CS322:Big Data

Final Class Project Report

Project (FPL Analytics / YACS coding): ______YACS_________ Date:_03-12-2020_

SNo	Name	SRN	Class/Section
1	Suhas Vasisht	PES1201800212	D
2	Maddi Siddart	PES1201800269	D
3	Chirag Kashyap S	PES1201800400	D
4	Amogh R K	PES1201801309	D

Introduction

Big data deals with workloading of multiple jobs from various applications. Where each job consists of many tasks.These jobs run on clusters of machines and we need a framework to manage the resources to these clusters of machines and this centralised scheduler helps us distribute these resources across the clusters.
It provides us a simulation which consists of a master and the respective workers. A master is a machine which manages other machines in the cluster.
There are various slots provided to each worker and these workers perform tasks in the slots assigned to them. The master is responsible for scheduling these tasks to workers based on scheduling algorithms

Related work

We referred a few links for better understanding about socket programming . And a few others for logging , queuing and threading in python. We understood the concept of YACS from the pdf sent to us.

Design

YACS is a centralised scheduling framework . It consists of one master which manages the other machines which consists of one worker each . We used sockets to communicate between master and the workers. The number of slots is allocated to each worker and these configurations are mentioned in the config.json file.
In the beginning the requests.py sends the details of the jobs which consists of the job id and details of map tasks and reduce tasks with their duration as json objects to the master. In the master there are three threads which control the function of listening to incoming jobs, one for listening to updates from workers and one for scheduling the tasks.

For listening to the incoming jobs we first connect via sockets then decode the json object to get the job id and map tasks and reduce tasks. We use two queues and one dictionary to store the job information . The first queue contains the map tasks and the other contains the reduce tasks . And the job dictionary where the key is job id , and the value is map tasks , reduce tasks, and the number of reduce tasks left to send out.

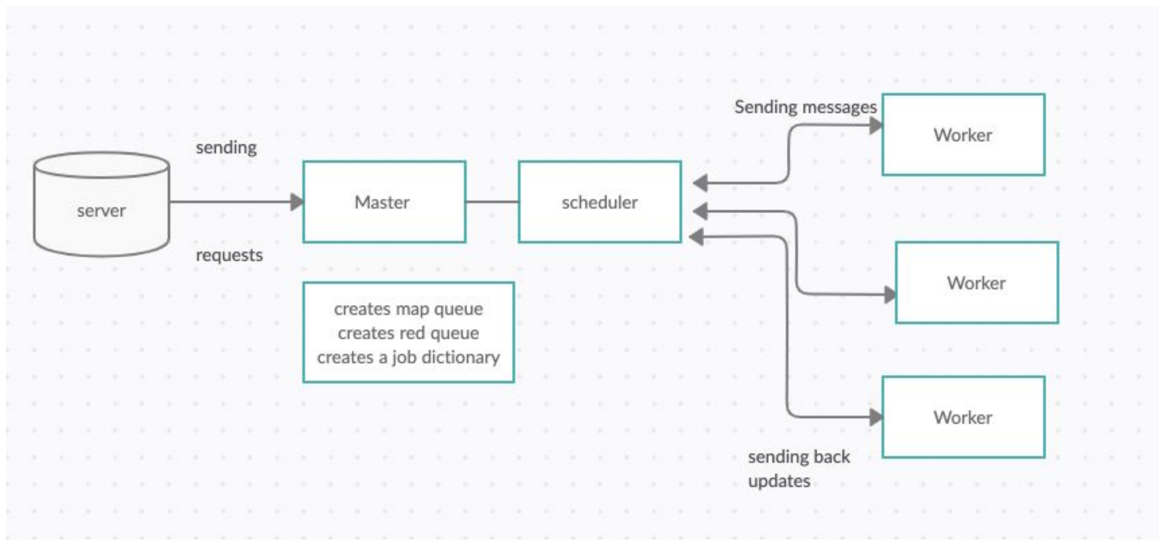
Once the job dictionary is built the master schedules the resources to workers in various techniques such as round robin, least loaded, random algorithms. In the random algorithm it randomly assigns a worker to do the specific tasks if the slots are available .

In least loaded it checks which worker has most slots available and assigns it the tasks.
In round robin it goes through each worker one by one and assigns tasks if slots are available.

The worker on the other hand has two threads one for listening to the master and other for actually doing the task. While listening to the master it receives messages which contains a flag which tells whether the task is the last task and job id , task id and duration of the task assigned to the worker. The message is then added to the pool where the actual processing is done . During processing it counts down till the duration of each task and ticks down if it's done . The response is then sent to the master and the time is then logged .

After receiving the response from the worker the master updates the corresponding slots available in each worker. And once all the tasks in a job is done the job is then removed from the dictionary. And the result is logged for analysis.
This is how the scheduling framework works .

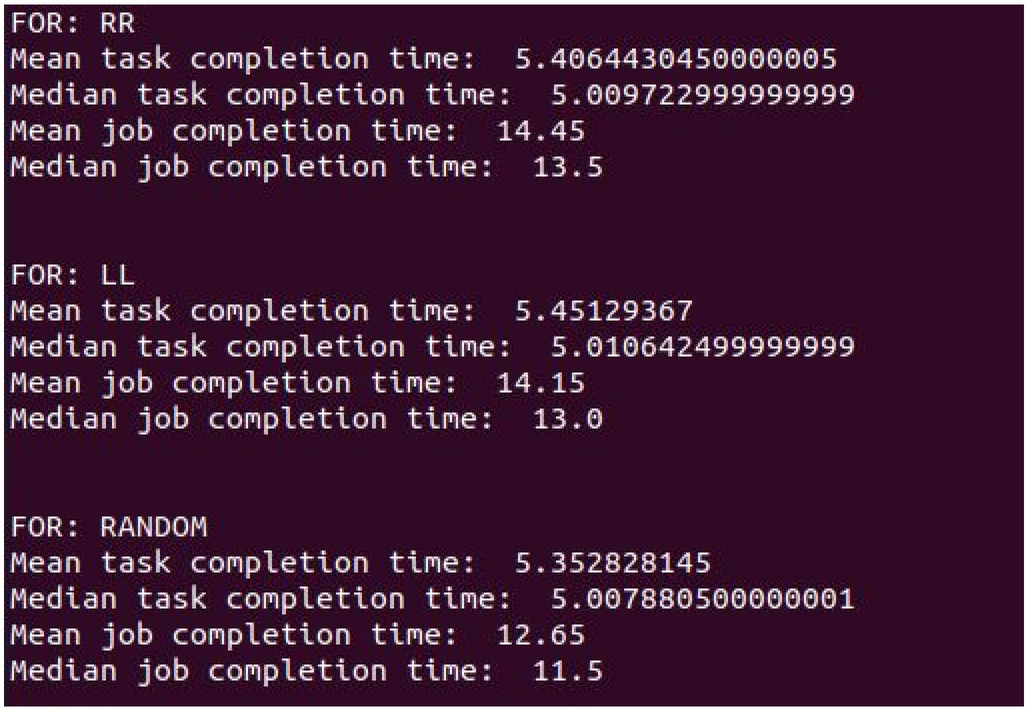
BLOCK DIAGRAM



Results

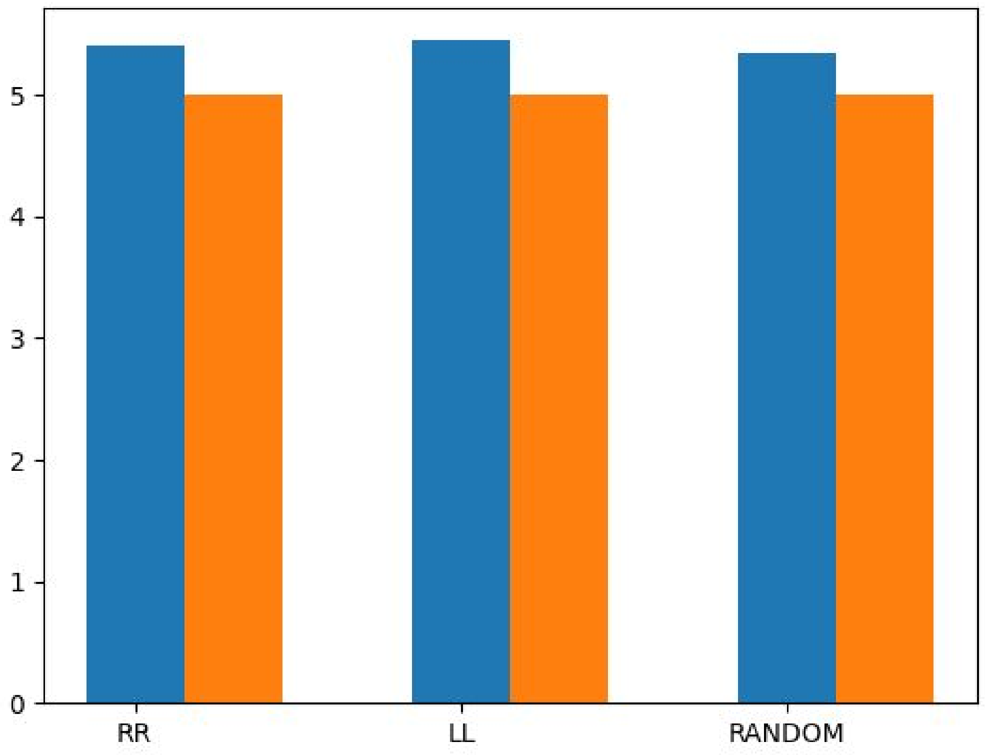
Result 1

20 requests with inter arrival time of 5 secs

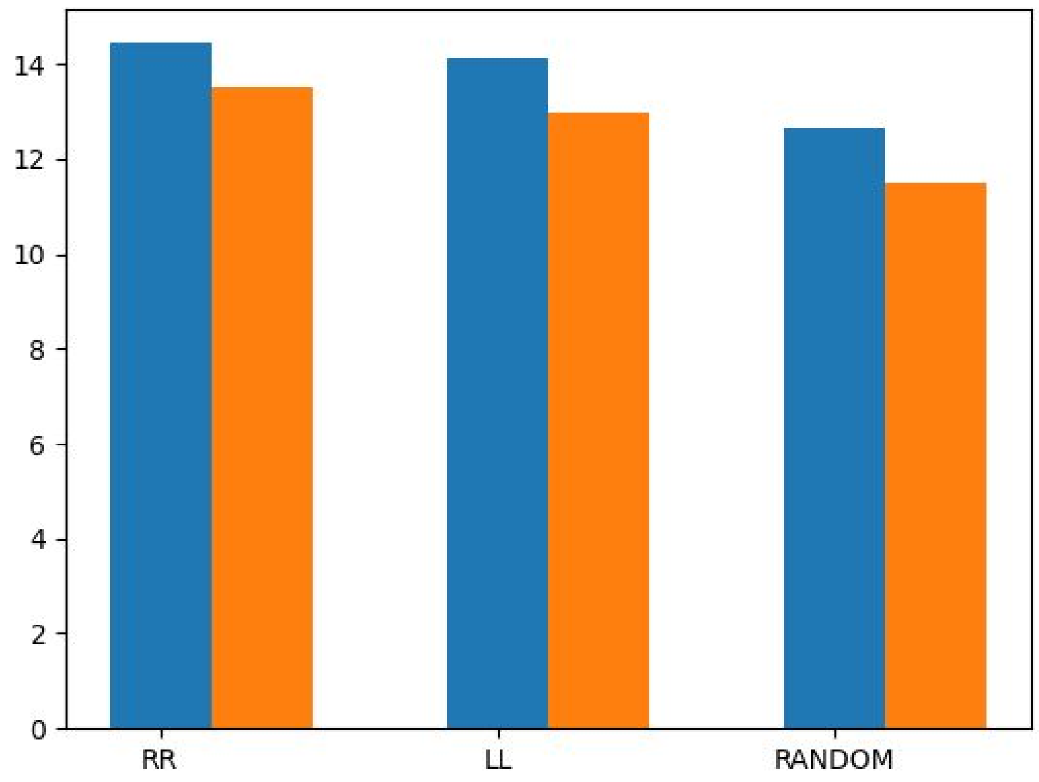


```
FOR: RR
Mean task completion time:  5.4064430450000005
Median task completion time:  5.009722999999999
Mean job completion time:  14.45
Median job completion time:  13.5


FOR: LL
Mean task completion time:  5.45129367
Median task completion time:  5.010642499999999
Mean job completion time:  14.15
Median job completion time:  13.0


FOR: RANDOM
Mean task completion time:  5.352828145
Median task completion time:  5.007880500000001
Mean job completion time:  12.65
Median job completion time:  11.5
```

TASK mean and median :



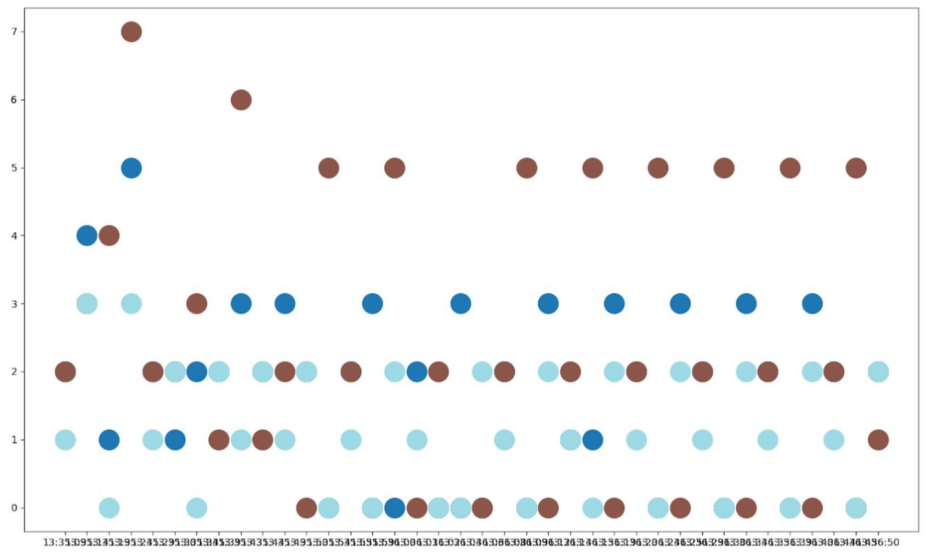
JOB mean and median:



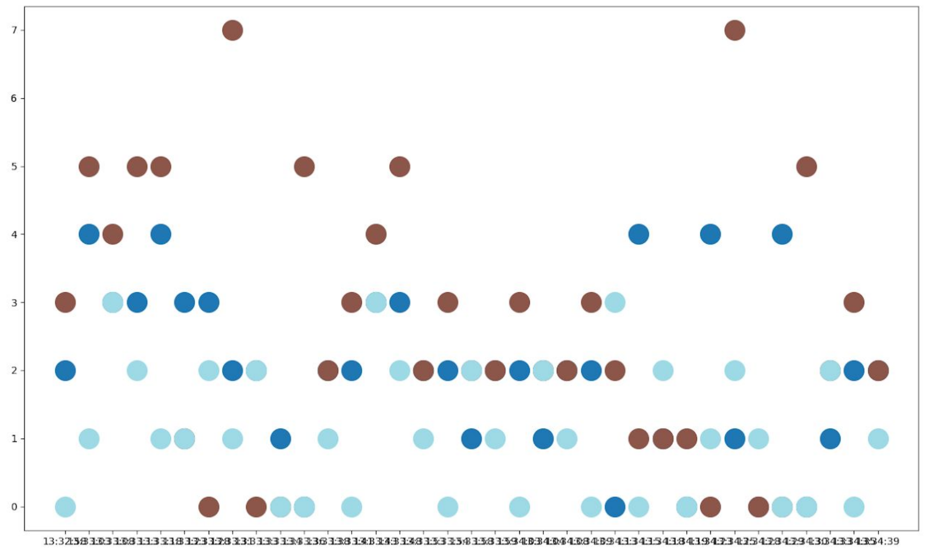
Result 2 :

Number of tasks running on each worker for each algorithm:

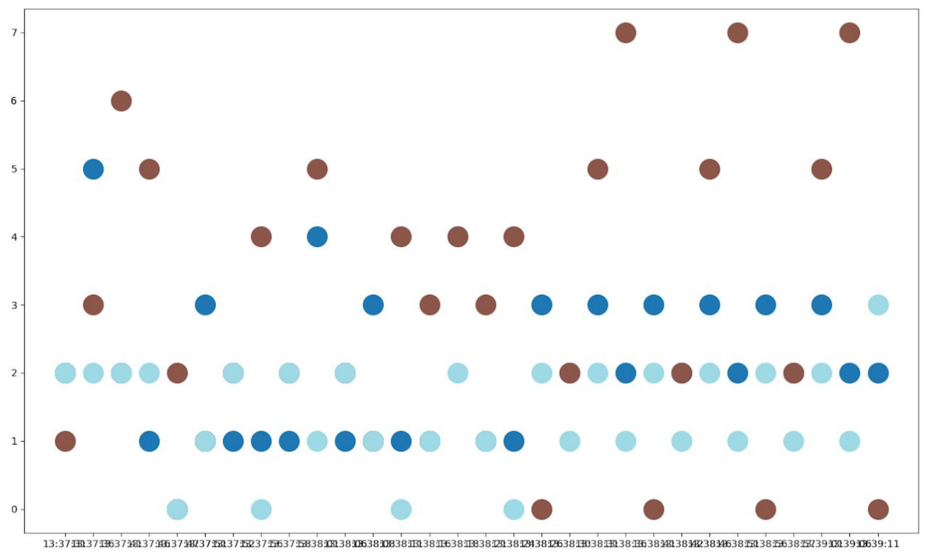
Round Robin



Least Loaded :



Random :



Problems

The main problem we faced was debugging the communication between master and the workers which we solved by sending single messages and logging the results.
Another problem is evaluating the correctness of the algorithm.
The other issues we faced were threading which we solved by referencing a few articles and videos which explained them in depth.

Conclusion

The main topics that we learnt about include socket programming which helped us connect the master and the workers, we learnt basics of threading which helped us perform multiple tasks at a time. We learnt how scheduling algorithms work on distribution systems with a centralized master.

Least loaded scheduler works the best because the worker is optimally chosen based on the number of free slots.

Round robin scheduler works less efficiently as compared to least loaded.

Random schedulers cannot be compared as it works differently every time.

EVALUATIONS:

SNo	Name	SRN	Contribution (Individual)
1	Suhas Vasisht	PES1201800212	master architecture, threading
2	Maddi Siddart	PES1201800269	scheduling , scripting,worker
3	Chirag Kashyap S	PES1201800400	scheduling, socket programming, worker
4	Amogh R K	PES1201801309	analysis , debugging,logging

(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.	