Deliverable #2

Technical Report – Performance Evaluation

By Pijus Mackevicius, Alec Scully, Asante Morris

April 30th, 2019

**Motivation**

This technical report is part of a deliverable for a project aimed to install and configure virtualization containers and automate selected benchmarks inside these containers for performance evaluation using CloudLab infrastructure. The project’s motivation is a study paper published by IBM in 2014 that compares the performances of Native, Docker, and Kernel-based Virtual Machine (KVM) technologies (“An Updated Performance Comparison of Virtual Machines and Linux Containers”). This report serves to assess the project’s progress and evaluate current and predict future results.

**Project Tools**

The team has created a GitHub repository(https://github.com/AS805456/cluster-template) for the project. It is a fork of Dr. Linh Ngo’s cluster-template repository (https://github.com/linhbngo/cluster-template) and the materials are used for CloudLab compatibility and Docker installation. We are also using the code and experiments of the motivation study paper’s team found in their own GitHub repository (https://github.com/thewmf/kvm-docker-comparison) for guidelines for benchmark installation and execution. Our current KVM manual installation code is taken from Cloud Computing course materials.

The deployment infrastructure used in this project is CloudLab (www.cloudlab.us) - a cloud service for scientific and academic research. CloudLab allows the option to upload a GitHub repository which it uses to allocate resources and execute operations defined inside.

**Project Progress**

The team began by creating the GitHub repository denoted in the Project Tools section. All materials, scripts and builds currently used in the project can be found inside this repository. The team has selected two benchmarks for performance evaluation:

* Linpack benchmark – a software library used to perform numerical linear algebra to measure system’s floating-point computing power by calculating dense systems of linear equations (in Floating Point Operations Per Second, or FLOPS)
* Stream benchmark – a program measuring memory bandwidth by performing operations on vectors. The performance is determined by bandwidth to main memory and cache miss’ costs. The benchmark has four measurable components - copy, scale, add and triad, that perform different operations in the kernel (measured in GB/s).

The team has also selected three virtualization technologies to test these benchmarks in:

* Docker (https://www.docker.com/) – a container virtualization software that runs isolated containers with their own contents that can communicate with each other using dedicated channels. All Docker containers use a single operating system kernel, thus working with less overhead than traditional virtual machine technologies.
* Kernel Virtual Machine (KVM) – a device node in the Linux kernel that allows the kernel to function as a hypervisor – a component that allows to create, use and manage virtual machines (VMs) and manages execution of guest operating systems by providing an interface/platform for interaction. The operations provided by the KVM device node include creating and allocating memory to a VM, reading and writing virtual CPU registers, handling interrupts inside a virtual CPU, and running a virtual CPU.
* Singularity – A containerization platform that aims to introduce containers to the world of high-performance computing. Singularity utilizes image files that physically contain a container and acts as a representation of the container environment.

Our GitHub repository contains KVM, Docker, and Singularity branches, which were modified to allow compatibility with CloudLab provisioning and deployment technology. Each branch has a profile.py file, which contains how CloudLab should initiate a virtual node deployment.

To test our benchmarks, our team needed to ensure that we operate with powerful computers able to execute the tasks properly. We aimed for the performance properties of our nodes to be as close to the ones described in the motivational study paper as possible. Currently, all of our nodes that perform testing are initialized with 16 GB of Memory and 4 cores, with an extra 8 GB of space on the disk. Ideally, we would like our nodes to be mapped to the exact same type of physical machines that CloudLab provides; this task is still in progress.

Initially, our team was set on to use the RandomAccess benchmark, which tests speed of memory operations. However, due to unattainability of RandomAccess benchmark that we hoped we could operate with, we have decided to replace it with the Stream benchmark.

**Progress Assessment**

Our team has automated Docker installation using a shell script that is executed when CloudLab instantiates the Docker branch. Docker is available to use immediately after node deployment is complete. Singularity’s installation has been fully automated as well. For KVM, our team managed to install Ubuntu operating system on a virtual machine manually using the node’s shell, and we are looking into ways of automating the installation process.

So far, we have successfully ran Linpack benchmark on Docker, using the motivational study paper’s GitHub repository code that was tweaked to remove old and no longer working references. This version of Linpack is also available online, which we were able to obtain and will use moving forward, because of poor compatibility of the study paper’s code. Until recently, for KVM and Singularity, we have used a different version of Linpack that operates with a small array size and produces significantly slower speeds of computation. For compatibility and comparability reasons, we have chosen to discard this version in our upcoming testing.

In our original plan, we have planned on testing the RandomAccess benchmark, but it proved a challenge due to lack of easily accessible code that we would be able to run. The study paper’s code for this benchmark lacks some key executables to launch, and online search for a similar version has beared no results so far. Due to these reasons, we have decided to move away from RandomAccess and instead start working on the Stream benchmark installation and execution.

Our initial approach for the execution of deployment was trying to automate the virtualization technology installation and benchmark execution on the profile. The first step should have been manual installation and testing, gradually moving onto automation - the team lost a lot of time trying to automate the process from the get-go due to unfamiliarity of the deployment and installation infrastructures. Nevertheless, the team can complete the tasks with manual input at this point. Therefore, the team believes that current state of the project is on par with expectations set out in our initial project description, with a change of RandomAccess benchmark being replaced with Stream benchmark.

At the current stage of the project, we are unable to identify any trends or draw conclusions from the collected data because of low sample size. We aim to have enough data to come to clear conclusions of the comparisons between the three virtualization technologies.

**Projected Milestone**

The team is confident that the requirements and tasks will be met by the project deadline. By the next deliverable, the project should contain:

* Automation for Docker, KVM and Singularity installation
* Execution of Linpack and Stream benchmarks in Docker, KVM and Singularity
* Data collection for Linpack and Stream benchmarks in Docker, KVM, and Singularity
* Comparison of benchmark findings between Docker, KVM and Singularity
* Relating the collected data to the findings of the motivation paper study
* Instructions for replication of our experiments with fully functioning profiles

Despite our recent addition of Stream benchmark to our project plan, we expect to be able to launch and test the benchmark by the final deliverable. In addition to our goals listed above, we will aim to ensure that our findings are correct based on our methods of execution. Furthermore, the team will attempt to automate the benchmark installation and execution, if the above tasks are accomplished and time permits.