Impact of Virtualization and Containers on Application Performance and Energy Consumption*

Thomas Huber
University of Delaware
Newark, Delaware
thuber@udel.edu

Junjie Li
Indiana University
Bloomington, Indiana
lijunj@iu.edu

Sunita Chandrasekaran
University of Delaware
Newark, Delaware
schandra@udel.edu

Robert Henschel
Indiana University
henschel@iu.edu

ABSTRACT

It is widely accepted that applications achieve superior performance when running on traditional native High Performance Computing systems; however, the practicality and adaptability of virtualization and its impact on cloud computing is undeniable. With the ever increasing speed, storage, and accessibility of computer hardware, the average researcher or domain scientist no longer needs to concern themselves with maximizing efficiency of their computation. Instead, these users are focused on finding the most intuitive platforms that offer a simple interface for managing Virtual Machine instances. Hypervisor based virtualization techniques, like the popular KVM(Kernel Virtual Machine), enable users to access cloud computing systems more conveniently, eliminating the difficulty and imperceptibility of using traditional HPC systems. Here, we compare performance of a single node on the native Jetstream cloud system versus a Virtual Machine running on single node on said system, and measure the overhead associated with KVM, OpenStack, and other services unique to a VM instance running Jetstream. THE SPECOMP2012 benchmark suite, comprised of fifteen scientific codes, is used the main performance metric for evaluating the performance of both systems, the native Jetstream node and the VM instance. Results support that memory intensive applications incur a great degradation in performance when running on a VM instance. Compared to performance of the native system, some of these applications in question saw 40 percent decrease in performance.

CCS CONCEPTS

• **Computer systems organization → Measuring Performance**; *Cloud Computing*;

KEYWORDS

Virtual Machine, Benchmarking, Cloud Computing

ACM Reference Format:
Thomas Huber, Junjie Li, Sunita Chandrasekaran, and Robert Henschel. 2018. Impact of Virtualization and Containers on Application Performance and Energy Consumption. In *Proceedings of PEARC Conference (PEARC'18).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.475/123_4

*Produces the permission block, and copyright information

1 INTRODUCTION

Cloud services, like AWS and Google Cloud, provide virtual computing resources to users, and allows them to spin-up virtual machine instances very easily. Indiana University joined this market-space several years ago with the addition of NSF funded Jetstream to the XSEDE system [4]. Jetstream, akin to AWS and Google Cloud, makes it easier for students, researchers, and professors to access computing resources and use popular software packages without any configuration, building, or chasing down dependencies. Users, using a graphic interface, can chose from a wide array of VM images, often times pre-configured with licensed software and popular packages, and begin running applications within minutes of creating an account. The typical user of this flavor of cloud system will most likely not be concerned with the performance of their applications. Nevertheless, for system administrators, software developers, and network engineers, it is valuable to have accurate performance metrics evaluating the most popular virtualization technique for clouds, KVM with OpenStack, by using one of the most widely-accepted computer benchmarks for cloud systems, SPECOMP2012.

It is vital to understand that when comparing the native node's performance to the performance of a Virtual Machine instance running atop a different native node, it is not a perfect feature-for-feature comparison. Due to the abstraction of the hardware and services required to create, manage, and assign a virtual machine, the native system undoubtedly has an advantage. Because of these reasons, this analysis serves not as direct comparison between the two, but instead as an accurate measure of how a user performs in a virtual machine instance versus how a user would perform if they were able to run as user with access to the native hardware.

1.1 State of the Art

In 2011, a group from the Pervasive Technology Institute at Indiana University compared and contrasted the most popular virtualization methods (Xen, KVM, Virtual Box), in an attempt to determine the most viable option for supporting HPC applications within a cloud infrastructure. SPEC OMP2001 and HPCC benchmark suites were used to evaluate the native system, and compare performances of several popular virtualization techniques running on the native system host OS. The results they produced led to the determination that KVM is the most viable option for a multitude of supported reasons [5]. The authors of this report saught to create a neutral test environment that inhibits a fair comparison between popular virtualization techniques.

In this paper we will do the opposite and, instead of trying to compare and contrast different virtualization techniques, we observe a non-synthetic cloud system that uses KVM and is already in production. The running cloud system, Jetstream, already utilizes a unique setup of both KVM and Openstack to support virtualization, and we aim to produce a fair performance evaluation of user code running in a virtual machine instance on the system. In comparing these results to the native performance of user code without virtualization, it will allow for an understanding of the relative overhead that exists in a unique cloud system such as Jetstream. This is not the first journal to rely on SPEC Benchmarks as a reliable metric, particularly because they are well regarded as a general stress test of HPC systems. There are several comparisons done on the performance of KVM vs native system, but only with the older SPEC OMP2001 benchmark. Remarkably, these older results allude to KVM having barely a 1 percent overhead, as measured by comparing VM vs native SPEC OMP2001 scores [5].

1.2 Experimental Setup

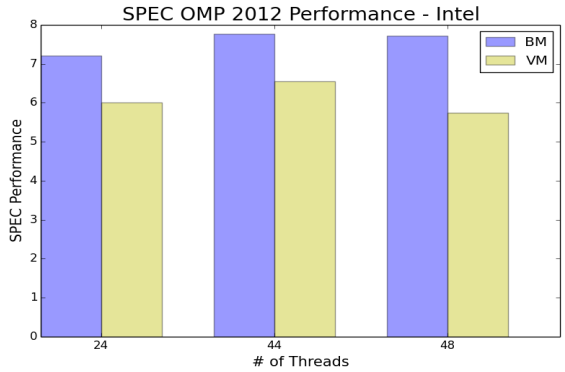
The Jetstream Cloud system is comprised of multiple cooperating entities—Jetstream-IU, Jetstream-TACC, and Jestream-AZ. The systems residing at the Indiana University and Texas Advanced Computing Center (TACC) are identical and maintain the same hardware; each offering 320 DELL M630 blade servers for user computation, amounting to 640 CPUs and 15,360 physical cores [1] [4]. Each individual blade server includes two Intel Haswell architecture E5-2680v3 2.5 GHz CPU. In order to evaluate single-node performance of this system, we will focus on measuring a single CPU node residing on only one of Jetstream's blade servers. The single node consists of 2 sockets—12 CPU cores per socket—which, with hyperthreading turned on (2 threads per core), can amount to 24 real cores for 48 theoretical/virtual cores. PGI Community Edition 18.4 and Intel(R) Parallel Studio XE 2018 Update 3 Cluster Edition are the compiler versions used to run SPECOMP2012 and STREAM benchmarks. The -fast compiler option is used for PGI and Intel. Both the virtual machine instances and the native host use a CentOS 7.5 Linux release image.

1.3 Benchmarking Setup

In order to accurately measure the performance penalty on applications running within a Jetstream VM instance, it is vital to use a standardized benchmark. Both the SPECOMP2012 and HPCC benchmark suites are staples in the HPC community, utilized to measure performance of shared-memory parallel processing and a wide range of computationally intensive parallel tasks, respectively. Because the SPEC benchmarks test computing systems with a wide spectrum of tasks—searching and sorting, linear algebra, matrix calculations, memory movement—they are the optimal benchmarks to consider when analyzing and drawing general conclusions regarding overall performance of a system.

Although the HPCC suite consists of seven individual and unique benchmarks, the authors chose to only use STREAM and HPL benchmarks in order to measure the main memory access speed and floating point performance, respectively. A fully reportable SPECOMP2012 run was completed for 24, 44, and 48 threads. For STREAM benchmark from HPCC suite, the same thread counts were used. Results were reproduced with PGI and Intel compilers. The SPEC OMP2012 benchmark is the most updated version of SPEC HPG's OpenMP benchmark suite, commonly used to evaluate shared memory parallel processing. Consisting of fifteen scientific application codes, the 2012 benchmark tests run with various programming languages, code size, memory demand, and amount of OpenMP usage [3]. For a reportable run, all of the application codes run three times. The scores are then averaged individually, and then combined to form an overall total run score.

Figure 1: Results of SPECOMP2012 Benchmark Run With Intel compiler



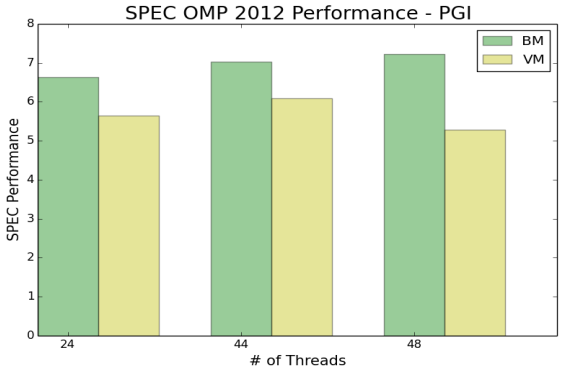
Intel − #ofThreads	*Native*	*VM*	*%performanceloss*
24	7.21	6.01	-16.6
44	7.77	6.55	-15.7
48	7.72	5.74	-25.6

PGI − #ofThreads	*Native*	*VM*	*%PerformanceLoss*
24	6.64	5.64	-15.0
44	7.03	6.09	-13.3
48	7.23	5.28	-26.9

1.4 SPECOMP2012: Results Explained

Reportable runs of SPECOMP2012 on Native and Virtual Machine systems, for both PGI and Intel compilers, were reproduced for 24, 44, and 48 threads. For the 24 and 44 thread runs, both compilers showed an approximately 15 percent performance degradation in the virtual machine instance (Figure 1, Figure 2). For the 48 thread

Figure 2: Results of SPECOMP2012 Benchmark Run With PGI compiler



run, again with both compilers, there is an even greater degradation in performance, approximately 26 percent.

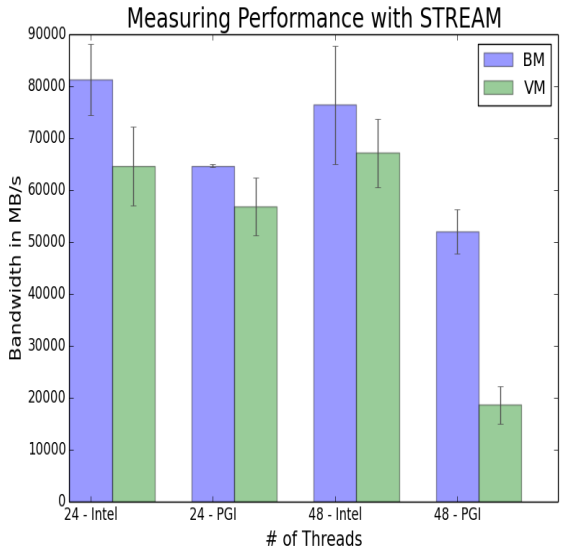
It is pertinent to understand that the greater amount of overhead observed in the 48 thread run is a direct result of the overall different number of CPU cores between the two systems, and is disproportional to results for thread counts less than 44. Because even the largest virtual machine instance running on the host OS can only be 44 hyper threaded cores, and the native system will always have 48 hyper threaded cores, any parallel applications requiring more than 44 threads will immediately see a significant decrease in performance on the VM.

1.5 Memory Bound Applications

The output from the SPECOMP2012 benchmark assigns scores to each of the 15 scientific applications it is comprised of. Analyzing the results of the individual applications on VM instance and native system, it was clear the applications that had the greatest performance degradation between the two systems were those with high memory demand—greater than 10,000 Mbs. To support the SPECOMP2012 results and further investigate if the memory intensive applications truly perform much worse on virtual machine, STREAM benchmark [2] was used to evaluate main memory direct access speeds in Mb/s for both systems. STREAM benchmark was run five times on both systems with both compilers. Results for STREAM benchmark were obtained by first rebooting both the host underlying the virtual machine image and the separate native host used to test native performance. This was repeated for every run of STREAM benchmark to ensure proper test environment and avoid any potential memory fragmentation.

Although variability was extremely high when measuring both systems, the results consistently show a minimum 15 percent performance degradation when accessing main memory on VM side with 24 and 48 threads. This was true for both PGI and Intel compilers (Figure 3). The results for STREAM benchmark with 48 threads using PGI compiler showed a performance reduction of both systems, proving that testing VM system with over 44 threads will be always be unreasonable (Figure 3). It is worth mentioning that the strictly computation intensive applications in SPECOMP2012 performed with very little performance degradation on the VM side, about 3 percent. This is supported by the original HPL linpack benchmark results from when the Jetstream system was first evaluated, also showing about 3 percent degradation on VM.

Figure 3: Results of the STREAM benchmark, reported in MB/s, for both PGI and Intel compilers.



1.6 An Imperfect One-To-One Comparison

Due to the uniqueness of the Jetstream cloud system, it becomes impossible to create a one-to-one performance comparison between a Virtual Machine instance and the native host. The services that Jetstream requires to operate a VM instance—KVM, OpenStack, Atmosphere— amount to a significant amount of overhead on the host OS side. Another variable to consider, which ultimately has a deficit on the VM performance, is the security patch related to Spectre and Meltdown exposure. With multiple complex services intermingling, it would be difficult to isolate KVM or OpenStack and measure solely the performance of just one of the services in a VM. Considering the complexity of this system, we have chosen to not evaluate individual components of the cloud system, but rather to strictly evaluate the performance of an XSEDE user on a Jetstream VM instance, and compare this to the performance of an application running on the native as if it were a true HPC system.

In order to ensure accuracy and ability to reproduce results when running benchmarks on the VM, we made sure the native host underlying the virtual machine image was isolated from the other nodes. To properly accomplish this, we prevented any jobs from being scheduled on the VM's underlying host, as well as preventing the VM from being switched to a different native host. This was executed in a manner by which OpenStack and Atmosphere API services were able to remain on, but would not communicate with other running instances in the cloud system.

2 CONCLUSIONS

Regardless of the related overhead or associated cost, platforms for launching Virtual Machine instances will continue to be the primary service option for users looking to run programs on powerful computers. While traditional HPC systems force users to deal with job submission and the possibility of their job waiting in a queue for days, the option to simply launch a VM and begin running code that day is far more appealing to the standard user. Although this is the general case for the average users, there are exceptions. For researchers, scientists, and HPC users alike, there are many cases where minimizing overhead and achieving maximum performance is necessary. Whether it be for reducing energy consumption, run times, or total number of resources needed, users may stray away from VM platforms and instead rely on traditional HPC systems with smaller job submission rates or generally smaller jobs.

Users concerned with minimizing overhead —especially in Genomics, low latency trading, and atmospheric research— should be concerned about the large performance degradation observed in a virtual environments. Specifically, anyone running large embarrassingly parallel applications that require data movement between host OS and hypervisor's guest OS, frequent indirect memory access, and read/writes to main memory need to be aware of the performance degradation observed in this report. Compared to earlier reports that present a similar evaluation of virtualization techniques, we observed a far greater overhead and performance degradation associated with programs running in VM.

3 FUTURE WORK

It would be valuable to understand exactly why performance for memory intensive applications when operating in a hypervisor's guest OS is extremely poor. In the future, taking a closer look into low-level interactions between the cache and main memory on both VM and native side would give insight into how the system deals with memory transfer between the hypervisor. If we were to observe real-time memory related activity on both the VM instance and the host OS underlying the VM instance, a greater understanding of how exactly this overhead is incurred would be achieved. Also, reproducing results using SPEC HPG's benchmark that targets accelerators, SPECACCEL, would give insight into virtualization impact on performance of devices such as GPUs.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1445604. This work was also supported by Indiana University. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or Indiana University.

REFERENCES

[1] Jeremy Fischer, Steven Tuecke, Ian Foster, and Craig A. Stewart. 2015. Jetstream: A Distributed Cloud Infrastructure for Underresourced Higher Education Communities. In *Proceedings of the 1st Workshop on The Science of Cyberinfrastructure: Research, Experience, Applications and Models (SCREAM '15)*. ACM, New York, NY, USA, 53–61. https://doi.org/10.1145/2753524.2753530

[2] John D McCalpin. [n. d.]. STREAM benchmark. ([n. d.]).

[3] Matthias S. Müller, John Baron, William C. Brantley, Huiyu Feng, Daniel Hackenberg, Robert Henschel, Gabriele Jost, Daniel Molka, Chris Parrott, Joe Robichaux, Pavel Shelepugin, Matthijs van Waveren, Brian Whitney, and Kalyan Kumaran. 2012. SPEC OMP2012 – an Application Benchmark Suite for Parallel Systems Using OpenMP. In *Proceedings of the 8th International Conference on OpenMP in a Heterogeneous World (IWOMP'12)*. Springer-Verlag, Berlin, Heidelberg, 223–236. https://doi.org/10.1007/978-3-642-30961-8_17

[4] Craig A. Stewart, David Y. Hancock, Matthew W. Vaughn, Jeremy Fischer, Tim Cockerill, Liming Lee, Nirav Merchant, Therese Miller, John Michael Lowe, Daniel C. Stanzione, James Taylor, and Edwin Skidmore. 2016. Jetstream: performance, early experiences, and early results. In *XSEDE*.

[5] A. J. Younge, R. Henschel, J. T. Brown, G. von Laszewski, J. Qiu, and G. C. Fox. 2011. Analysis of Virtualization Technologies for High Performance Computing Environments. In *2011 IEEE 4th International Conference on Cloud Computing*. 9–16. https://doi.org/10.1109/CLOUD.2011.29