Lipika Chandrashekar, Laura Gomez, Kyree Fuller & Austin Noska
CSC496
May 1st 2020

Deliverable #2 Technical Paper

Introduction:

The goal of this project is to configure, build & run a docker container capable of running four different types of benchmark software with the purpose of testing different performance capabilities of a docker container. The selected benchmarks include the Linpack for testing high performance computing, STREAM for testing memory bandwidth, Nuttcp for testing network bandwidth, and SysBench for testing memory, filesystem and the networking subsystem under heavy load. Using an IBM research report from 2014 as our basis we set out to update their experiments using the tools and methods learned in class. The following document will describe our process, conflicts, solutions, results and goals for deliverable #3.

Process:

We approached this project from several different ways and ran into a myriad of problems during our initial trials. Docker desktop seemed like a reasonable approach at first however using Dockerfiles and attempting to automate the process proved very difficult. We have included some failed Dockerfiles in the repo to show attempts at partial automation of the tests. We realized that using Alpine Linux as our base of operations and installing Docker on that distro to run the containers and taking more of a manual approach was more comfortable for all of us and allowed us to tinker and experiment with all that we have learned. For the Stream and Linpack benchmarks as they are an evaluation of individual hardware they can be run locally on a VM through virtual box while Nuttcp and the MySQL test require multiple nodes, deployment through Cloudlab is the preferred method of approach.

Nuttcp:

The objective of this benchmark was to measure the network bandwidth between two communicating containers connected over a network using docker and CloudLab with the help of the nuttcp tool.

For this we created a cloudlab experiment with two nodes with an UBUNTU 18.04 image. Using the ubuntu terminals we were able to ssh into the nodes separately. We then installed docker, created a docker swarm and added the nodes onto it.

```
LipikaC@clnode170:~/mydockerbuild$ sudo docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
jy5yll2w5h6unrdws338i47oo     clnode054           Ready               Active                                  19.03.8
ea3123ydagqecemu25cfyeo7h     clnode071           Ready               Active                                  19.03.8
ff47q7lp8mcv1wmtv80kuvoi4     clnode072           Ready               Active                                  19.03.8
kaohoz4gz6bj8xj0gfj1ysez0 *   clnode170           Ready               Active              Leader              19.03.8
auxfxwz5740zho75z8nnkpm55     clnode172           Ready               Active                                  19.03.8
```

Our next step was to create a directory and a dockerfile that had the recipe for the creation of an image (testing for node 1 and testing1 for node 2) on the nodes. The dockerfile's content makes sure all packages and tools like nuttcp are installed in the image and container.

```
LipikaC@clnode172:~$ mkdir mydockerbuild
LipikaC@clnode172:~$ cd mydockerbuild/
LipikaC@clnode172:~/mydockerbuild$ sudo vim Dockerfile
LipikaC@clnode172:~/mydockerbuild$ ll
total 12
drwxr-xr-x 2 LipikaC cloud-edu-PG0 4096 May  1 13:31 ./
drwxr-xr-x 6 LipikaC cloud-edu-PG0 4096 May  1 13:31 ../
-rw-r--r-- 1 root    root           117 May  1 13:31 Dockerfile
LipikaC@clnode172:~/mydockerbuild$ sudo docker build --tag="testing:Dockerfile" .
```

```
LipikaC@dnode172: ~/mydockerbuild
FROM ubuntu:18.04

RUN apt update
RUN apt install -y \
    apt-utils \
    net-tools \
    iputils-ping \
    nuttcp
~
```

We then created an overlay network called Proj to help nodes on the swarm communicate with each other.

```
LipikaC@dnode170: ~/mydockerbuild
LipikaC@clnode170:~/mydockerbuild$ sudo docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
f00bbe785a25        bridge              bridge              local
f9d416e5e793        docker_gwbridge     bridge              local
b2fdee41dcff        host                host                local
qq54f5j57mqg        ingress             overlay             swarm
2d9d5a35dc43        none                null                local
izxnx5bp4ln9        proj                overlay             swarm
LipikaC@clnode170:~/mydockerbuild$
```

Two containers were created for the two nodes and then connected to the overlay network proj using the following commands:

```
LipikaC@clnode170:~/mydockerbuild$ sudo docker run -it --name cont1 --network proj testing:Dockerfile
```

```
LipikaC@clnode172:~$ sudo docker run -it --name cont2 --network proj testing1:Dockerfile
```

The last part of this benchmark involves conducting the test using nuttcp to measure network bandwidth. One node serves as the server and the other as a receiver.

To perform the test:

This command was entered on the transmitting containers terminal

```
root@868c75392303:/# nuttcp -S
```

And the following was entered on the receiving containers terminal.

```
root@613b3a0df468:/# nuttcp 10.0.2.30
```

This is the final result of the nuttcp benchmark:

```
1085.7786 MB /  10.02 sec =  908.5625 Mbps 3 %TX 19 %RX 0 retrans 0.35 msRTT
```

SysBench with MySQL:

To be able to measure the performance of MySQL, first we had to download the MySQL software. This required creating a root and password to be able to access it later on when testing. Then we decided to approach the project by creating a VM with Alpine. In this VM we downloaded Ubuntu and installed Docker to be able to create the container and images. Inside the container we installed a few tools needed such as:

- The Sysbench tool which is what we will use to test against MySQL.
- Nano editor to be able to add and edit files in the container.

We have not yet been able to test MySQL due to errors that we have encountered when adding files to the container. However this is something that will be done before the final part of the project. Once we have the testing done, we will upload an instance of the MySQL image to openstack using a cloud lab experiment.

Linpack:

The Linpack benchmark seeks to measure high performance computing capabilities by having a cpu solve a dense system of linear equations. The program that runs this series of tests is supplied by intel and they provide the necessary binaries to run the benchmark. We were able to get these binaries unpackaged inside of a docker container by using a dockerfile with docker toolbox which can be seen in this repository within the linpack directory. From this point we are stuck though as we are unable to

properly compile or run the program and therefore can not produce results from this benchmark at this time. The goal for deliverable three is to figure out why we are unable to compile and run or switch to a different form of the linpack test other than the one provided by intel as there are more examples with other hpc benchmarks that may prove to give us a better rate of success.

Stream:

For this benchmark, the idea was to test memory bandwidth on a given system. We first created an experiment with cloudlab and tried to build a container on the head node so that we could use the resources of cloudlab servers which would allow us to run the max parameters for our test. As listed above, we got away from using cloudlab to test this benchmark for the time being so that we could figure out how to test it by building a container with our own hardware. We were successful with building the container and the Dockerfile successfully copied all of the source C files for the benchmark that we edited from the previous GitHub Repository, however we were not able to successfully compile the benchmark or figure out how to debug all of the c files so that they compile and run correctly to give us the output that we want. We believe that this may be due to outdated code that was being used on outdated systems. For deliverable three, our plan is to have the Stream benchmark finished and the code for the actual test updated so that it runs on modern systems without issues. After we are able to successfully run our tests in a container that was created locally, we will try to deploy the same test on a cloudlab platform so that we can use max resources.

Deliverable #3 Goals and Conclusion:

We have gone through a lot of trials and tribulations attempting to get all of these benchmarks to successfully run and although we have only gotten one benchmark to run so far we are confident in our ability to get the rest of them running in time for the deliverable 3 deadline. Taking all that we have learned to get the nuttcp benchmark running and what we hope to learn from other teams deliverables we will be able to apply this new knowledge and learn from all of our mistakes so far to be able to complete this project and see all four benchmarks running successfully.