Docker Platform : Report for Cloud Computing Coursework

Haoran Duan[1]

170733151, MSc Data Science, School of Computing

Abstract. Use Docker Platform to build service, monitor them and analyze the benchmark results.

1 Task 1 Pull the webapplication image and test

In task 1, I use both python SDK and commend line in Ubuntu 16.04 to finish. After 'pull' the image, it can be runned directly. I think the differences between commend line and programming languages are not too big. It is a bit like that programming languages follow the way of commend line, even the name of some variables are same. For the using of docker, I think the best way is to use the commend line, because it is better to learn the properties of system and set what we want.

2 Task 2 Multi-Service application in Docker

In this task, I build a single node with multi-service system in docker(Fig.1).

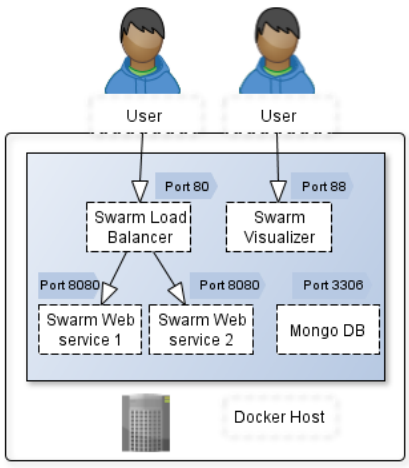


Fig. 1. UML Diagram about Docker Swarm

Follow the UML configuration, I use Swarm web service 1 & 2(2 replicates) to provide the load balancer web service. I use the Swarm Visualizer to visualize the structure of my service(Fig.2). Also I use Mongo Image and the port 3306 to visit the service of mongoDB service.

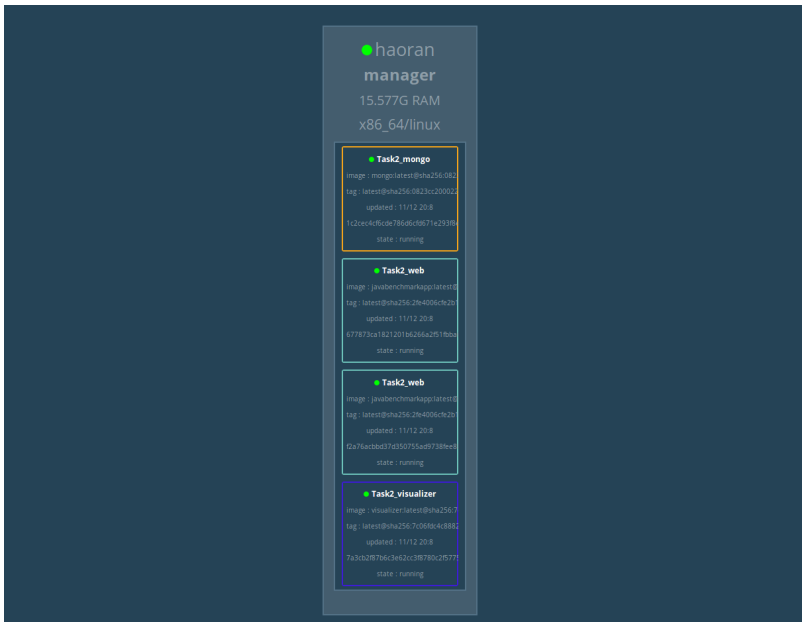


Fig. 2. Visualizer

3 Task 3 Load Generator

In this task, we need a program to send a request and get the content in different time, which can create load on the web application by calling its URL multiple times. For randomly and rationally choosing, I use the Normal distribution and Poisson distribution to generate the request intervals which is the waiting and sleep time of the program.As it is shown(Fig.3) that there are four parameters to choose the distribution.

At the beginning, some of the values from distribution are negative, so I use the absolute value, and I will compare the different between using or not using the absolute value.

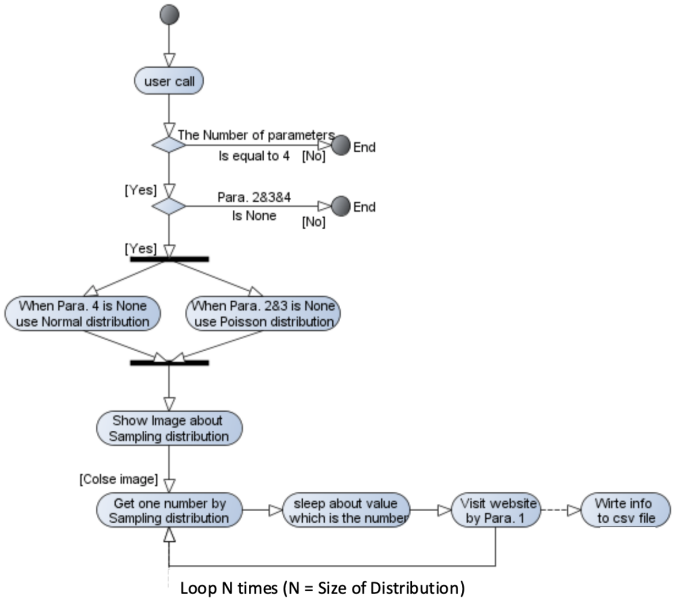


Fig. 3. UML of Load Generator

4 Docker monitoring tool

It is very easy to use the cadvisor to monitor the usage of the hard wares like CPU and Disk. The google image of cadvisor can be easily used by 'docker run',and it can be show some information like Fig.4.

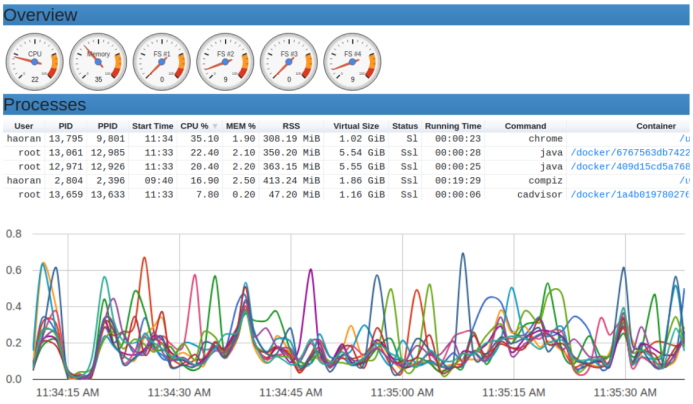


Fig. 4. Cadvisor

5 Benchmark

First of all I use MongoDB to save the data from the Cadvisor Api, the process is shown in Fig.5. After that,I use python to analyze the benchmark results. The 60s,which is for each time of getting the information from API, is last minute information.

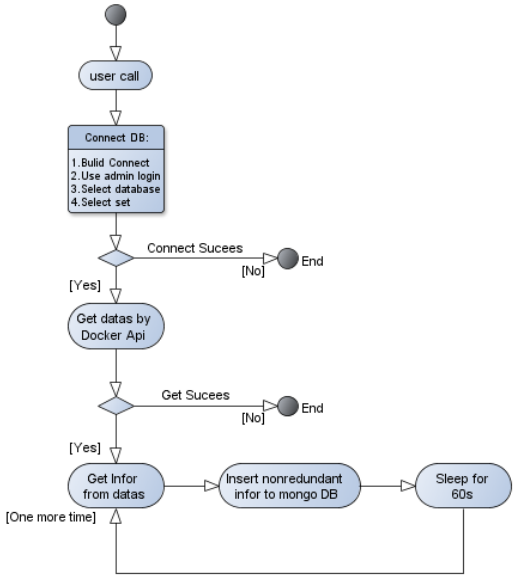


Fig. 5. Data Saving Process

And the final process of my system can be (Fig.6)

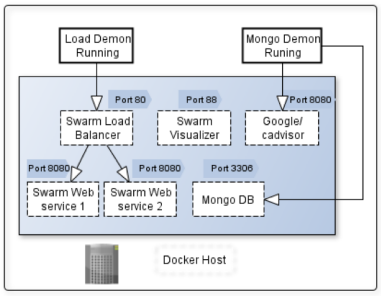


Fig. 6. Data Saving Process

6 Analysis

First of all, I consider the values from the two distributions. Considering to keep most of value be positive, I choose three group of two distribution(Table.1). And 100 values were set as the size of each distributions.

Table 1. Three Groups of Parameter of two distributions

Group No.	Normal distribution		Poisson distribution
	μ	σ	λ
1	5	1.5	5
2	3	1	3
3	1.5	0.5	1.5

We can use the Fig.7 to check the distribution of each group,which should follow our setting.

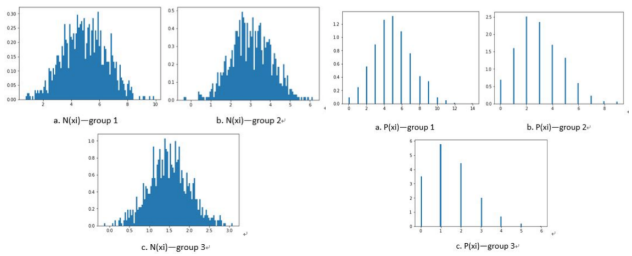


Fig. 7. Distributions of Number

6.1 How Different Distributions of Time Influence

There are two groups as six different situations,the time from the web shown at the left of the Fig.8, Normal Distributions looks like from high to be low and be high again. And it is inverse in Poisson Distributions.I think it is because most of the values in Poisson distribution is a real value which will can get the '0', and '0' is a key that it can influences the elapsed time. Also, if we check the example of web time of the sixth group(Right in Fig.8), the more outliers will leads the more bias of results.

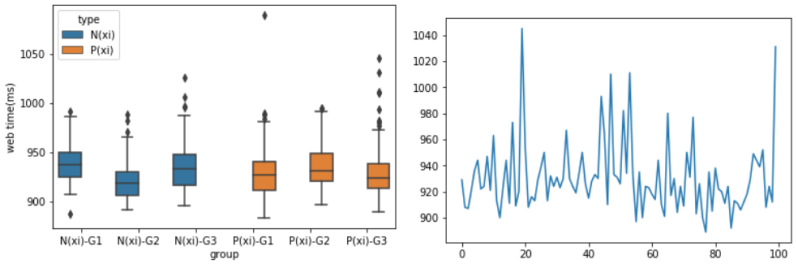


Fig. 8. Web Time

And I think the reason, why the value for Normal Distribution is decrease first and then increase, is that there are some caching mechanism in our system. So after a small time interval, the utilization will increase.

6.2 CPU

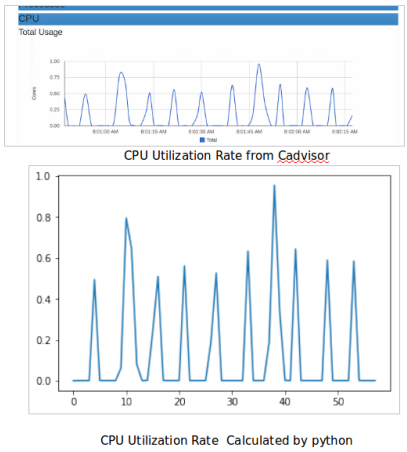
6.2.1 Calculate First of all in CPU benchmark results, I use the data I got to calculate the CPU utilization rate, and compare to the Cadvisor to check the results. The method is:

- For each data of CPU, I use this time step minus last time step(X1).
- Changing the per unit of time to nano-seconds, then also use this time step minus last time step(X2).
- Use X1 / X2

And they can be wrote as : For each time step CPU data

$$\frac{C_{this\_time} - C_{last\_time}}{C_{this\_nanotime} - C_{last\_nanotime}} \tag{1}$$

We can verify it use the Fig below(Bottom is mine),as we can see the results are same. So we can say that our benchmark is appropriate.



6.2.2 Analysis As it is shown that(Fig.9):

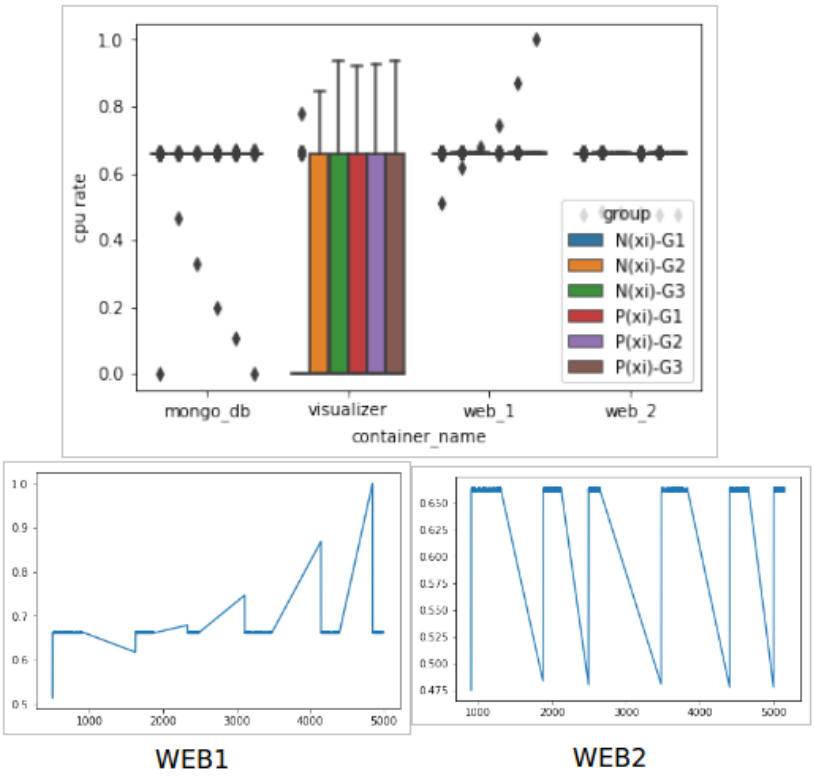


Fig. 9. CPU Utilization in six Situations(Table.1)

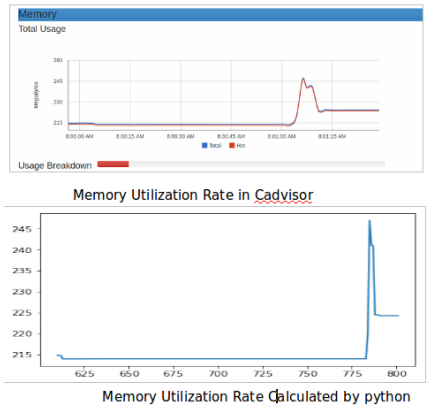
From the total CPU utilization(TOP Fig.9),the minimum values of MongoDB are decreasing , the maximum values of WEB1 are increasing. But most of the values of each container are almost same. And if we check two bottom graphs(Fig.9) which is changing with time, I delete the waiting time and concat all the six group (From left to right:NG1, NG2, NG3, PG1, PG2, PG3),the six horizontal lines(six group) in each WEB application are keep same, other lines represent the interval time. So we can infer that the fluctuate is because the WEB application and docker need to do some adjust before next group of sampling.

6.3 Memory

6.3.1 Calculate First of all in Memory benchmark results, I use the data I got to calculate the Memory utilization rate, and compare to the Cadvisor to check the results. The method is:

$$\frac{Data\_Memory}{1024} \tag{2}$$

We can verify it use the Fig below(Bottom is mine),as we can see the results are same. So we can say that our benchmark is appropriate.



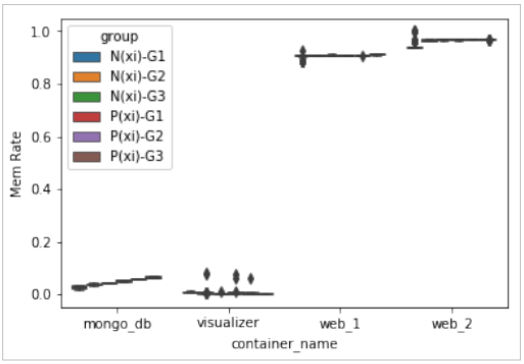


Fig. 10. memory rate in six groups

6.3.2 Analysis There is no obvious information from Total Memory utilization(Fig.10).But it still shows the memory utilization in Web Application is much more than others, and WEB2 is higher than WEB1.

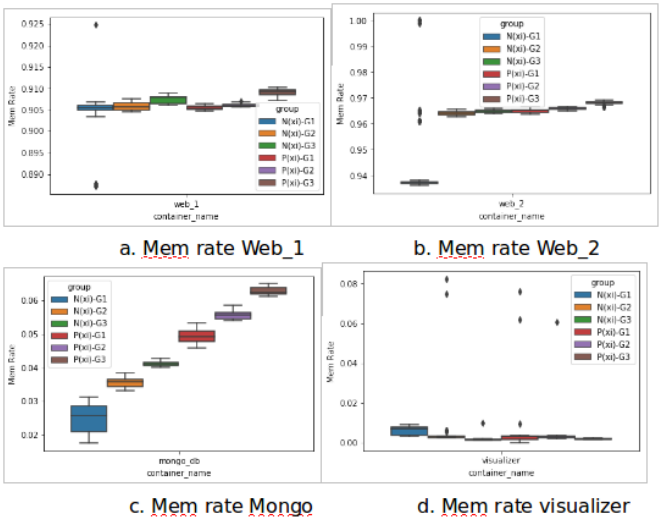


Fig. 11. memory rate in six groups

If we check the detail of each container:

– The memory utilization in Visualizer are very small, even close to zero, because when we call some request and wait some time, we didn't use Visualizer.

– About the MongoDB, because the more and more data were stored(1-100), the memory utilization become bigger. This graph(c.Fig.11) is relate to time, not group, the different groups are used one by one.

– The memory utilization in WEB2 is more than WEB1, so I think WEB2 play a more important role. Also, in both containers, the memory utilization will decrease with the less waiting time and smaller mean.

6.4 DiskI/O

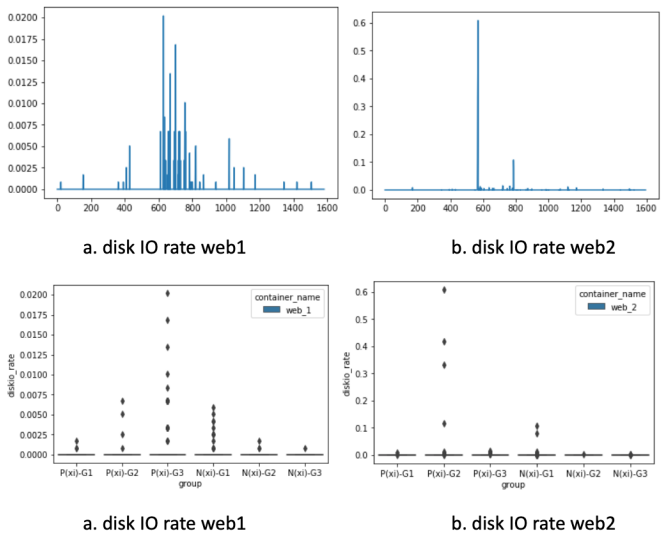


Fig. 12. Disk IO rate for web1web2

First,in my experiment, the Disk IO is not always exit, this is what I am confused,in VM from coursework,it is sometimes not empty, but in local Ubuntu(At least 5 classmates) it is empty. I think for WEB applications, they truly should not have many Disk IO, but have many Net IO. But I still use the data I got to analyze(Fig.12). In WEB2, the value of WEB2 Disk IO increase suddenly sometimes, because there are many noise and

outliers. So we can infer that the data in WEB1 can be more confident and reliable. And different distributions have different influence, For example, the value of Disk IO will increase with smaller mean(Table.1) of Poisson and decrease with smaller mean(Table.1) of Normal distribution. Finally,I also find that beacause there are more and more data, the memory utilization of MongoDB become more and more.

6.5 Network

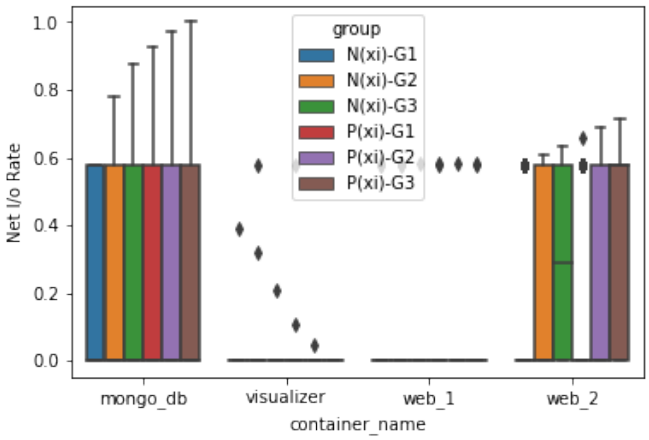


Fig. 13. Net IO for All

The Fig.13 shows all general situation of Net IO. The WEB2 Net IO is bigger than WEB also because of more ouliers and noise. We can see that the Net IO in visualizer become smaller,because when all the things started, the visualizer will visualize the structure for them,then if there is no any other change of structure,the visualizer will sleep and do not have activity.

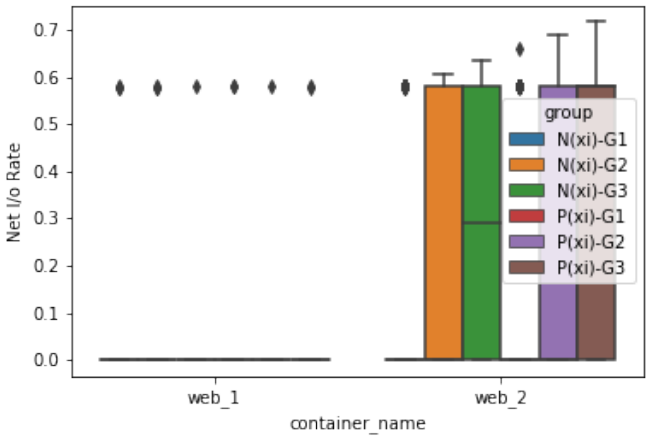


Fig. 14. Net IO for All

If I plot the graph only for two WEB applications(Fig.14). As it shows that, the bigger mean(Table.1) for both distributions leads to the more centralized values of Net IO,so with the decrease of mean(Table.1), the values of Net IO become more disperse. And I find in WEB2 application(Fig.15), as the time changing, the max values of Net IO for WEB2 is becoming bigger. Also these sudden max values appear at the beginning of connecting.

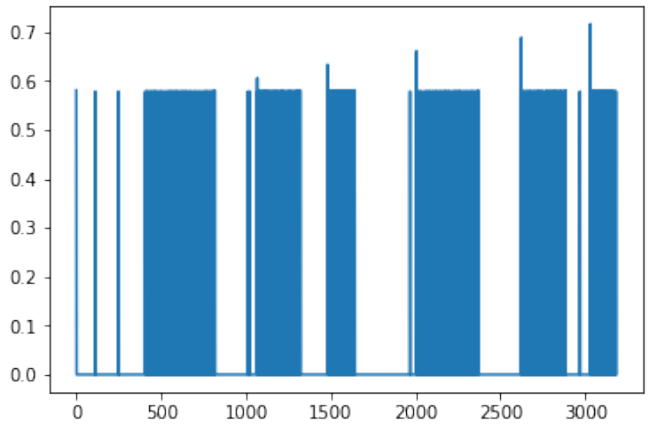


Fig. 15. Net IO for All

7 Cloud and LocalMachine

I think that the cloud platform is very convenient, it is stable and we can keep it always on line. But the local machine is cheaper and if something broken, it is easy to be fixed.