# Assignments for Distributed Computing CS G557: Scalable Service Implementation

SANTONU SARKAR (PI)

February 13, 2024

## Instructions

The latest documentation for docker container is available here: `https://www.docker.com/` We advise that you spend time in learning docker usage, download the docker engine and create your own container where you can run your own application. Common Instructions:

1. Install Docker in your computer and learn the basics
2. Try running a simple application as a service, i.e. it should be awake all the time, waiting for a user-input. For instance, try implementing a simple calculator function that takes two numbers, adds them and returns it back. Then it waits for an user input.
3. Run such an application inside a container. Try calling the service (the addition service) from outside, from another "master" application. This application should call the containerized addition application by supplying two numbers, get the result and display/print it. This "master" application can be a simple terminal application or an web-based application, or a containerized application, it's your choice.

## 1 Assignment: Compute $\pi(x)$ in a load-balanced manner

The Sieve of Eratosthenes(ca 240 BC) is an iterative and efficient way to find small primes less than 10,000,000. This technique helps in computing $\pi(x)$ which computes the number of primes not exceeding $x$. You need to implement $pi(x)$ service which will return for example $\pi(29,996,224,275,833) = 1,000,000,000,000$ (primes less than or equal to 29,996,224,275,833). Try this: `https://t5k.org/nthprime/`.

In this assignment, you should use a pre-computed $\pi$ function supplied as a text file. Your application should read this text file to provide the answer $\pi(x)$, when $x$ is supplied as an input. You are NOT supposed to use hash table to store the content of the text file. The text is very big, nearly 700MB. You have to perform a sequential search. Put this application (and the text file) in a container, as a service, and call the service from a master application from outside. The master application from outside, calls this service by supplying $x$ and the containerized application provides the answer. In the actual assignment, you have to make a load-balanced version of this solution.

### 1.1 Actual Assignment

Implement a load balanced version of $\pi(x)$ in the following manner. Divide the text file into chunks of $2, 4, 8, 16$, and 32. Clone your application - create $2, 4, 8, 16$ and 32 instances of your docker application. Each instance will work on one chunk of the text file. Create a master application that takes $x$. Depending on the value of $x$, the master redirects the $\pi(x)$ computation request to the appropriate instance (that has the right chunk), and gets the result.

#### 1.1.1 Tasks

Create another text file, that contains a set of 1000 randomly generated integers. These integers should be from small numbers to very large numbers(such as `sys.maxsize` in Python3 returns $9,223,372,036,854,775,807$). If the random numbers are widely distributed, you will get better results. So create this text file judiciously.

For each cluster size $(1, 2, 4, 8, 16, 32)$ do the following:
1. The master application should read these numbers one by one from the text file, send the request to the appropriate instance, and store the result in an output text file.
2. Measure the time to compute individual request (average) and the total time to complete all the requests.

Once you collect the data, plot these results, x axis - cluster sizes $1, 2, 4, 8, 16, 32$ Y axis- time to complete (avg and total)

Using Python API dump this result in a tabular format.
If plotting is difficult, create an equivalent table, though correct plotting will fetch full-marks.

#### 1.1.2 Cluster Creation

It is upto you how you want to create clusters. Simplest would be to create a cluster comprising a set of containers manually. Then run the experiment (step 1 and 2). Next change the cluster size and repeat the experiment and collect data. Do it for cluster sizes $1, 2, 4, 8$ and so on. it is expected that you automate this process using a script- start the experiment where you supply the cluster size as input- then complete. Demonstrate your automation by running the script repeatedly with different cluster sizes.

### 1.2 Evaluation

1. We will check the result in the lab.
2. Upload the result in Google classroom before due time. Name the zip file as yourGroupID.zip and upload it in Google Assignment. The Google Assignment will have a due date and time. This will ensure that you finish your assignment ON time. If you can't upload withtin the time, I will assume that you have not been able to complete the assignment.

Total marks= 10. We verify the correctness of the solution by looking at the output value, and actually checking the result in `https://t5k.org/nthprime/`. – 2 marks
Plot - 1.5*2=3 marks
Demonstration of automation - 2 marks.
Explanation of the plot - one para - 3 marks. If the explanation is not convincing, you won't get marks.