have to be compiled before execution; second, if the dynamic code has been executed before and was not cached, such as by a PHP accelerator, it will have to be compiled again; third, in a LAMP scenario, the dynamic code will most likely involve connecting to the database and waiting for data to be returned or processed. Any compilation of dynamic code will require processor cycles so caching of the compiled code is desirable in order to reduce processor utilization.

When requests are made to the database from the dynamic code execution, a combination of reads and writes are likely, and the I/O pattern can vary depending on the operation required. Database read queries will likely result in small random, read I/O operations. Database write queries will likely consist of a combination of random I/O writes for the data and sequential I/O writes for the log.

When you combine the characteristics of the various stack components you get the following: TCP/IP socket connections handled by the web server, disk reads and writes by the web server, processor intensive compilation and execution of dynamic code, and disk reads and writes (both sequential and random) from the database engine. For an underutilized system, the magnitude of these characteristics is relatively low and of little concern, however, in a consolidation scenario that can change.

When large numbers of guests are consolidated on a single system, the workload properties of the consolidated guests are stacked on top of each other. In the case of the TCP/IP connections, due to the fact that these are lightly loaded servers being consolidated, the number of requests are quite low and therefore not an overriding factor in the test. The processor utilization is cause for some concern, but it is one of the finite resources that consolidation is trying to maximize, so it will inherently run out at some point anyway. The real area of concern is the disk I/O that the workload drives. In an ideal world of spinning disks, all I/O would be sequential in order to minimize the penalty of head seeks. Unfortunately, this is not the case, and most workloads, such as this LAMP stack application, have a mix of sequential and random I/O patterns. Virtualization exacerbates the problem, though because due to the fact that all guests have their own carved out disk space and the drive heads are forced to seek more and more with each added guest. This means that drive I/O capacity will decrease with each added guest until the I/O pattern becomes completely random, and the ability to complete requests will be bounded by the random I/O capabilities of the storage system. In some of the consolidation studies we have done, on systems with large amounts of processor power (large numbers of fast processor cores) the maximum consolidation factor could be not reached due to the I/O capacity of available storage systems being maxed out well before processor horsepower was exhausted. A reality of today's storage systems is that systems capable of high random I/O performance are quite expensive, and for workloads with large amounts of I/O, success of consolidation will depend greatly on the ability to pair the target virtualization system with the the proper storage.

3.3 Processor Scalability Workloads

For the 2006 Ottawa Linux Symposium, we conducted several scalability tests [18]. These workloads were designed specifically to measure and improve the scalability of hypervisors and the guests they manage. Unlike other workloads discussed here, these are more synthetic, and are designed to isolate and study specific scalability problems. They do not necessarily strive to represent typical virtualization scenarios, but try to target extreme scalability situations.

3.3.1 Two Types of Scalability

Typically, one would measure scalability by testing a scenario with N resources (in our case, processors), then testing again with N*M resources, and observing the relative increase in throughput. A benchmark would normally run "all out" to maximize the use of the resource. Without virtualization, this is fairly straight forward. One would boot with one processor enabled, run a test (for example dbench), record the result, then boot with N processors and run the test again. The scalability would be the N-way throughput divided by the 1-way throughput. There are, of course, variations of this theme—for example, using 1 and N sockets or NUMA nodes instead of processors. For virtualization scalability, we alter this method slightly in order to study two types of scenarios: scalability of a single guest and scalability of many guests.

3.3.2 Single Guest Scalability

Scaling just one guest involves assigning the guest one processor resource (or socket, or NUMA node), testing, then assigning the guest N processors (or sockets, NUMA nodes) and testing again. This is probably the easiest way to test virtualization, as it follows the methodology that one would use for traditional scalability testing. With just one guest, there is no plurality of benchmarks to manage and synchronize, and no sets of results to aggregate. One complication is that there may be a service and/or I/O guest which should also be accounted for.

3.3.3 Multiple Guest Scalability

For scaling many guests, we start with one guest, assign a fixed processor resource (core, socket, etc.), then test with N guests, running the same benchmark at the same time, each assigned the same resource amount (core, socket, etc.), such that we have enough guests to maximize that resource. Our goal is to analyze the scalability of the hypervisor, and not necessarily the scalability of the guests. In the previous scenario, any scalability inhibitor within the guest could affect the overall scaling, while in this scenario that is not true. In this type of scalability test, we do have a little more work to do. Because we are running many guests, one must ensure that all of the benchmarks begin and end at the same time. One must also sum the benchmarks' results into one result.

3.4 SPEC and Virtualization

The Standard Performance Evaluation Corporation (SPEC) is a non-profit corporation that creates, maintains and endorses a standardized set of benchmarks. Members of SPEC include many computer hardware and software manufacturers across the world. Recently, SPEC has formed a new sub-committee, Virtualization, to create and maintain virtualization benchmarks.

3.4.1 SPECvirt_sc2009

SPECvirt_sc2009 is the first virtualization benchmark from SPEC. As of this writing, this benchmark is still under development. The characteristics, methodologies, and run rules described here are subject to change. The SPEC virtualization sub-committee contains members who have been involved in many of the virtualization benchmarks outside of SPEC, including the benchmarks mentioned earlier in this paper. As such, SPECvirt_sc2009 takes influence from these benchmarks, both from their methodologies and from the lessons learned.

SPECvirt_sc2009 follows a server consolidation scenario, similar to other benchmarks described here. The benchmark uses the same *tile* concept as VMmark, similar to the vConsolidate CSU and server sets on the POWER5 Server Consolidation Study. The benchmark metric is the number of tiles one can run while maintaining the quality of service of all benchmarks participating. However, there are some attributes that differentiate this benchmark from the others previously mentioned

SPEcvirt_sc2009 also addresses the issue of portability. There are no restrictions on the type of architecture or operating system. All services tested can be implemented with proprietary and/or open-source solutions. The client driver also makes no requirement of architecture or operating system.

The proposed SPECvirt_sc2009 tile consists of 6 guests: a web server, mail server, application server, database server, infrastructure server, and an idle server. To drive a single tile, three SPEC benchmarks are used: SPECweb2005, SPECmail2009, and SPECjAppServer2004. SPECweb2005 injects requests to a web server guest. A back-end database simulator, or Besim, resides on the infrastructure guest, simulating database requests from the web guest. In addition, the web server has part of its document root NFS mounted from the infrastructure server, so some HTTP requests are served by just the web guest, and some involve the infrastructure guest as well. The SPECmail2009 benchmark injects requests using the IMAP mail protocol to the mail server, driving load on the mail server guest only. The SPECjAppServer2005 benchmark injects requests to the application server. The application server guest requires a database, located on the database server guest. The SPECjAppServer2004 benchmark drives load on both the application server and database server. The idle guest is used to represent the overhead of a guest which is running but not actually doing anything. It does not interact with any of the other tile components.

One of the big issues for other virtualization benchmarks, which had heterogeneous guests and load generators, was synchronization. When using the original SPEC benchmarks to prototype SPECvirt, one encounters similar issues. It is possible to configure each benchmark type such that they begin and end their measurement period at nearly the same time, however it can not be done with a high level of confidence. The SPEC virtualization sub-committee is changing these benchmarks to support a more tightly controlled apparatus, ensuring the benchmarks' start and stop times coincide exactly.

SPECvirt_sc2009 aims to introduce a characteristic which is not too prominent on most of the other virtualization benchmarks: intra-guest network communication. The benchmark introduces dependencies which will require network communication between guests within a tile. A portion of the document root for the web server guest is served by the infrastructure server guest, generating a significant amount of NFS traffic between these two guests. SPECjAppServer is required to use two guests, one for the application server and one for the database server, so that communication between the two services is between guests, compared to both services on the same guest communicating over loopback.

4 Considerations for Future Virtualization Benchmarking

After working with these benchmarks, we would like to propose some ideas for consideration when designing, running, and analysing virtualization workloads. These ideas are not strictly for benchmarking or marketing collateral, but also for general testing of virtualization.

4.1 Simplification

Many of today's benchmarks are already quite complicated. Configuring a multi-tier benchmark such as SPECweb2005 or SPECjAppServer2004, can be quite a task by itself. Combining several instances of benchmarks like this can be daunting at first. We propose leveraging the concept of virtual appliances for both the server under test and the client driver system. Maintaining a library of virtual appliances allows the user to spend more time on evaluation and analysis of the total solution instead of dealing with details of service implementation, OS configuration, and related tasks.

4.2 New Workload types

Our strategy here is to test more of the emerging features that virtualization provides. Currently we have embarked on just one of the common scenarios for virtualization but have not fully explored that area yet. For example, most of the server consolidation workloads use benchmarks that have a fixed load level. However, in real-world situations, all guests do not run a constant load all the time. The dynamic nature of guests' resource requirements needs to be explored. A hypervisor's ability to accommodate these changes may vary greatly from one solution to another.

Another area to look into could center around the RAS features that virtualization offers. Scenarios that include BIOS and other software updates, requiring the live migration of guests during such operations, could be tested. Other serviceability scenarios could be considered, like impact of guest provisioning on hosts that have active guests. Benchmarking scenarios like these may not be traditionally covered, but with a much more dynamic data-center, these situations need to be studied more closely.

In addition, one might want to explore the concept of whole data-center management using virtualization. This builds on previous concepts like server consolidation and availability, but takes it further. For example, a benchmark may evaluate the performance per watt of an entire data-center. A virtualization management solution and the hypervisors in use can significantly impact the performance and power consumption, especially when the load is dynamic, driving actions like migration to and from hosts to meet quality of service guarantees while conserving the most energy possible.

Another area that probably needs attention is the desktop virtualization scenario. This solution is quickly becoming very competitive, and drawing any conclusions from a server consolidation benchmark may not be prudent. Desktop virtualization has significantly different characteristics than a server consolidation scenario. Desktop users' perceived performance, rather than throughput, may be far more important to this solution.

5 Conclusions

This paper surveys various virtualization benchmarks, comparing their strengths and weaknesses. The art of virtualization benchmarks is in its infancy, however, we believe it is making progress. We are still seeing growing pains in most implementations, as we try to go beyond what traditional benchmarking scenarios have done. Issues such as overall complexity, heterogeneous workload control, quality of service guarantees, and verification all need improvement. These benchmarks have also just begun to simulate the vast number of present and future usage cases that virtualization introduces. We are confident, as long as there is a need for improving virtualization, there will be a drive to improve these benchmarks.

6 Trademarks and Disclaimer

References

[1] IBM Corporation, Virtualization, `http://www-03.ibm.com/servers/eserver/zseries/virtualization/features.html`

[2] VMware Corporation, Build the Foundation of a Responsive Data Center, `http://www.vmware.com/products/vi/esx/`

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield, *Xen and the Art of Virtualization* SOSP'03, October 19–22, 2003, Bolton Landing, New York, USA.

[4] Ian Pratt, Keir Fraser, Steven Hand, Christain Limpach, Andrew Warfield, *Xen 3.0 and the Art of Virtualization,* Ottawa Linux Symposium 2005

[5] Microsoft Corporation, Virtualization and Consolidation, `http://www.microsoft.com/windowsserver2008/en/us/virtualization-consolidation.aspx`

[6] Jeffrey P. Casazza, Michael Greenfield, Kan Shi, *Redefining Server Performance Characterization for Virtualization Benchmarking,* `http://http://www.intel.com/technology/itj/2006/v10i3/7-benchmarking/6-vconsolidate.htm`

[7] Ziff Davis Media, PC Magazine benchmarks, `http://www.lionbridge.com/lionbridge/en-US/services/outsourced-testing/zdm-eula.htm`

[8] Microsoft Exchange Server 2003 Load Simulator, `http://www.microsoft.com/downloads/details.aspx?FamilyId=92EB2EDC-3433-47CA-A5F8-0483C7DDEA85&displaylang=en`

[9] Alexey Kopytov, SysBench: a system performance benchmark, `http://sysbench.sourceforge.net`

[10] Standard Performance Evaluation Corporation (SPEC), SPECjbb2005, `http://www.spec.org/jbb2005`

[11] CMS Eventcorder, `http://www.eventcorder.com`

[12] Vikram Makhija, Bruce Herndon, Paula Smith, Lisa Roderick, Eric Zamost, Jennifer Anderson, *VMmark: A Scalable Benchmark for Virtualized Systems,* `http://www.vmware.com/pdf/vmmark_intro.pdf`

[13] Standard Performance Evaluation Corporation (SPEC), SPECweb2005, `http://www.spec.org/web2005/`

[14] Oracle Corporation, SwingBench, `http://www.dominicgiles.com/swingbench.html`

[15] Andrew Tridgell, dbench benchmark, `http://samba.org/ftp/tridge/dbench/`

[16] VMware, Measure Virtualization Performance with Industry's First Benchmark, `http://www.vmware.com/products/vmmark/`

[17] Yong Cai, Andrew Theurer, Mark Peloquin *Server Consolidation Using Advanced POWER Virtualizatin and Linux,* `http://www-03.ibm.com/systems/p/software/whitepapers/scon_apv_linux.html`

[18] Andrew Theurer, Karl Rister, Orran Krieger, Ryan Harper, Steve Dobbelstein, *Virtual Scalability: Charting the Performance of Linux in a Virtual World*, Ottawa Linux Symposium 2006