CLOUD BENCHMARKING IN BARE-METAL, VIRTUALIZED, AND CONTAINERIZED EXECUTION ENVIRONMENTS

Soheil Mazaheri, Yong Chen, Elham Hojati, Alan Sill

Texas Tech University, Lubbock, Texas, 79409, USA
soheil.mazaheri@ttu.edu, yong.chen@ttu.edu, elham.hojati@ttu.edu, alan.sill@ttu.edu

Abstract: Cloud data centers rely on virtualization to increase their productivity and reduce complexity for end users while delivering access on-demand as a service. Because virtualization involves increased abstraction, there is an inevitable trade-off between virtualization and potential performance degradation. As a result, there has been a considerable amount of work in developing and improving different virtualized environments that can isolate a workload from another. Virtual machines and Linux containers are the most well-known techniques that provide an isolated environment to allow applications to run independently. Each of these methods has its own advantages and disadvantages. This research aims to compare the performance of systems in bare-metal, containerized (Linux containers), and virtualized (virtual machines) execution environments. It also analyzes and understands the implications of the benchmarking results in these different execution environments. The benchmarking results, analyses, and insights discussed can provide a guidance to cloud computing researchers and developers in the process of designing cloud computing solutions, deploying cloud systems, and developing algorithms, programs, and applications on different cloud platforms.

Keywords: Benchmarking; virtualization methods; virtual machines; Linux container; bare-metal; cloud computing; Infrastructure as a Service

1 Introduction

Cloud computing is a solution based on consumption of computing resources as services. It involves deploying remote servers and software stack that allow on-demand online access to computing and storage resources. At the foundation of cloud computing is the broader concept of shared services. Cloud computing uses virtualization to encapsulate and distribute instances of services and resources. Although cloud and virtualization are two close concepts, they are not the same. While these technologies might share a common bond of maximizing computing resources, there are differences between them. Virtualization can reduce complexity for end users while allowing resources to be utilized more effectively. Cloud computing takes the use of those resources to the next level by delivering on-demand access to those components as a service, further reducing complexity, cost, and burden.

The physical host or the actual machine is usually called bare-metal. In this research, by bare-metal, we mean the real host on which an operating system (OS) and other software layers are installed. Virtual machines (VMs) imitate certain factual or conceptual hardware. A virtual machine is a software implementation or an emulation of a machine that executes programs in the same manner as a physical machine. This approach can be implemented using a type 1 hypervisor, which runs directly on the hardware, or a type 2 hypervisor, which runs on another operating system.

Operating system level virtualization allows resources of a system to be partitioned via kernel's support for multiple isolated user space instances, which are usually called containers. A feature in the kernel of Linux known as control groups provides the ability to organize processes into virtualized containerized environments called LinuX Containers (LXCs). Based on this feature, open-source projects such as Docker [1] and Rocket have emerged. These projects expand the basic functionality of LXCs by adding API controls and features to manage containers in groups. Containers are sometimes described as extremely lightweight virtual machines. More accurately, however, it is merely a way to group processes and manage them together in isolated environments that share underlying features of the OS, such as the kernel, but are otherwise separated from other containers. This approach allows the containers to share the resources of the host machine, without the overhead of a hypervisor. In addition, projects like CoreOS have emerged to provide operating system environments that are streamlined for this purpose and include tools to manage and orchestrate these containers. As an example, CoreOS is a distribution of Linux which provides a minimal operating system and includes "etcd" as a System Discovery method, and in which everything runs in a container. As a result of this minimal operating system environment, there are more hardware resources, including memory, to run applications inside of containers.

In this research, we study and evaluate the performance of these environments (the target system (Bare-Metal), Linux containers (Docker), and virtual machines (KVM)) to have a clear point of view regarding the comparison between the characteristics of each environment with one another. In detail, using a variety of benchmark tools, we investigate, measure, and analyze the performance of a

978-1-5090-1256-5/16/$31.00 ©2016 IEEE

system in bare-metal versus Linux containers versus virtual machines. These results, analyses, and findings can assist cloud computing researchers and developers in designing and developing more effective cloud computing solutions, systems, algorithms, and applications.

Our work makes the following contributions:

- Provides an up-to-date comparison of native, container, and virtual machine environments using latest hardware and software across a variety of interesting benchmarks and workloads that are relevant to the cloud.
- Provides analysis and understanding on Cloud Tester Benchmark Suite (CTBS).
- Identifies the impact of current virtualization options for cloud.

2 Cloud benchmarking methodology

This research primarily focuses on benchmarking and comparing three different execution environments, including bare-metal, containerized (Docker), and virtualized (KVM) execution environments. We introduce the benchmarking methodology in this section.

2.1 Background

Benchmark often refers to the concept of a System Under Test (SUT). The SUT is a collection of components necessary to run a benchmark scenario. A cloud benchmark is a benchmark in which the SUT contains a cloud service as component of interest. Cloud benchmarking can be done in each layer of cloud (IaaS, PaaS, and SaaS). In this research, we have focused on benchmarking at the IaaS layer of cloud. There are numerous well-known benchmarking suites, which are useful for benchmarking at this layer. These benchmarks include, but not limited to, IOR (Interleave Or Random), Bonnie++, IOZone, SHOC (Scalable Heterogeneous Computing), VMmark, HPCC (High Performance Computing Challenge), LINPACK, HPL, and CTBS (Cloud Tester Benchmark Suite) [2]. We have used the CTBS, and primarily IOR and HPCC in it, for this research. We briefly explain it below.

CTBS (Cloud Tester Benchmark Suite) is a benchmark suite for standardized computing performance benchmarking of cloud Infrastructure-as-a-Service (IaaS) environments. It adopts a bottom-to-top approach, which means that it evaluates the performance of physical hardware then the impact of virtualization of compute, network, storage performance, and finally the performance of the management software [2]. It determines basic computer system (i.e. compute, interconnect, and storage) performance for individual servers deployed within a cloud IaaS system. Part of the suite can also be used to determine native (non-virtualized) server performance as well. It consists of some well-known benchmarks including HPCC, IOR, SHOC, and LAPACK. Computing and interconnect performance are determined using the HPCC software suite; storage performance is determined with the IOR software suite; and heterogeneous processing performance is determined with the SHOC software suite. The entire CTBS system is self-contained within a Cloud Tester VM Image (CTI), which can be easily deployed to a cloud IaaS system uses either a KVM or VMware ESX/ESXi hypervisor.

2.2 Benchmarking bare-metal execution environment

This phase of research was performed with using a research cluster hosted at the High Performance Computing Center of Texas Tech University. The cluster was installed from the scratch with CentOS 6.6. To minimize the variance of the result of the benchmarks, the minimum base distribution of CentOS was used. Then, all required packages, which were necessary to compile CTBS were installed on the node. Since the required installed packages are not a service to run in background, they do not have any overhead to the system. We consider the system as the pure minimal installation of CentOS 6.6.

The general specifications of the hardware include: CPU (2 * Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz (24 CPUs)), RAM (64GB (8 * 8GB) @ 1600MHz), storage (250 GB Hard Disk Drive, 8 * 2TB Hard Disk Drive (16 TB)), and network adapters (1GbE BMC, 2 * 1GbE, 2 * 10GbE).

2.3 Benchmarking containerized execution environment

This phase of research was performed with using the minimum base CentOS 6.6 plus Docker v1.4.1. The Docker container was evaluated with no limitation in terms of the resource of the system. As a result, the container had access to the all of the system's resources (CPUs, memory, etc.). The required packages for compiling CTBS were installed inside of the container. Since the required installed packages are not a service to run in background, they do not have any overhead to the container, and the container can use all resources of the system.

2.4 Benchmarking virtualized execution environment

This phase of research was performed with using KVM virtual machine with minimum based CentOS 6.6 installed on the physical host. All the physical resources of the system were assigned to the virtual machine, which means that the KVM virtual machine had access to the all of the system's resources. The required packages of the CTBS were installed inside of the virtual machine. Since the required installed packages are not a service to run in background, they do not have any overhead to the VM, and we consider the VM as the virtualized execution environment with base minimal CentOS 6.6.

3 Cloud benchmarking results and analysis

This research focuses on benchmarking the IaaS layer of cloud under bare-metal, containerized, and virtualized execution environments with the CTBS benchmark suite. We primarily report the results and analysis with the HPCC and IOR benchmarks in the CTBS suite.

3.1 HPCC results and analysis

The HPCC (High Performance Computing Challenge) benchmark consists of 7 tests, HPL, DGEMM, STREAM, PTRANS, FFT, “Communication Bandwidth and Latency”, and “Random Access”. Due to the page limit, we only present a few selected results as shown below. The general observation can be made that the containerized environment (i.e. Docker) achieved comparable performance with the bare-metal solution, while the virtualized environment (i.e. KVM) incurred rather significant, observable overhead.

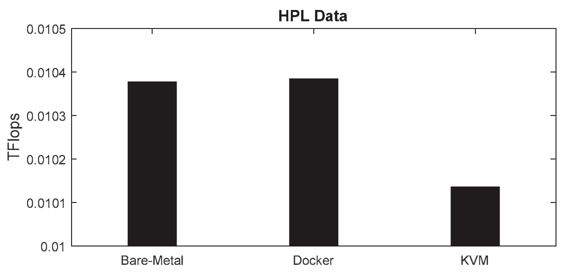

Figure 1 HPL results with bare-metal vs. Docker vs. KVM

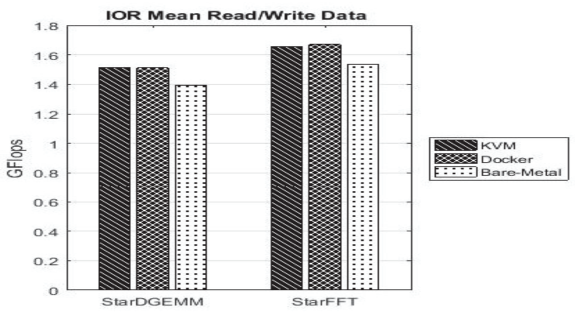

Figure 2 StarDGEMM and StarFFT results

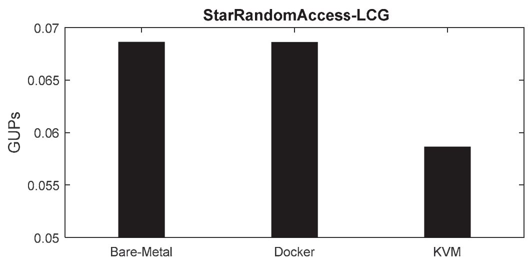

Figure 3 StarRandomAccess_LCG results

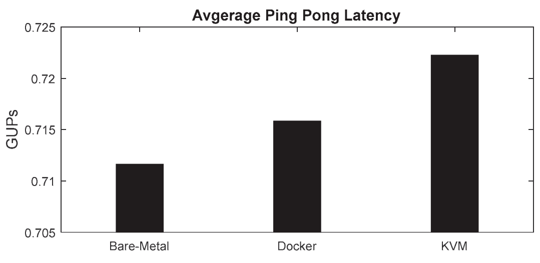

Figure 4 AvgPingPongLatency results

The results of the HPCC benchmark tests are the representation of compute and interconnect performance of systems. The following radar chart compares the HPCC results in bare-metal, Docker, and KVM environments. The radar plot is a normalized chart with the best (highest) result from all the data files assigned with a 1.0 value and the other values normalized to this result.

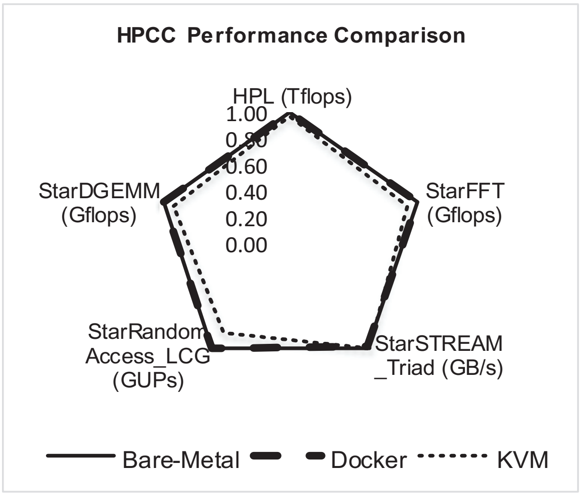

Figure 5 HPCC result comparison

As Figure 5 shows, the performance of the bare-metal closely overlaps with the performance of Docker in general. On the other hand, KVM has less performance in comparison with the other two execution environments. The top vertex of the pentagon shape is the HPL benchmark result. In HPL, the vast majority of compute operations are spent in double precision floating point multiplication of a scalar with a vector and adding the results to another vector. The execution has fairly regular memory accesses and mainly stresses the floating-point capability of cores. As Figure 5 shows, performance is almost identical on both bare-metal and Docker. However, the KVM performance was 2.33% less than the bare-metal, which shows the cost of abstracting and hiding the nature of the system from the execution environment.

Similar to HPL, DGEMM and FFT primarily perform mathematical floating point operations on double precision real matrix-matrix multiplication and double precision complex one-dimensional Discrete Fourier Transform (DFT), respectively. However, since their executions require more accesses to memory than those of HPL, they have relatively worse performance than that of HPL. If we consider the bare-metal performance as the base case, StarDGEMM and StarFFT achieved 7.93% and 7.25% less Gflops, respectively, which again represents the costs of hiding system information from the execution environments.

The bottom right vertex of the pentagon shape represents the STREAM benchmark, which measures sustainable memory bandwidth when performing simple operations on vectors. The benchmark consists of four components COPY, SCALE, ADD, and TRIAD. These components compute the value of $a[i]=b[i]$, $a[i]=q*b[i]$,

$a[i]=b[i]+c[i]$, and $a[i]=b[i]+q*c[i]$, respectively. The benchmark execution is dominated by the memory bandwidth and the benchmark's workload is significantly larger than caches. The benchmarks have regular memory access pattern. The performance was almost identical on both bare-metal and Docker. However, the KVM performance was clearly worse than the bare-metal especially in COPY, SCALE, and ADD cases. If we consider the result of bare-metal as the base reference, the sustained bandwidth rate (GB/s) of each component for KVM was reduced by 15.94%, 15.58%, and 14.34%, respectively. Although there was significant performance overhead in COPY, SCALE, and ADD components, there was only 1.40% slowdown in TRIAD component (as shown in Table I). However, the overall performance overhead in KVM shows the costs of adding another layer of abstraction to the execution environment.

Table I StarSTREAM result

Function	Rate (GB/s) for Bare Metal	Rate (GB/s) for Docker	Rate (GB/s) for KVM
Copy	6.4097	6.3992	5.3879
Scale	6.3335	6.1996	5.3468
Add	6.8339	6.8847	5.8538
Triad	7.2255	7.1303	7.3263

The last reported benchmark result of HPCC is Random Access, which is shown in the bottom left vertex of the pentagon shape (Figure 5). In contrast to the STREAM benchmark, which has regular memory access pattern, Random Access benchmark stresses random memory patterns. Similar to the STREAM benchmark, the workload is significantly larger than caches. The benchmark strongly exercises the hardware page table walker that handles TLB misses. As a consequence, the result of the benchmark is determined by the latency of the hardware page table walks and main memory load latency. The virtualized environment has been observed with clear overhead too.

3.2 IOR results and analysis

IOR (Interleaved Or Random) is a benchmark program used for benchmarking storage systems. The CTBS benchmark run gathers the performance information using three different file I/O APIs: POSIX Buffered-IO, POSIX DirectIO, and MPI-IO.

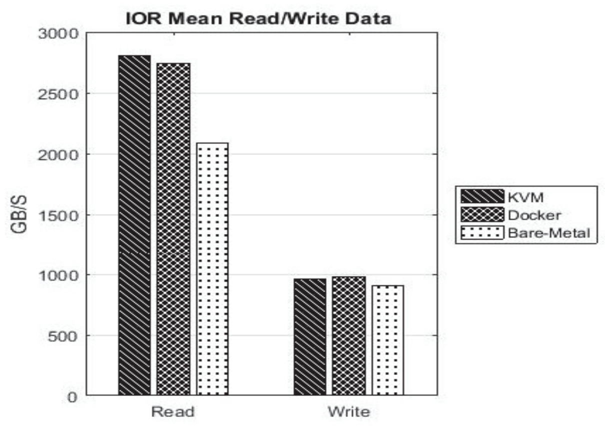

Figure 6 IOPS comparison: bare-metal vs. Docker vs. KVM

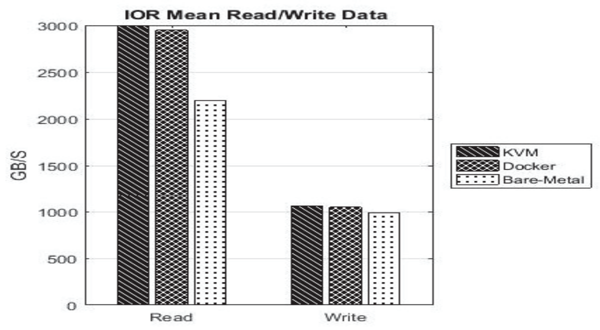

Figure 7 IOPS comparison with POSIX Buffered IO

Due to the page limit, we show a few sets of representative results in Figure 6 and 7. Figure 8 compares the IOR results with three different file I/O APIs, POSIX Buffered-IO, POSIX DirectIO, and MPI-IO with running the random IOPS operations in bare-metal, Docker, and KVM environments, respectively. Similarly, the radar plot is a normalized chart with the best (highest) result from all the data files assigned a 1.0 value and the other values normalized to this result. As Figure 8 shows, the performance of the bare-metal closely overlaps with the performance of Docker in general. On the other hand, KVM achieved less performance in comparison with the other two execution environments.

As the results show, KVM is relatively weak in Read IO performance. For Random Read in KVM execution environment, we observed that there were 25.55% and 26.67%, reduction in mean achievable IOPS respectively for MPI-IO and POSIX Buffered IO APIs when compared to the bare-metal case. While the result shows that KVM does not perform well for Read IO, we observed almost the same performance for the Random Write operating in all the three APIs compared to the bare-metal configuration. Moreover, we observed almost identical performance on both bare-metal and Docker within marginal difference of 2% (including noise).

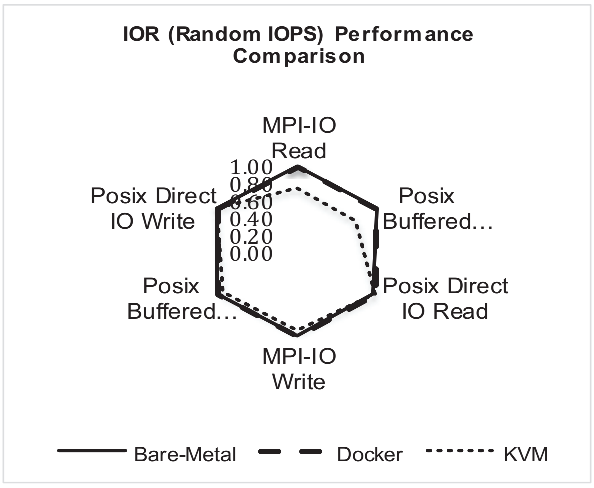

Figure 8 IOR - random IOPS performance comparison, bare-metal vs. Docker vs. KVM

Table II shows the percentage of slowdown or speedup of the random IOPS throughput of each API for Read and Write operations, and Table III shows that of the bandwidth measurement. In general, these results show that the containerized environment clearly outperformed the KVM. The virtualized environment incurs a significant overhead and is also sensitive to workloads.

Table II Random IOPS slowdown/speedup based on bare-metal: Docker v.s. KVM

	Read			Write		
	MPI-IO	POSIX Buffered IO	POSIX Direct IO	MPI-IO	POSIX Buffered IO	POSIX Direct IO
Docker	-2.07%	-1.51%	3.37%	1.42%	-0.55%	0.00%
KVM	-25.55%	-26.67%	2.25%	-6.52%	-6.35%	0.00%

Table III Bandwidth slowdown/speedup based on bare-metal: Docker v.s. KVM

	Read			Write		
	MPI-IO	POSIX Buffered IO	POSIX Direct IO	MPI-IO	POSIX Buffered IO	POSIX Direct IO
Docker	-0.18%	-0.36%	-3.82%	-9.42%	-4.02%	-0.54%
KVM	-30.73%	-20.24%	3.94%	-7.03%	-3.96%	-29.05%

4 Related work

Numerous recent studies have focused on benchmarking the performance of virtualization in cloud environments. For instance, Li et al. [3] conducted a performance test for MapReduce workloads of three different hypervisors, two open source hypervisors Xen and KVM, and a commercial one, CVM. Their results do not show any significant difference between the performance of these three hypervisors for CPU-bound workloads. For IO-bound workloads, there is a major difference between the performance of these three hypervisors based on the size, and the type of the workloads, and the number of virtual machines.

Another example is the literature study [6], where the authors conducted a performance benchmarking with two different virtualization methods, Citrix XenServer 5.5, and VMware ESX 4.0. The IO performance of virtualized cloud computing systems was examined in [4], using the IOR benchmark, to compare two different testbeds, Amazon cloud and Magellan cloud. The results show that the performance of local storage systems is better than block store volumes and shared file systems. In [5] the authors introduced a methodology for ranking virtual machines in cloud systems based on the results of cloud performance benchmarking and weighting technique.

Different from these existing efforts, our research focused on evaluating and comparing bare-metal, containerized, and virtualized environments at the IaaS layer of cloud systems.

5 Conclusion

For a long time, cloud administrators have had multiple virtualization options to choose from. Recently, Linux containers, especially Docker container, have emerged as another virtualization method. In this research, we used Cloud Tester Benchmark Suite to study and understand the performance of systems in three different execution environments, bare-metal, Docker, and KVM. Based on benchmarking results and analyses, we observed that Docker outperformed KVM in most cases. We also observed almost identical performance on both bare-metal and Docker in numerous cases. Using the HPCC benchmark, we found that Docker delivered near bare-metal performance while KVM performance was relevantly less, especially in Random Access benchmark that KVM suffered 14.55% degradation in comparison with bare-metal. Moreover, we found that since KVM incurs overhead to I/O operation, it is less suitable for workloads that are latency-sensitive or have high I/O rates. For example, we observed 25.55% and 26.67% reduction in mean achievable IOPS respectively for MPI-IO and POSIX Buffered IO APIs when compared to the bare-metal.

Since containers offer the control and isolation of virtual machines with the near bare-metal performance, cloud providers can build the IaaS layer based on containers. Features like fast deployment and near bare-metal performance are attractive to use containers as the building block of IaaS. However, virtual machines have existed for a long time and features like live migration across platforms, over allocation, and snapshots can be challenging for containers to substitute virtual machines. As a result, there is a tradeoff between the ease of management and performance.

Acknowledgements

This research is supported by National Science Foundation under grant IIP-1362134 through the

StackVelocity membership contribution and the Aerospace Corporation technical partnerships.

References

[1] Docker, available at: https://www.docker.com.

[2] D. Enright, A. Brethorst, J. Everist, J. Stoermer, B. Winkenbach and L. Wang, "Cloud Benchmarking & The Cloud Tester Project", The Aerospace Corporation, 2014.

[3] J. Li, Q. Wang, D. Jayasinghe, J. Park, T. Zhu, C. Pu, "Performance Overhead Among Three Hypervisors: An Experimental Study using Hadoop Benchmarks", IEEE International Congress on Big Data, 2013.

[4] D. Ghoshal, R. Shane Canon, L. Ramakrishnan, "I/O Performance of Virtualized Cloud Environments", DataCloud-SC'11, Seattle, Washington, USA. Copyright 2011 ACM 978-1-4503-1144-1/11/11, 2011.

[5] B. Varghese, O. Akgun, I. Miguel, L. Thai, A. Barker, "Cloud Benchmarking for Performance", IEEE 6th Intl. Conference on Cloud Computing Technology and Science, 2014.

[6] N. Huber, M. von Quast, M. Hauck, and S. Kounev, "Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments", In Proc. of the 1st Intl. Conference on Cloud Computing and Services Science (CLOSER), 2011.