SparkFHE Resource Logging Design

Suwen Gu

May 2019

1 Experiment Tools

- Cloud lab m510 machine: Xeon 16 Cores CPU and 64G RAM
- SparkFHE: standalone version
- Docker Container
- Kubernetes

2 Experiment Description

The goal of this experiment is log resource usage, including CPU and Memory usage in percentage, for different spark jobs running in Docker containers and Kubernetes pods. Due to the sample tests provided by SparkFHE library are relatively light workload. To simulated heavier spark jobs, I ran 4, 8 and 12 jobs in parallel at at time. The jobs I tested including basic OP, dot product and a mix of both. The followings are the summary of the jobs tested in the experiment (All the tests are run in one container):

- 4 Basic OPs
- 4 Dot Product Operations
- Mix of 4 jobs, 2 basic OPs and 2 Dot Product Operations
- 8 Basic Ops
- 8 Dot Product Operations

- Mix of 8 job, 4 basic OPs and 4 Dot Product Operations
- 12 Basic Ops
- 12 Dot Product Operations
- Mix of 12 job, 6 basic OPs and 6 Dot Product Operations

3 Resource Logging

3.1 Introduction

I primarily used two commands to log the resource usage when running test jobs in Docker containers and Kubernetes.

For Docker container, the command **"Docker stats [container name]"** is used. This command returns the memory and CPU usage in percentage for a user specified-container.

For Kubernetes, the command **"kubectl get pods [pod name]"** could be used to return the resource usage of a running job. However, the caveat is that this command could return incomplete dataif the run time of a job is relatively short. This is due to the command uses an aggregator called **metrics server**, which takes time to initialize while calling the command.

To work around this problem, due to the nature of Kubernetes who use Docker image to create containers. The command **"Docker stats"** is also employed to log resource usage when jobs are run in Kubernetes pods.

3.2 Experiment Procedure

The script, mySParkSubmitLocal.bash, provides five examples, the first three of which are used to generate data, and the last two of which are basic OP and dot product tests. The data should be generated before calling the last two examples. Thus, the procedure of this experiment was conducted in the following order:

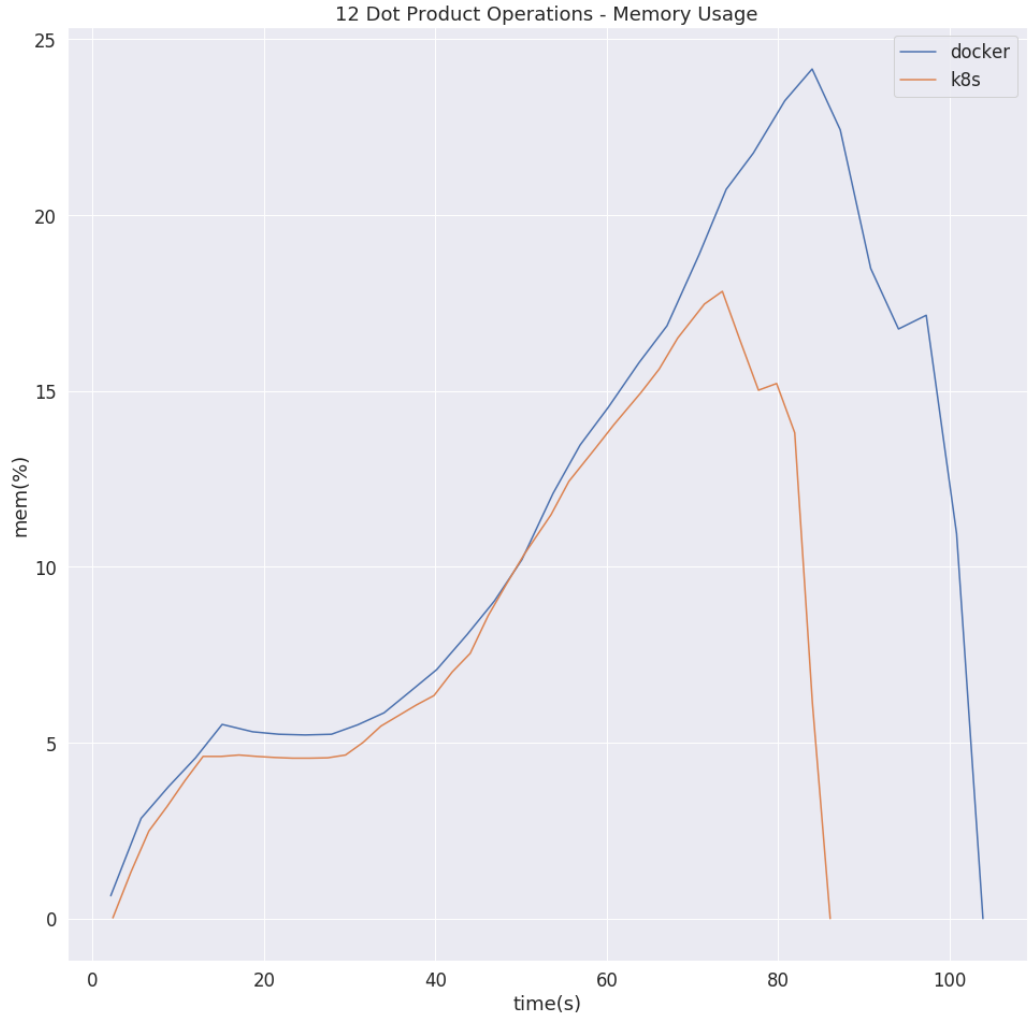
1. Generate data using:
    - sparkfhe_basic_examples
    - sparkfhe_keygen

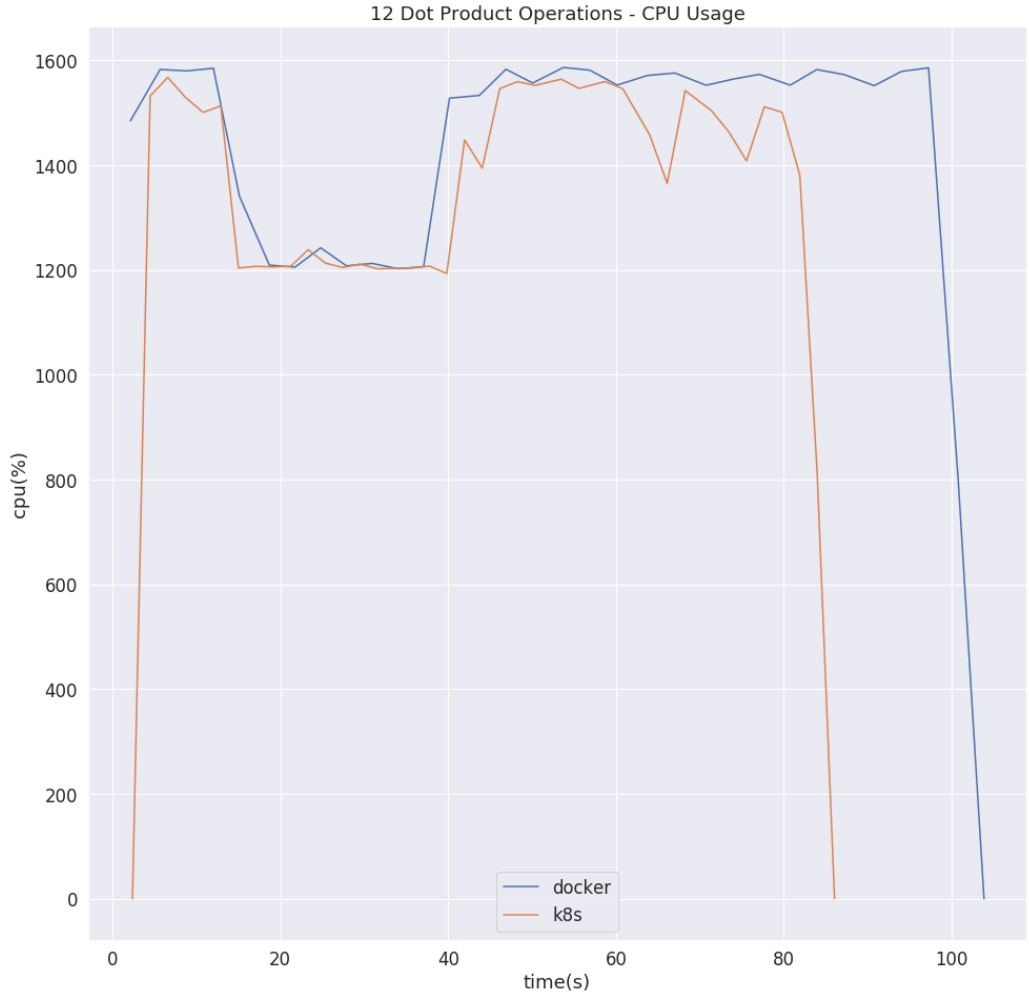
- sparkfhe_encryption_decryption

2. Run N jobs of:
    - Basic OP
    - Dot product
    - A mix of both
3. Run data logging script

4 Remarks

I have examined the data by visualizing them in line plots. One thing I noticed is that the resource usage of Kubernetes containers is less than the one of Docker containers. Also, the jobs ran in Kubenetes pods finished faster than the ones running in Docker containers. For example, as the graphs show below, for a job of 12 dot product operations, Kubernetes used around 17% of memory, and the job took around 80 seconds to finish. Despite nearly identical CPU consumption (slightly less than 1600% of 16 cores), for Docker containers, it used around 25% of memory, and the job took around 100 seconds to complete. The result is counter-intuitive since Kubernetes added another level of abstraction on top of Docker containers. For the next two days, I plan to use a third-party monitoring tool, such as Prometheus and cAdvisor, to log the resource usage when jobs running in Kubernete pods. I believe using these tools can give us more accurate resource usage metrics.


12 Dot Product Operations - Memory Usage
docker
k8s
mem(%)
time(s)
0
5
10
15
20
25
0
20
40
60
80
100


12 Dot Product Operations - CPU Usage
cpu(%)
time(s)
docker
k8s
0
200
400
600
800
1000
1200
1400
1600
0
20
40
60
80
100