

CS322:Big Data

Final Class Project Report

**Project (FPL Analytics / YACS coding): YACS**  **Date: 06-12-20**

|  |  |  |  |
| --- | --- | --- | --- |
| SNo | Name | SRN | Class/Section |
| 1 | Neeli Krishna Dheeraj | PES1201800182 | C |
| 2 | Alluri Nayan Varma | PES1201800208 | J |
| 3 | Akhil Chaduvula | PES1201800282 | D |
| 4 | Thota Saran Sai | PES1201800845 | J |

## Introduction

This project is also known as YACS(Yet another Centralised Scheduler). In this we are supposed to prepare a simulation of the master and workers Architecture. The whole project is done in order to show how multiple mapper and reducer tasks of various jobs are performed in real life. This setup is used because it is very tedious for only one machine to perform all the tasks, hence multiple machines(workers) perform the tasks whose process and order of execution is managed by one machine (Master).

## Related work

To get an idea of the various components of our project, we have referred to the courses Bid Data, Computer Networks, Operating systems to learn a few concepts such as Master worker Architecture to store data in HDFS(from Big Data Course), working of TCP connections(From Computer Networks), and the working of Locks, Threads and Round Robin algorithms(From Operating systems). Apart from these, we also referred to a few websites to familiarize ourselves how TCP socket connections are made to the localhost.

1. [Socket Programming in Python - GeeksforGeeks](https://www.geeksforgeeks.org/socket-programming-python/)
2. [Program for Round Robin scheduling | Set 1 - GeeksforGeeks](https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/)
3. [Python - Multithreaded Programming - Tutorialspoint](https://www.tutorialspoint.com/python/python_multithreading.htm)
4. [Hadoop Distributed File System | Apache Hadoop HDFS Architecture | Edureka](https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/#:~:text=Apache%20Hadoop%20HDFS%20Architecture%20follows,of%20machines%20that%20support%20Java.)
5. [Reading and Writing to text files in Python - GeeksforGeeks](https://www.geeksforgeeks.org/reading-writing-text-files-python/#:~:text=Reading%20and%20Writing%20to%20text%20files%20in%20Python.,handle%20is%20positioned%20at%20the%20...%20More%20items)

## Design

We were given with two files

1. Config.json :- This files given to mention the number of Workers, their Worker id’s , the port number of each worker, and the number of sockets in each worker. This information was necessary to feed into the master so that the mapper knows which workers and what ports to connect to.
2. Requests.py:- This file is the one that creates the jobs and all the tasks in them and then connects to port 5000 and then waits for the master to connect to it.

Our Design depended mainly upon the creation of two python programs i.e

1. master.py
2. worker.py
3. analysis.py

**Master.py:-**

This file is the most important part of the whole simulation. This file firstly connects to port 5000 to receive data generated from the requests.py file. This data obtained as the text file is then converted into a Json format using json.loads(). 3 Scheduling Algorithms were then introduced.

1. Round Robin Scheduling:- Round robin scheduling is an algorithm mainly used by operating systems and applications that serve multiple clients that request to use resources. It handles all requests in a circular first-in-first-out (FIFO) order and eschews priority so that all processes/applications may be able to use the same resources in the same amount of time and also have the same amount of waiting time each cycle; hence it is also considered as cyclic executive.
2. Least-Loaded:- In This case, the mapper checks for available slots in all the workers. It assigns the task to that worker that has the most number of free slots. If there are no free slots in any of the workers, it then waits for one second, to then again check for free slots and then assign tasks to them.
3. Random:- The mapper selects any worker at random. If there are free slots in them, then the mapper assigns a task to it, else it searches for another worker and then assigns a task to that.

The above scheduling algorithms are used to assign tasks to the workers so that they complete all the jobs. To connect to all the workers from the master file, the master is then connected to three ports given in the config file.

When the master is connected to the workers, the master starts sending tasks according to the scheduling Algorithm (which algorithm to select is given as a user input to the file by the user). The time stamps of sending each task is also recorded. The timestamps of the receiving tasks(to receive, the master is connected to port 5001) is also recorded. A time difference between these is calculated. This time difference between each of the tasks is then recorded in a logs file.

**Worker.py:-**

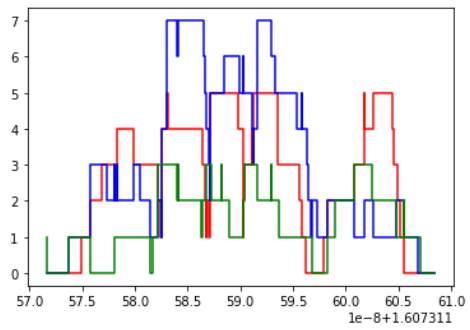
The main task of the worker file is to connect to their respective ports (In this case, since the workers have no access to the config file, the port numbers and worker id’s are given as command line arguments) to listen to the tasks given by the master. The tasks are then assigned to the slots as asked by the master. When the tasks are completed, the master is notified about the completion of the task. For this, all the workers are supposed to connect to the master through the port 5001. For this two functions were created, one is to gather data from the master, and the second to wait for the task to complete and then notify the master about the completion of the task.

## Results

The results that we found were

1. Mean time of jobs = 6.82
2. Median Time of Jobs = 7.153
3. Mean time of Tasks = 3.382
4. Median time of Tasks = 3.329

This is a step Plot of the Workers for RANDOM.txt file



## Problems

The main problems that we faced while solving the Assignment were:-

1. Understanding the problem statement correctly. (Which we misunderstood earlier)
2. Using locks correctly. Using locks unnecessarily lead to wrong results.

## Conclusion

This project helped us get insights over very important concepts of the actual working of the Master workers model. We learnt how locks function, how socket programming works. This model is very efficient to work on huge corpus of data as the load is distributed among machines. But the model might fail if the master stops working.

## EVALUATIONS:

|  |  |  |  |
| --- | --- | --- | --- |
| SNo | Name | SRN | Contribution (Individual) |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |

## (Leave this for the faculty)

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Evaluator | Comments | Score |
|  |  |  |  |

## CHECKLIST:

|  |  |  |
| --- | --- | --- |
| SNo | Item | Status |
| 1. | Source code documented |  |
| 2. | Source code uploaded to GitHub – (access link for the same, to be added in status 🡪) |  |
| 3. | Instructions for building and running the code. Your code must be usable out of the box. |  |