IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 20, Issue 1, Ver. II (Jan.- Feb. 2018), PP 36-43
www.iosrjournals.org

Testing Docker Performance for HPC Applications

Alexey Ermakov, Alexey Vasyukov
Corresponding Author: Alexey Ermakov

Abstract: *The main goal for this article is to compare performance penalties when using KVM virtualization and Docker containers for creating isolated environments for HPC applications. The article provides both data obtained using synthetic tests (High Performance Linpack) and real life applications (OpenFOAM). The article highlights the influence on performance of major infrastructure configuration options – CPU type presented to VM, networking connection type used.*
Keywords: *Docker, KVM, MPI, HPL, OpenFOAM, benchmark.*

Date of Submission: 20-01-2018 Date of acceptance: 13-02-2018

I. Introduction

One of the most important issues related to high performance computing that one may encounter is the availability of certain execution environment. It means that many scientific programs require a specific set of dependencies (such as compilers, runtime libraries etc.), that often may even conflict with dependencies of other software. There is a number of ways to solve the issue; one the most mature technologies that is used for such a purpose is a virtualization. Despite the fact that virtualization provides full environment isolation, by-design it has some performance penalty. Another approach to provide isolated environment is operating-system-level virtualization that implies all such environments have common kernel and separate isolated user-space libraries. The main goal for this article is to compare performance penalties when using two mentioned ways of creating isolated environment (KVM and Docker containers, to be precise).

II. Related work

Cloud computing environments for HPC applications are commonly based on KVM for virtualization and isolation and OpenStack for cluster management, auto-provision and user self-service. An example of the system based on these technologies can be found in [1], describing an experience of TechnischeUniversitat Dresden. Similar KVM-based clusters are deployed in different organizations over the world. However, performance penalties for real life applications may be significant when running in virtualized environment [2]. Container-based systems for HPC applications emerge during recent years [3], [4], [5] and benchmarks look promising [6], [7]. This article also contributes to public benchmarks of KVM and Docker containers for HPC applications.

III. Virtual machines and containers

The main difference between virtualization and containerization is that containers share the same kernel and maybe even some host devices, when each virtual machine has its own kernel and virtualized devices (e.g. network card)1. Detailed information on the technologies used for this research may be found in official documentation for KVM [8], QEMU [9] and Docker[10]. This difference in architecture is outlined on fig. 1. One might expect lower performance overhead of containers compared with traditional virtual machines. However, scientific high performance applications typically use a lot of low level optimizations to achieve maximum calculation speed. So, detailed testing is required to ensure, if virtualization and containerization impact their performance or not.

1 In this article we do not consider usage of paravirtualization or any “passthrough” technologiesto make host devices available to virtual machine

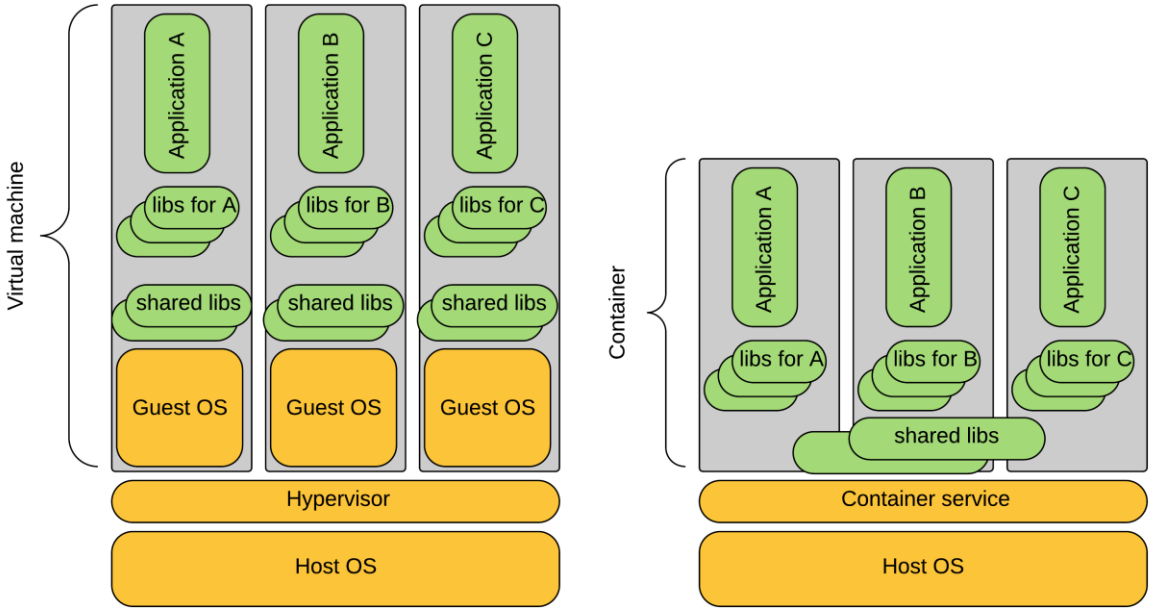

Figure 1: Comparison of virtualization and containers architecture

IV. Benchmark setup and methodic

To perform benchmarks the following setup was used: two identical hosts with Intel Core i7-5820K CPU (6 physical cores) and 64 GB RAM, connected with QDR Infiniband and 100 MB/s Ethernet. Hyperthreading was disabled using corresponding BIOS settings, since it drastically decreases performance (see fig. 2). MPICH was used as MPI implementation, because it's a bit faster than OpenMPI (see fig. 3 and 4) and does not require any configuration to execute program on two hosts when they belong to different subnets (this is very important to be able to inter-connect virtual machines and containers.)

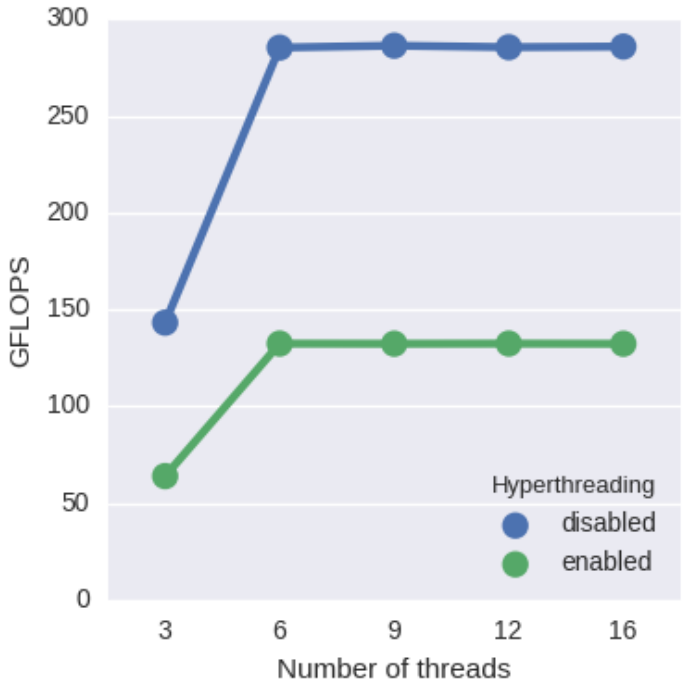

Figure 2: Hyperthreading performance impact according to Intel Linpack benchmark results

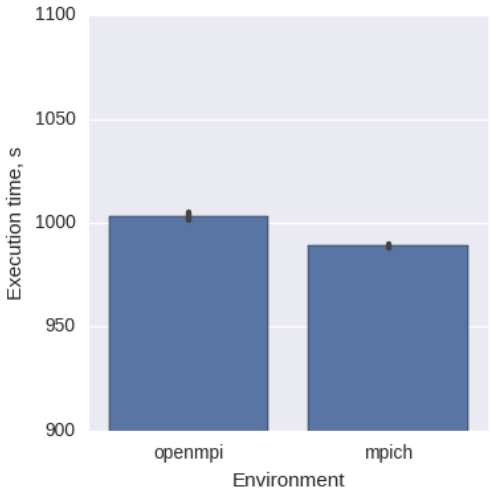

Figure 3: Comparison of OpenMPI and MPICH performance on QDR infiniband connection using HPL

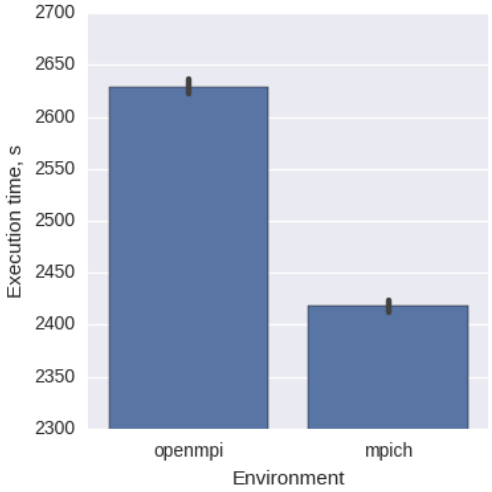

Figure 4: Comparison of OpenMPI and MPICH performance on 100 Mb/s Ethernet connection using HPL

Benchmarks described in this article use Intel Linpack benchmark[11], High-performance linpack[12] and interFoam solver as "real world" application[13]. All experiments were run 10 times to reduce statistical errors, so each plot shows mean value for measurement and error bars for confidence interval of 0.95.

V. Benchmark

5.1 Single host benchmarks

First of all let's see how usage of containers and virtualization impacts performance. The tests in this subsection were performed on a single host to avoid networking influence on performance results.

Intel Linpack results are shown on a fig. 5, and these results should be treated as follows: there is no significant difference in performance when running CPU-intensive highly-optimized application in KVM, Docker or on bare metal. It should be noted that in these tests Intel Linpack demonstrates 90% of theoretical CPU performance, thus performance comparison may be considered reliable. We can see that Docker shows a bit better performance even than bare metal, but it should be considered as statistical error. Another cause for this may be operating system scheduler that for some reason gives a bit more priority for containerized processes.

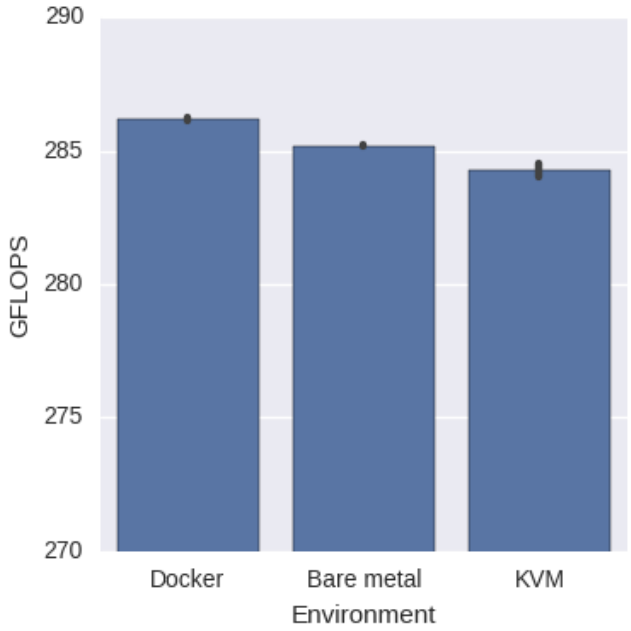

Figure 5: Performance comparsion of KVM and Docker on a single host using Intel Linpack

In the previous test QEMU was run with host-model CPU set, that's why it showed pretty good performance. However, host operating systems configuration may influence this result significantly. In case of different CPU type presented to a virtual machine there is a huge performance spread: depending on exact CPU model used to run a virtual machine the result may be up to 5 times slower than a bare metal. An example is presented on fig. 6 – a performance of an application inside KVM virtual machine depends heavily on CPU presentation to this machine by the host system.

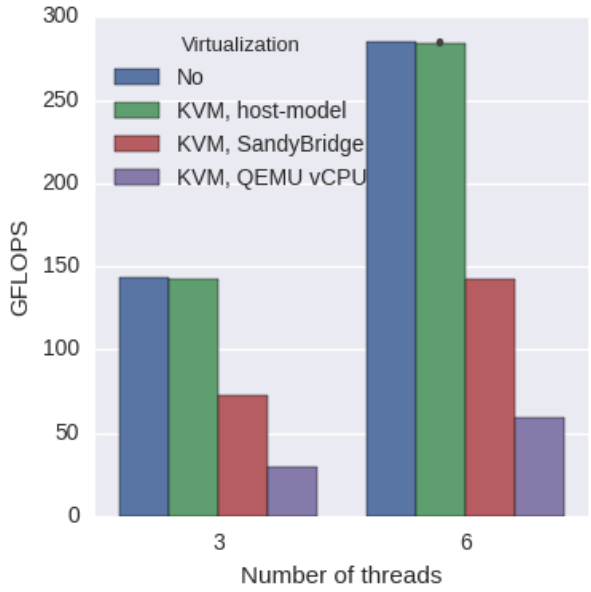

Figure 6: KVM performance spread

5.2 Two hosts benchmarks

Another component, besides CPU, that has a major performance influence is networking. The next series of tests demonstrate what performance impact one may have when using unappropriate networking stack.

First thing to note is quite obvious but anyway should be mentioned: networking type matters. According to fig. 7 HPL performs about 2.5 slower on 100 Mb/s Ethernet than on QDR Infiniband.

Another thing that should be taken into account is the way virtual machine or container is connected to network. For HPL tests there is almost no difference (see fig. 8) in performance of dockerized network application between bridged connection and host networking stack, KVM virtio performs about 15% slower, and KVM rtl is almost 5 times slower (see fig. 9) than host netwoking stack.

Earlier we've noticed that CPU type used to run virtual machine has significant influence on overall performance. Let's see if it still applies to distributed MPI application. According to pic. 10 CPU used to run virtual machine does not affect resulting performance. Reason for this behaviour is the fact that HPL is not as CPU-intensive as Intel Linpack: network performance is more important for HPL rather than CPU performance.

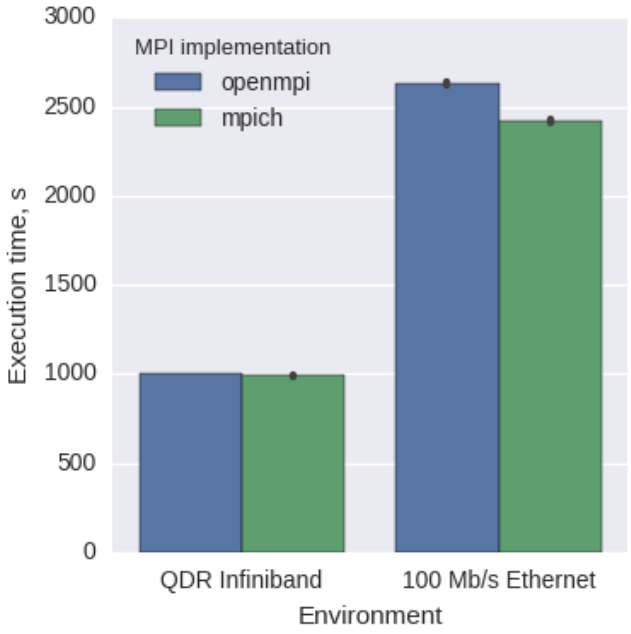

Figure 7: HPL performance comparsion using different inter-connection

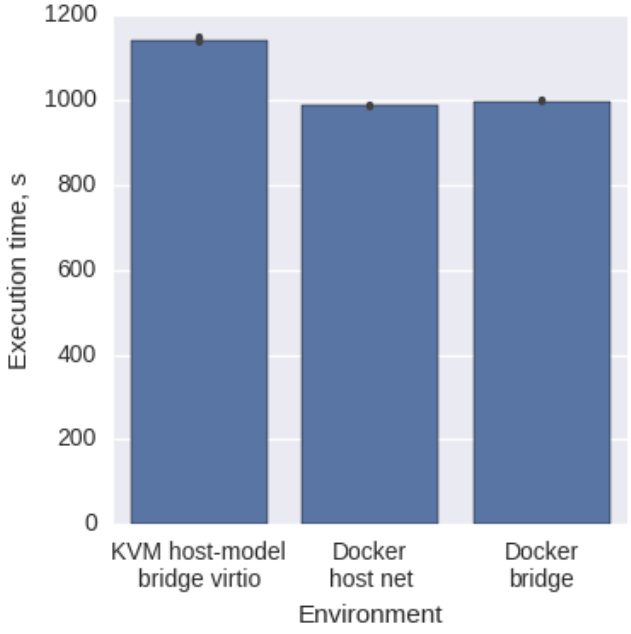

Figure 8: Docker and KVM networking performance comparsion

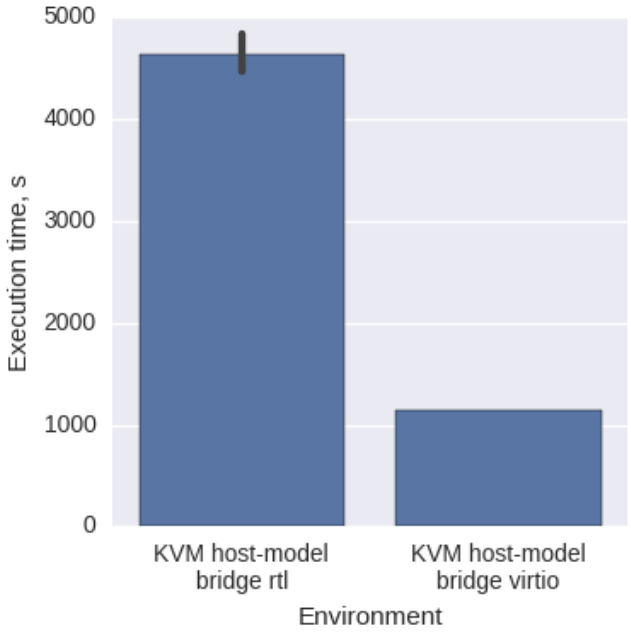

Figure 9: virtio and rtl networking performance comparsion

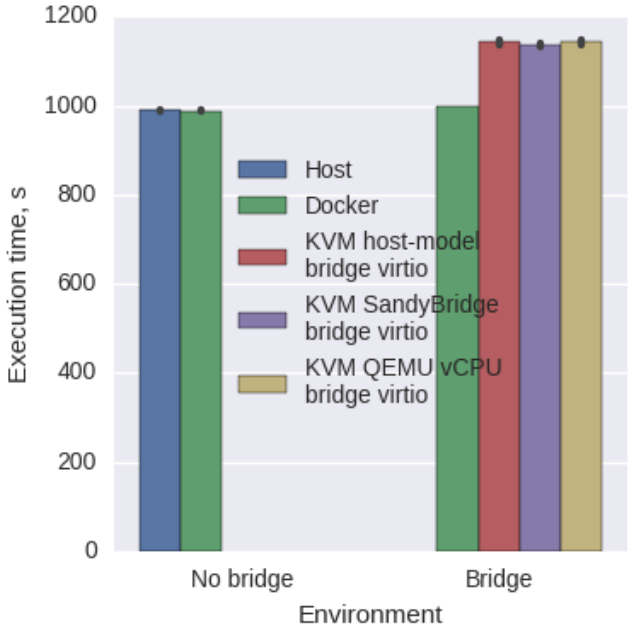

Figure 10: CPU type influence on performance of HPL

5.3OpenFOAM benchmarks

Finally, let's see how "non-synthetic" distributed MPI application performs depending on used environment. We used interFoam solver from OpenFOAM in few scenarios to realize how virtualization type and kind on networking connection influences overall performance. As we can see on fig. 11, interFoam performs almost 10 times slower on 100 Mb/s Ethernet rather than on QDR Infiniband. The application depends heavily on network performance. Fig. 12 shows that bridge performance impact is about 20% for interFoam solver running on QDR Infiniband.

The most important results are shown on fig. 13: a real "non-synthetic" application is more than 2 times slower in KVM with virtio networking than the same application in Docker with host or bridged networking.

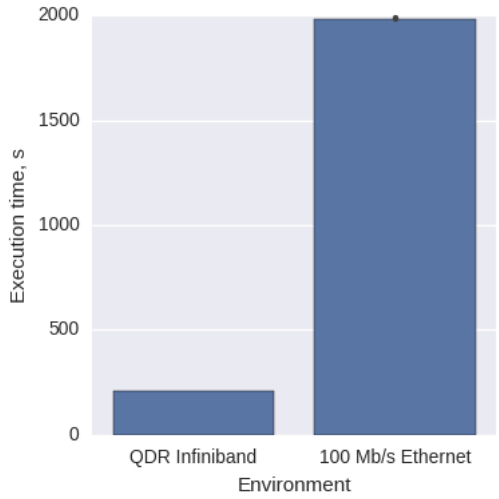

Figure 11: Networking type influence

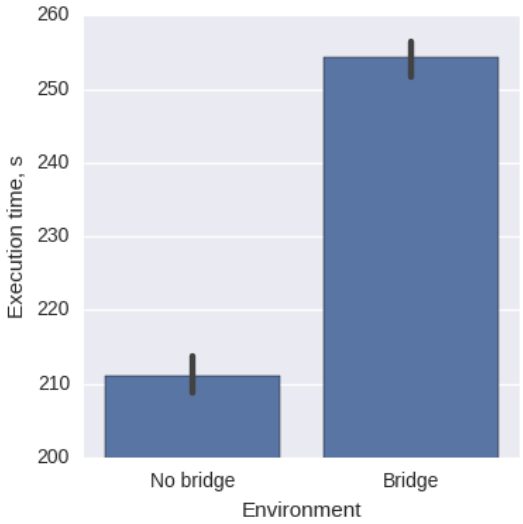

Figure 12: Network connection type influence

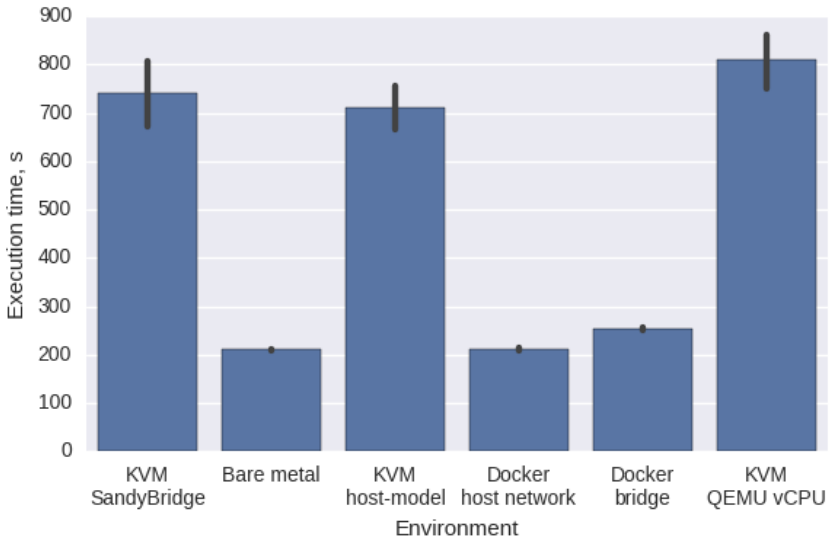

Figure 13: Comparsion of interFoam performance depending on used environment

VI. Conclusion

Performance of virtualized or containerized applications depends on many factors, such as CPU type (for virtualization) and networking type. In some cases performance may degrade up to 10 times, thus environment to run application in must be carefully selected and verified. A really important thing is that "synthetic" benchmark does not provide you with a full exhaustive information necessary to decide if environment is suitable for an application or not. That's why it's strongly recommended to run benchmarking on exact application you're going to run when considering virtualization or containerization as an option for HPC.

Acknowledgements.

The research was supported by Russian Foundation forBasic Research grant 15-29-07096.

References

[1]. U. Markwardt. Running virtual machines in a slurm batch system.In Slurm User Group, 2015.
[2]. W. Felter, A. Ferreira, R. Rajamony, and J. Rubio.An updated performance comparison of virtual machines and linux containers. Technical Report Technical Report RC25482 (AUS1407-001), IBM Research Division, Austin Research Laboratory, July 2014.
[3]. D. M. Jacobsen and S. Canon. Contain this, unleashing docker for hpc. In Cray User Group Conference Proceedings, 2015.
[4]. D. Jacobsen, D. Botts, and Canon S. Never port your code again – docker functionality with shifter using slurm. In Slurm User Group, 2015.
[5]. H.-E. Yu and W. Huang.Building a Virtual HPC Cluster with Auto Scaling by the Docker.ArXiv e-prints, September 2015, 1509.08231.
[6]. C. Kniep. Containerization of high performance compute workloads using docker. Technical report, QNIB Solutions, November 2014.
[7]. B. Varghese, L. ThamsuhangSubba, L. Thai, and A. Barker.Container-Based Cloud Virtual Machine Benchmarking.ArXiv e-prints, January 2016, 1601.03872.
[8]. Kernel virtual machine project. [Online]. Available: http://www.linux-kvm.org/page/Main Page (accessed 2016-05-12).
[9]. Qemu process emulator. [Online]. Available: http://www.qemu-project.org/ (accessed 2016-03-17).
[10]. Docker documentation. [Online]. Available: https://docs.docker.com/ (accessed 2016-02-25).
[11]. Intel math kernel library benchmarks. [Online]. Available: http://software.intel.com/en-us/articles/intel-math-kernel-library-linpack-download (accessed 2016-01-11).
[12]. High-performance linpack benchmark for distributed-memory computers. [On-line]. Available: http://www.netlib.org/benchmark/hpl/ (accessed 2016-01-11).
[13]. Openfoam, open source cfd software. [Online]. Available: http://www.openfoam.com (accessed 2016-03-02).

Alexey Ermakov "Testing Docker Performance for HPC Applications." IOSR Journal of Computer Engineering (IOSR-JCE) 20.1 (2018): PP 36-43.