2010 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises

CloudGauge: A Dynamic Cloud and Virtualization Benchmarking Suite

Mohamed A. El-Refaey
Arab Academy for Science, Technology and Maritime Transport
College of Computing & Information Technology
Cairo - Egypt
melrefaey@acm.org

Prof. Mohamed Abu Rizkaa
Arab Academy for Science, Technology and Maritime Transport
College of Computing & Information Technology
Cairo - Egypt
m_rizka@aast.edu

***Abstract*— Cloud Computing and virtualization technology are taking the momentum nowadays in data centres, research institutions and IT infrastructure models. Having a reference and performance benchmarks for the dynamic workloads in virtualized and cloud environments is a vital thing to develop.**

Virtualized workload benchmark will help data center's architects and administrators to design an elegant data center's architectural models and establish a workload balance between virtualized and consolidated servers, according to the workload's analysis and characterization produced by this reference benchmark.

In this paper, we will present an overview about CloudGauge, a dynamic and experimental benchmark for virtualized and cloud environments, and the key requirements and characteristics of virtual systems' performance metrics, evaluation and workload's characterization which can be considered a step further to implement a production-ready virtual systems and cloud benchmarking suite and performance models.

Keywords— *CloudGauge; virtualization; Workload characterization; Benchmark; Computer Performance; Cloud Computing; hypervisor; vConsolidate; VMmark; SPEC; Xen, KVM, OpenVZ.*

I. INTRODUCTION

Performance evaluation and benchmarking in commodity hardware, operating systems and their normal applications workloads can be considered an achievable and easy task to implement and there are lots of already implemented benchmarking suites for these applications, database workload, CPU and web servers' workloads,... etc.

However, the same benchmark and performance evaluation process can't be directly applied to dynamic workloads in virtualized systems in which the virtualization layer and the dynamicity in consolidating these workloads impact the performance's measurements and evaluation process.

In this paper we shall shed some lights on the key characteristics of virtual machine benchmark and the design of the proposed benchmark, CloudGauge.

CloudGauge, the core of this study is an ongoing project, still in development, aims to implement an intelligent configurable and customizable benchmarking suite that will help in evaluating workload performance and characterization in the cloud computing context and consolidated servers for different setup scenarios and different workloads and also helps in achieving an intelligent workload balancing in clustered, virtualized and cloud environments.

I. BACKGROUND

Cloud computing and virtualization technology are taking momentum nowadays in research and industry, and they constitute a new strategic approach and delivery model to deploy large-scale software services that enables IT as a service while reducing virtual sprawl in data center. Instead of running on dedicated hardware, virtualization and cloud technologies enable the customers and user to share the key four computing resources (processing power, memory, and storage and network bandwidth) of large data canters according to their needs and the workload varies frequently over time according to their usage.

Advancing in both technologies to put them in production use requires an accurate background about the various workloads behaviour injected or consolidated in virtualized environment and accordingly in clustered cloud setups. Workloads' characteristics and performance information are very important in deciding where and when to distribute the loads between cloud nodes and make the user experience more comfort with the application in use.

Critical systems effectively incorporate the knowledge obtained through the analysis of workloads. Such knowledge can be obtained either through performance monitoring of running workloads or workload characteristics under different testing scenario. As it is very invaluable to understand the workload and properly servicing business demand, the two important questions arising are how to do so and what happens once you understand that demand?

Workloads vary according to their types; like intensive numerical and scientific processing, event-driven, request/response, high throughput, etc. Understanding the workload requirements and specifications requires the knowledge and the establishment of the metrics that will be used to measure each workload's quality of service.

Designing and implementing a real reference or benchmark for the cloud dynamic workloads is invaluable and a must to accommodate new business models and satisfy the

978-0-7695-4063-4/10 $26.00 © 2010 IEEE
DOI 10.1109/WETICE.2010.17

enterprises' requirements and their on-demand computing services' needs, it will also help data center's architects and administrators to design an elegant data center's architectural models.

A successful virtual machine benchmark must exhibit the following characteristics: repeatable, representative, easy to run, verifiable, precision and complete coverage.

A comprehensive insight of performance's information across physical and virtual machines enables IT organizations to analyse the usage and utilization's trends and make future decisions about their virtual infrastructure's capacity and how to optimize it and improve ROI.

Designing a benchmark with above characteristics is not an easy task; indeed each of the above characteristics presents its own challenges. In this research, we shall investigate techniques to address the challenges to cloud benchmarking's implementation.

Some benchmarking's efforts have been done in this area from the industry and research, like VMware VMmark, Intel vConsolidate; these projects address high-level performance's characterization of co-located virtual machines.

A. *Performance Metrics and Workload Characterization*

In the field of computer science, there is a little agreement on how to measure something as fundamental as the performance of a computer system. For example, the speed of an automobile can be readily measured in some standard units such as meters travelled per second [3].

The use of these standard units then allows the direct comparison of the speed of the automobile with that of an airplane, for instance. Comparing the performance of different computer systems has proven to be not so straightforward, however [3]. The same rule applies for measuring performance of virtualized systems and workloads in the cloud.

The dictionary defines workload as "the amount of work assigned to, or done by a worker or unit of workers in a given time period" (The American Heritage Dictionary, 2nd Edition).

Within the confines of a network, workload is the amount of work assigned to, or done by a client, workgroup, server, or internetwork in a given time period. Therefore, *workload characterization is the science that observes, identifies and explains the phenomena of work in a manner that simplifies your understanding of how the network is being used*.

In order to test multiple alternatives under identical conditions, the workload should be repeatable. Since a real-user's environment is not generally repeatable, it is necessary to study the real-user's environments, observe the key characteristics, and develop a workload model that can be used repeatedly. This process is called workload's characterization. Once a workload model is available, the effect of changes in the workload and system can be studied in a controlled manner by simply changing the parameters of the model[1].

Requirements of Virtual Machine Benchmarking

A virtual machine benchmark should meet the following requirements [5]:

- The ability to capture the virtualized environment key performance characteristics.
- The ability to define an easily understandable metric that scales with underlying system's capacity.
- With these scalable metrics, the same benchmark can be used for different servers' sizes.
- The measurements can be generated using a controlled policy based on a combination of increased individual workload's scores and running an increasing number of workloads.
- The ability to easily compare different hypervisors and virtual machine environments' setups (i.e. the benchmark should run on any hypervisor without major changes and adaptations).
- Ability to predict the performance of different types of applications.

Benchmarks are categorized according to the type of workload that they impose on the system under test, or based on the type of proper metrics that they use to characterize the system. The following is a list of the most commonly used metrics [19] [20]:

- Latency, the interval between stimulation and response.
- Throughput, the amount of data or events per time.
- Dilation, the extension in length.
- Utilization, the percentage of utilized computing time on a processor.

Another set of metrics contributed by Eeckhout et al.[21] For Multiprogram performance's metrics:

- ANTT (Average Normalized Turnaround Time), a user-oriented performance's metric.
- STP(System throughput), a system-oriented performance's metric

A metric represents only certain aspects of performance, some of which might be more important than others for a given combination of workload's and user's requirements.

B. *Related Work*

In the context of virtual performance's evaluation and workload's characterization, there should be an overview about the benchmarks already existing in the field;

A number of research and industry's studies have been presented to study the impact of virtual machine runtime overhead imposed by hypervisors layer on a variety of workloads [6][7]. It is very important for benchmarking's applications in production to know the amount of overhead resulting from the virtualization layer. The study of the impact of consolidated several application on a single physical server running Xen [15]hypervisor is introduced in [8]; Padma Apparao et al. present a study about the Implications of Cache Asymmetry on Server Consolidation Performance to evaluate the different form of cash asymmetry and its implication on workload performance[9]. Pradeep et al. [11] evaluated how OpenVZ [16] and Xen [15] perform when used for consolidating multi-tiered applications, and how performance

is affected by the different configurations for consolidation in a deep analysis of results using OProfile [12].

One of the benchmarks proposed by Intel is vConsolidate (vCon) [5], which consists of four key virtual machines: a compute intensive workload, a database workload, a Web server workload, a mail server workload and an idle VM added to the mix to emulate the real world scenarios on which servers are not fully utilized all the time.

VMmark [13] is a tile-based benchmark consists of several common workloads running simultaneously in each virtual machine. Tile is defined as a collection of different workloads aggregated into a unit of work and the workloads are mail, Java, standby, web, database, and file server. The multiple tiles run simultaneously and a score, which may be higher than the overall of each benchmark's score, would be generated when tiles run over. The final goal of VMmark is to create an effective measurement for virtual machine performance's evaluation across different hardware's platforms.

VSCBenchmark [10] is another benchmark that mainly evaluates the dynamic performance of server's consolidation such as creating and killing the VMs under different workload along with some experimental results and characterization of the dynamic performance of Xen [15]and OpenVZ [16].

II. SPEC (Standard Performance Evaluation Corporation) [2] committee is developing a standard vSPEC benchmark for virtualization. The SPEC Virtualization Committee is developing a new industry standard benchmark for evaluating virtualization's performance for data center's servers. The committee plans to make use of the comprehensive system-level application-based benchmarks that SPEC offers. The workloads represented by these benchmarks are also representative of commonly virtualized server's applications. SPEC's expertise in system level's benchmarks that support a wide range of hardware architectures and operating systems will greatly benefit from the committee's efforts to develop a standard methodology to evaluate the performance of servers using virtualization for server's consolidation [14]

III. CLOUD CLOUDGAUGE BENCHMARK

A. CloudGauge Design and Architecture

The CloudGauge architecture is depicted in Figure 1;

CloudGauge virtual machine benchmark is an experimental framework that accommodates the aforementioned requirements needed by a VM benchmarking. It has been taken into consideration while designing this benchmark the virtualization layer's impact on performance and the virtual machine's interference. This framework is an integrated system for dynamically injecting workloads, configuring and customizing the virtualization layer, measuring the performance and communicating measurements and statistics' reports throughout its dashboard. And finally this can be a starting point to an intelligent workload's balancing components that allocate, de-allocate and move virtual machines between virtualized cluster to make their services highly available and help in maintaining physical servers that host the virtual machines.

As shown in Fig 1, the benchmarking suite can be divided into three layers:

- System Under Test layer.
- Benchmarking Suite engine layer.
- Test Control layer

A detailed description and explanation for each layer and how they interact with each other in the full testing lifecycle is covered in the next section;

1) **S**ystem **U**nder **T**est (SUT):

Mainly this layer represents the physical servers along with their hosted virtual machines' candidates for workload's characterization and performance's measurements.

This layer consists of 2-3 virtual machines (number of VMs is chosen according to the test setup needs), Xen,

KVM or openVZ-based, Or even Amazon AMI-like virtual machine in a private cloud setup environment (e.g. images deployed using Ubuntu Enterprise Cloud, EUC[30]).

Each virtual machines is injected with virtual machine Tentacles components (sensors, hooks and scripts to measure and collect performance readings) to feed the engine layer with performance statistics and measurements; interesting to mention that 'Tentacles' are inspired from elongated flexible organs that are present in some animals, especially invertebrates, and sometimes in the hairs of the leaves of some insectivorous plants. Usually, they are used for feeding, feeling and grasping.

Also a Libvirtd[25] daemon is installed on each virtual machine in order to be able to manage the virtual machine from within the testing engine layer.

Virtual machine can be injected with any desired workload (e.g. SQL-Bench[32], Unixbench[31], Automated kernel compilation workload, CPU compute intensive script etc.) to measure certain aspects of the system and virtual machine under test.

2) Benchmarking suite engine:

The middle layer in the architecture above, this layer is mainly the core of the CloudGauge benchmarking suite and responsible for bootstrapping the machines under test, test provisioning and workload injection, collecting and generating test reports, read the hypervisor states from the performance agent component, read the workload measurements data from the traces database and finally generates artifacts to the workload manager which can benefits from these artifacts to balance the virtualized environment workloads.

This layer consists of the following main modules:

- Test Provisioning.
- Workload Management.
- Unified Hypervisor Interface (UHI).
- Virtual Machine Image Repository.

Test Provisioning: is the section responsible for defining test hooks (onTestStart, onTestRun etc.), defining the test plug-ins (for user to be able to define his own test plug-ins and to pick up the needed workload).

It is also responsible for bootstrapping the machine with required images (by interacting with the UHI design-time module to create the required images) and initiating the testing experiments and communicating with the workload management's module to select and inject the workload required into a selected virtual machine.

Test provisioning interacts with SUT layer through tracing db (the database that stores the measurements and traces events occurred at each virtual machine) and interacting with Test Control layer through web interface to report the results in a predefined reporting format.

Workload Management is the module responsible for picking up the workload required from a workloads definitions' file and then generating and injecting that workload into the virtual machine under test.

This module can also be further used in intelligently balance the workloads between virtual machines in the cluster or the system under test using automated workload balancing methodology based on the performance readings that CloudGauge provides for the system dashboard (This module is out of our study scope but it has been put into the architecture to stress the importance of the benchmark in the workload decision making process to make the load easily balanced throughout the virtual environment).

This module enables the user to pick up any workloads' needs to be tested from the workloads defined in the CloudGauge load definitions repository.

Unified Hypervisor Interface (UHI), this module demonstrates the Cloud Gauge's ability to be a hypervisor-agnostic benchmarking suite by which the benchmarking can be configured to run Xen[15], OpenVZ, KVM or any hypervisor existing.

In this module the libvirt (an open source management toolkit to interact with the virtualization capabilities of recent versions of Linux) library is being used; to make it easy to manage the virtualized environment under test; it also supports many hypervisors that are existing (e.g. Xen[15], KVM[17], OpenVZ[16], User Mode Linux[33], VirtualBox[34], VMware ESX and GSX)

In UHI module there are two modules, Design-time and Run-time modules.

Design-time component is a set of tools provided by libvirt to help in image creation, cloning,... etc through its utilities 'virt-image' and 'virt-clone'.

These components are being called by the test provisioning module to create the image required of the operating system and hypervisor.

Run-time component is a set of tools to provision and install (Python virt-inst, a command line tool for provisioning new virtual machines using the "libvirt" hypervisor management library) the virtual machine and mange it through virsh utility, which interface with

virtual domains, i.e. create, pause, and shutdown domains.

Virtual Machine Image Repository, in order to facilitate deployment of virtual machines, we created a repository of predefined virtual machine images to pick up one of them to provision and deploy it. The image provides metadata describing the requirements of the operating system, minimal resource allocations, and pre-installed disk.

3) Test Control layer: this layer is a web-based interface to manage the testing lifecycle along with the oVirt web tool and monitor the virtual machines deployed in the virtual environment. It enables testing configurations needed and acts as a reporting and dashboard for the testing cycle and its reports.

B. Anatomy & Design Philosophy

CloudGauge design convey some ideas that make it easy to measure the characteristics and performance of the workload and get the most benefits out of the results by allowing an efficient load balanced and highly available services across the clustered environment.

So, CloudGauge is not just a way for testing and measuring performance and workload characterization of virtual machines. It is an integrated framework that unifies the way workload is being injected, configured and measured. The same applies in the case of hypervisor and virtualization layer required for testing, as CloudGauge makes it easy to select the image of the needed operating system its appliances and its hypervisor from a pool of virtual machine images to setup the test scenarios.

It provides the reporting facility to give an accurate indicative results by reading and inspecting real-time performance data from the Tentacle components and the VMM states of the performance agent component and display the measurements and performance readings to end observer in a web-based dashboard and it can also be used to feed another workload manager for subsequent decision-making processes for load balancing and most probably live migration of virtual machine in case of load starvation or in case of going down for maintenance, upgrade,... etc.

It won't be Xen [15], KVM [17], or OpenVZ [16] only hypervisors that will be used throughout the benchmark, but it will be extensible to accommodate VMware ESX [18] and other hypervisors.

The actual workloads that the benchmark suite runs will be configurable as well as there are many applications candidates to be virtualized, so the flexibility to inject such applications into the suite is a key value of our benchmark that make it more suitable for general purpose use instead of being limited to some fixed set of workloads.

CloudGauge is intended to report system level performance metrics (e.g. performance counter events), VMM states, and high –level performance metrics (e.g. system calls, page faults, execution time).

C. Implementation

Implementation details are omitted for the sake of space, but what worth mentioning is that the CloudGauge is a composition of open source libraries and tools like libvirt, oVirt[26], Faban[28], sar[27] and Python language which is used to manage all the chosen components and to facilitate accessing a lower level of system calls when needed.
This project will be soon an open source-project to the community to contribute in its development.

D. Tools & Technologies

We used the following tools and technologies in implementing CloudGauge testing scenarios:

- Libvirt[25]
- oVirt[26]
- Sar[27]
- Faban[28]
- KVM[17]
- Collectd[29]

E. Workload Characterization and Testing workflow

In order to implement and establish a benchmarking with particular configurations, we need to follow the following testing workflow:

1- Prepare the needed test setup configurations, the workload needed, metrics to be measured and number of machines to be provisioned.
2- The lifecycle starts with the testing provisioning and according to the testing configuration it instructs the UHI layer to prepare, install and provision the virtual machine type required.
3- Then it picks the workload required from the load definitions files if it is one of the predefined workloads in the benchmarking suite; otherwise the tester should define his workload script or program to be used.
4- Once the virtual machine under test is ready for benchmarking; the UHI will start the machine through its libvirt interface and call back the testing provisioning indicating that the machine is ready to execute the workload.
5- Testing provisioning starts to inject the workload into the virtual machine either through SSHing access to the machine and execute the required workload (e.g. kernel compilation automated script) or through the deployment of the load into the machine and executing it remotely.
6- Test hook (onStart) is called and performance sensors and statistics collecting daemons (sar, virt-top, xentop, Collectd etc) start to read and collect the

virtual machine state while the workload is being executed.

7- Once the workload script or program finishes the onEnd hook is being called and the collected data is sent to the test reporting to be viewed on the web dashboard.

F. Experiment

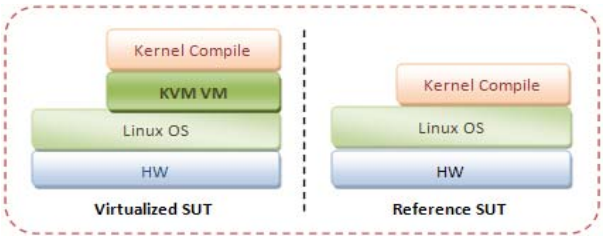

Fig. 2 Experiment Setup

To demonstrate the effectiveness of CloudGauge on a full-fledged virtualized system, we built a test-bench based on KVM. The guest was running primarily Ubuntu Linux.

In one early experiment, we used CloudGauge testing suite and workflow system to deploy several virtual Linux instances under the KVM hypervisor. One of them ran a kernel compilation workload and the workload was timed to begin so as to vary the overall load on the host system. We collected metrics from within the systems using sar (1), Collectd and virt-top.

The main objective of this experiment is to use CloudGauge to perform the experiment which compare the performance of a reference setup (base Linux operating system without any virtualization layer), with a virtualized solution running on the same hardware. The same workloads were running on top of the two architectures.

The difference in system performance would only be a consequence of the virtualization layer (VMM).

The virtualization layer participated in the benchmark is KVM.

The tests used different benchmark tools to stress the base Linux platform and the virtualization solutions, and were conducted many times for the accuracy of measurements.

Each experiment consisted of the following steps:

1. Starting up a KVM guest virtual machine.
2. Starting virt-top, configured to log measurements of the domain.
3. Sar and Collectd are also configured to collect performance data accumulatively.
4. Start/inject the workload in question
5. Collecting data for about tow minutes delay between each measurement.

Native Setup

In this setup a core 2 duo with 4 GB machine was used to benchmark the performance of the system under a heavy CPU-intensive workload; kernel (version 2.6.31) compilation is used. Server is hosted with a Linux Ubuntu

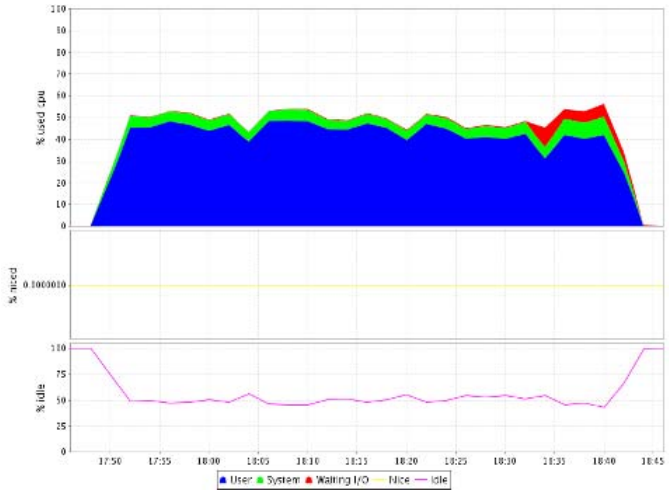

Fig. 3 CPU Usage on Base Linux

9.10, Karmic Koala 64-bit PC (AMD64) server installation.

Kernel compilation under the above setup took around 52 minutes. And the performance of the system under this heavy workload is depicted in fig 3, 4, 5 and 6:

In which the percentage of CPU usage was about 50% of the compilation time as shown in fig 3.

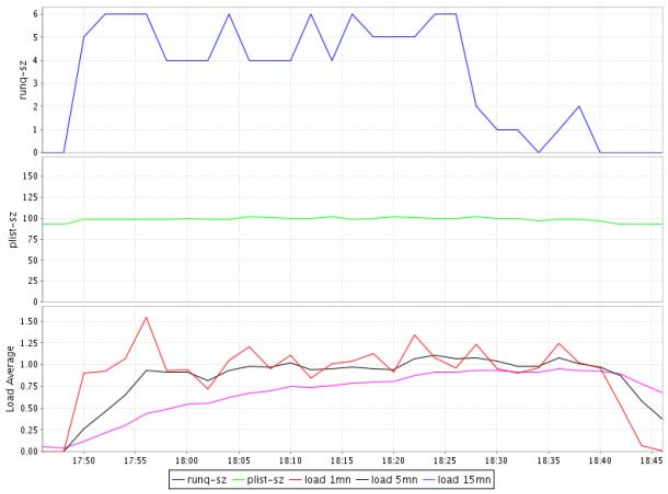

Fig. 4 Average Load

The average load is changing between 1.25 and 1.5 and memory usage was about 1GB at the first half of the kernel compilation and around 3.5 GB at the second half of the kernel compilation (85% of the total memory of the system under test) as shown in figure 5 and 6.

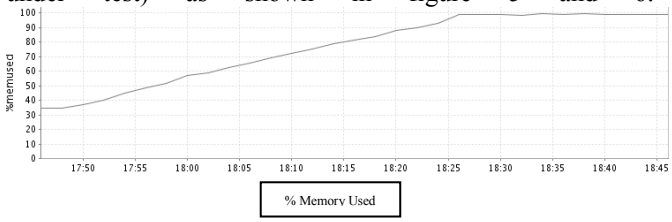

Fig. 5 Memory Usage

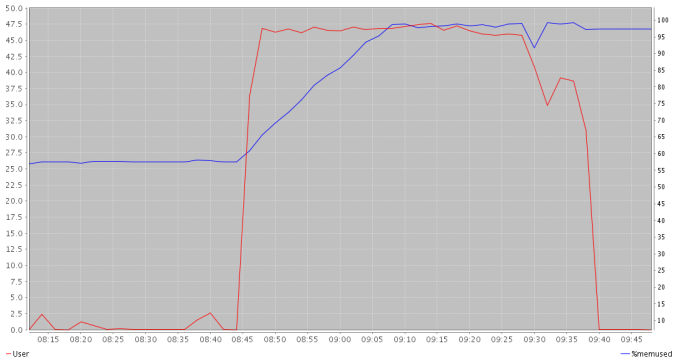

Fig. 6 Memory and CPU Usage

Virtualized Setup

In the virtualized setup, the same operating system is used and the same server configurations.

KVM hypervisor is used; virtual machine was provisioned with the following:

- Ubuntu AMD64 operating system.
- 1 VCPU.
- 1 GB of memory

A kernel compilation workload is used throughout the experiment and took around 72 minutes (about 20 minutes over the time of the base Linux compilation).

Libvirtd was used to communicate with the provisioned virtual machine.

System performance measurements were collected for both the host and the virtual machine to see how much the virtual machine has exploited of the host machine resources.

As shown in fig 7, the CPU usage was about 90% throughout the experiment (72 minutes).

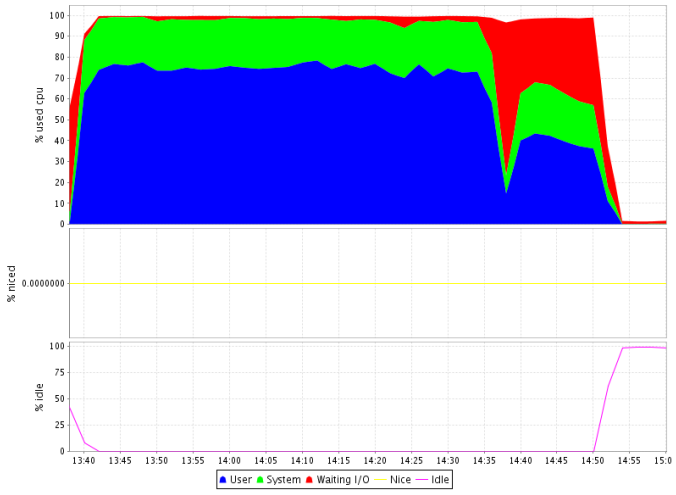

Fig. 7 CPU usage of the Virtual Machine

The memory usage was about 954MB as depicted in fig. 8 and the average load was about 1.75 as shown in fig 9.

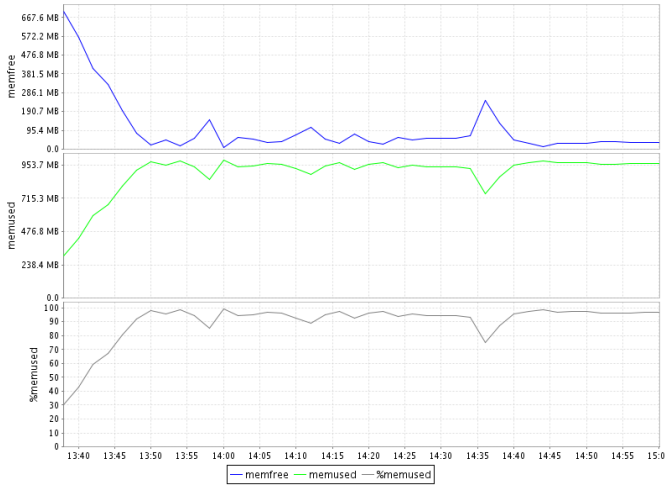

Fig. 8 Memory Usage

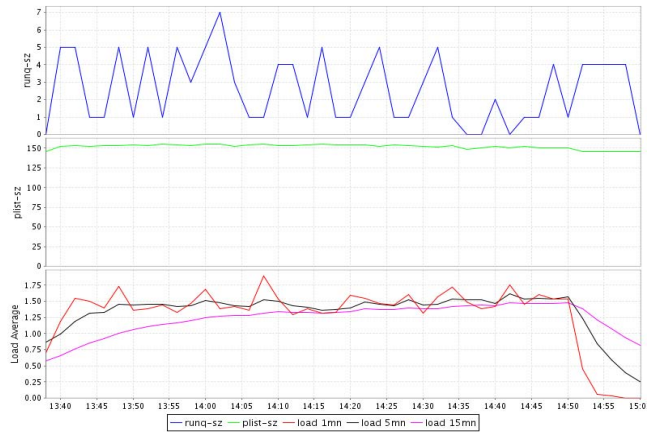

Fig. 9 Virtual Machine Average Load

Host Machine with running Virtual Machine

In this section, we shall cover the performance measurements of the host virtual machine while the workload is being exerted on the guest virtual machine.

The reason behind this is to know the measurements of the performance of the virtual machine internally from within the virtual machine itself and externally from the host machine to know what the overall performance of the virtualized setup and the percentage of the guest to host's usage.

The memory of the host machine was about 3.7 GB all over the test of the kernel compilation (differs from the native Linux setup as in the native one it is about 3.7GB at the second half of the experiment) as shown in fig 12.

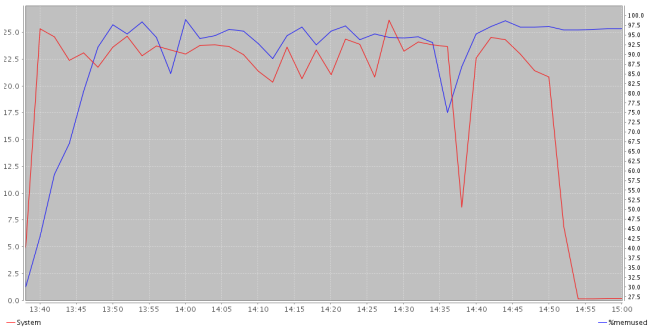

Fig. 10 Memory and CPU usage

The CPU usage of the host machine is depicted in fig. 11; and the graph illustrates how the CPU was about only 7.5 % of its all capacity.

Fig. 13 illustrates the percentage of the guest CPU usage and the host CPU usage performance.

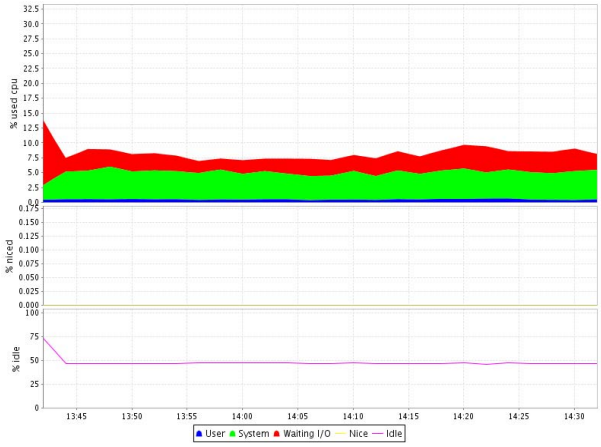

Fig. 11 CPU Usage for the Host

Fig. 12 Memory Usage for the host machine

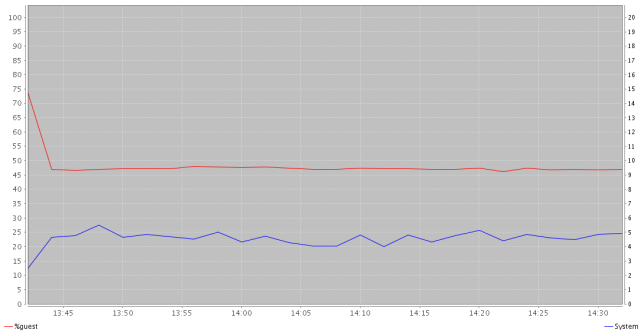

Fig. 13 CPU Guest Usage and the Host CPU

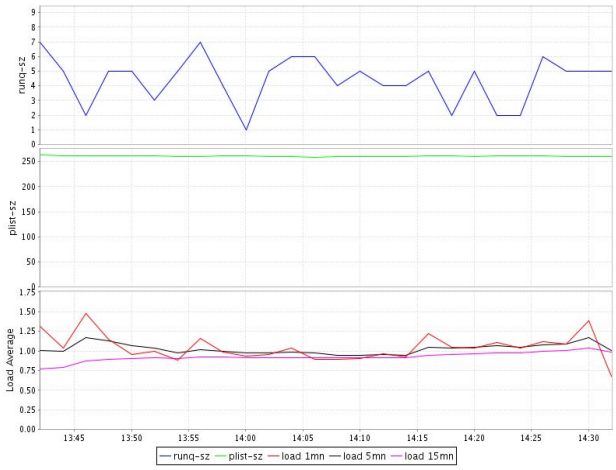

Fig. 14 Average Load of the Host machine

Results & Observations

- CloudGauge provided us with automated scripts to provision and measure the required virtual environment setup in a small amount of time.
- Kernel Compilation on a native Ubuntu Linux box took around 52 minutes.
- Kernel Compilation on a KVM-based virtual machine with Ubuntu Linux guest took around 72 minutes with 1 GB RAM and 1 VCPU.
- The impact of the virtualization layer was less than 10-12 percent.
- From the results, we can investigate how to customize the virtualization layer depending on the applications needs.
- The systems under test, native and virtualized one didn't reach its limit while the workload was running.
- We collected performance counters for CPU usage, Average load and memory usage for systems under test.

II. Conclusions & Future work

In this paper we presented an overview about the importance of measuring performance and characterizing the workload in virtual and cloud environments. We also presented our vision about an integrated, hypervisor-agnostic benchmarking suit represented by CloudGauge system and its architecture. And used it to perform performance analysis and workload characterization on KVM-based virtual machines and comparing it with another system with basic Linux installations using Linux kernel compilation as a workload.

Future work will focus on intensively measuring other hypervisors and cloud based systems using CloudGauge and apply it on a large scale test beds (opencirrus [22], PlanetLab [23] or Emulab [24]) or in production environments and also try to integrate Faban [4]project with CloudGauge as a workload generator.

References

[1] R. K. Jain. Art of Computer Systems Performance Analysis Techniques for Experimental Design Measurements Simulation and Modeling, John Wiley & Sons (1991).

[2] Standard Performance Evaluation Corporation-http://www.spec.org

[3] David J. Lilja. Measuring Computer Performance a practioner's Guide, Cambridge University Press 2004.

[4] Sun Microsystems: Project Faban. http://faban.sunsource.net

[5] Casazza, J.P., Greenfield, M., Shi, K.: Redefining server performance characterization for virtualization benchmarking. Intel Technology Journal 10(3) (2006)243–251

[6] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer†, Ian Pratt, Andrew Warfield: Xen and the art of virtualization. In: SOSP '03: Proceedings of the 19th ACM Symposium on Operating Systems Principles, New York, NY,USA, ACM (2003) 164–177

[7] Muli Ben-Yehuda, Amit Shah, Balaji Rao, Todd Deshane, Zachary Shepherd,Jeanna N. Matthews: Quantitative comparison of Xen and KVM. In: Xen Summit 2008.

[8] Padma Apparao, Ravi Iyer, Xiaomin Zhang, Don Newell, Tom Adelmeyer:Characterization & Analysis of a Server Consolidation Benchmark. In: VEE '08: Proceedings of the fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, New York, NY, USA, ACM (2008) 21–30

[9] Padma Apparao, Ravi Iyer and Don Newell: Implications of Cache Asymmetry on Server Consolidation Performance. In: IISWC-2008: Proceedings of the IEEE International Symposium on Workload Characterization, Seattle, WA, USA(2008).

[10] Hai Jin, Wenzhi Cao, Pingpeng Yuan, Xia Xie.:VSCBenchmark: benchmark for dynamic server performance of virtualization technology . In: Proceedings of the 1st international forum on Next-generation multicore/manycore technologies (2008), Cairo, Egypt.

[11] P. Padala, X. Zhu, Z. Wang, S. Singhal, and K. G. Shin, "Performance Evaluation of Virtualization Technologies for Server Consolidation", HP Labs Technical Report, HPL-2007-59R1, 2008.

[12] Oprofile: A system profiler http://oprofile.sourceforge.net/.

[13] VMmark: A Scalable Benchmark for Virtualized Systems.Technical Report VMware-TR-2006-002 September 25, 2006 http://www.vmware.com/pdf/vmmark_intro.pdf

[14] SPEC Virtualization http://www.spec.org/specvirtualization/

[15] Xen Hypervisor, http://xen.org/

[16] OpenVZ, http://wiki.openvz.org/Main_Page

[17] Kernel Base Virtual Machine, http://www.linux-kvm.org/page/Main_Page

[18] VMware ESX and ESXi, Bare-Metal hypervisor for Virtual Machines, http://www.vmware.com/products/esx/index.html
[19] Yasuhiro Endo, Zheng Wang, J. Bradley Chen, and Margo Seltzer. Using latency to evaluate interactive system performance. ACM SIGOPS Operating Systems Review, 30(SI):185–199, 1996.
[20] Andrew S. Tanenbaum and Maarten van Steen. Distributed Systems: Principles and Paradigms. Prentice Hall, 2002.
[21] Lieven Eeckhout, Stijn Eyerman: System-level Performance Metrics for Multiprogram Workloads, IEEE Micro 2008.
[22] Open Cirrus, https://opencirrus.org/
[23] PlanetLab | An open platform for developing, deploying, and accessing planetary-scale services, http://www.planet-lab.org/
[24] Emulab.Net - Emulab - Network Emulation Testbed Home, http://www.emulab.net/
[25] Libvirt : The virtualization API, http://libvirt.org/
[26] oVirt, http://ovirt.org/
[27] sar: Linux Man page, http://linux.die.net/man/1/sar
[28] Faban, http://faban.sunsource.net/
[29] Collectd – The system statistics collection daemon, http://collectd.org/
[30] Ubuntu Enterprise Cloud, http://www.ubuntu.com/cloud/private
[31] byte-unixbench - Project Hosting on Google Code, http://code.google.com/p/byte-unixbench/
[32] mysql-benchmarks,http://dev.mysql.com/doc/refman/5.0/en/mysql-benchmarks.html
[33] The User-mode Linux Kernel Home Page, http://user-mode-linux.sourceforge.net/
[34] VirtualBox, http://www.virtualbox.org/