CS322:Big Data

Final Class Project Report

Project (FPL Analytics / YACS coding): ____YACS____ Date: ____01/12/2020____

SNo	Name	SRN	Class/Section
1	Devika S Nair	PES1201800372	J
2	K M Mitravinda	PES1201801872	J
3	Prathima B	PES1201801889	J
4	Ramya C P	PES1201801554	A

Introduction

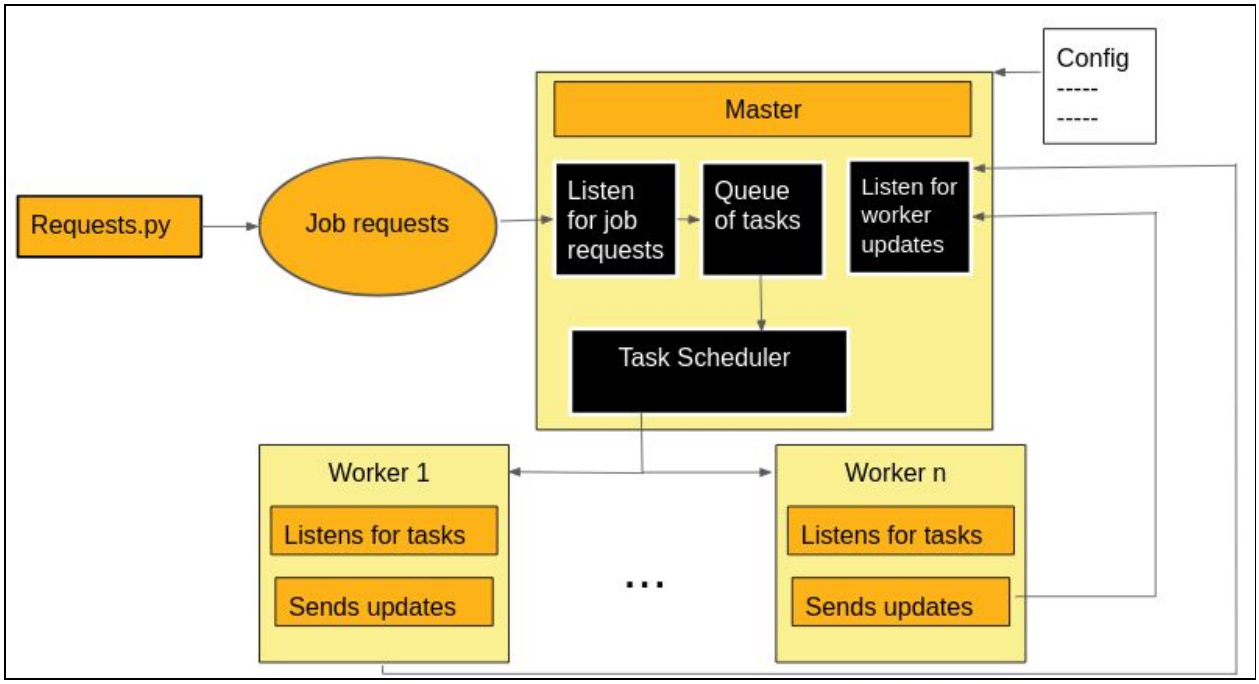
This project 'Yet Another Centralized Scheduler' (YACS) is a tool for simulating the working of a distributed processing system on big data. It is responsible for scheduling jobs from various applications among numerous workers. Practically, the big data workloads run on many different clusters on multiple machines that are interconnected to each other. Though YACS is built to execute on a single machine, it is coded in such a way that it behaves as if each process is running on a separate machine. This simulated framework consists of one master and multiple workers**.** Each of the workers is simulated as though it is on a separate dedicated machine and each machine is divided into a fixed number of equal-sized slots which are resource encapsulations. Through our abstraction, we have simulated a centralized framework with established communication between the master and its workers, inter-task dependency protection and scheduling,

Related work

All forms of communication between the master and workers were through sockets- receiving requests, and making updates. A thorough understanding of sockets was required, for which we referenced articles on the internet related to sockets and communication. Processes on the master and the workers were to be executed parallely using threads and demanded a sound understanding of job scheduling algorithms. Thus, we referred to the textbook 'Operating System Concepts' and other internet articles for their implementation.

Design

Workflow



Master

In a realistic implementation, the Master process bears the onus of executing and managing clusters in machines with different slots. The concept of slots is only an abstraction here, but the master is still responsible for task assignments, maintaining a log of jobs and scheduled tasks, selecting the requested scheduling technique and communicating and with the various workers. The workflow of the Master is charted as below:
The working of the master can be broadly identified as comprising of 4 different stages:

1) Thread creation: The master calls three threads that are responsible for the following:
    - Thread 1: Listens for job requests at port 5000
    - Thread 2: Listens for updates from the three workers- whether the task has been completed, at port 5001
    - Thread 3: Schedules the different tasks in the task queue
2) Receiving requests:
   Upon receiving requests to perform jobs at port 5000, the master must now maintain records of the jobs and their associated map and reduce tasks. It does this by updating its shared resources and by performing job tracking using log files.
3) Scheduling different tasks:
   The master must now ensure that the tasks are implemented using appropriate scheduling algorithms and appropriate steps are taken to ensure that no reduce task is assigned before the mapper task of the job.
4) Scheduling algorithms:
    1. Round Robin:
       The machines are ordered based on their worker ids. The Master has to pick a machine in round-robin fashion using a circular queue. If the machine does not have a free slot, the Master moves on to the next worker id in the

ordering. A slot number of 0 indicates that all slots have tasks scheduled in them. This process continues until a free slot is found.

2. Random Algorithm:
   The Master chooses a machine at random. It launches tasks in different workers based on slot availability.
3. Least Load:
   The Master looks at the state of all the machines and checks which machine has the most number of free slots. It then launches the task on that machine. If none of the machines have free slots available, the Master waits for 1 second and repeats the process. This process continues until a free slot is found.

Each of the schedulers now have the responsibility of scheduling the tasks among the free workers

Workers

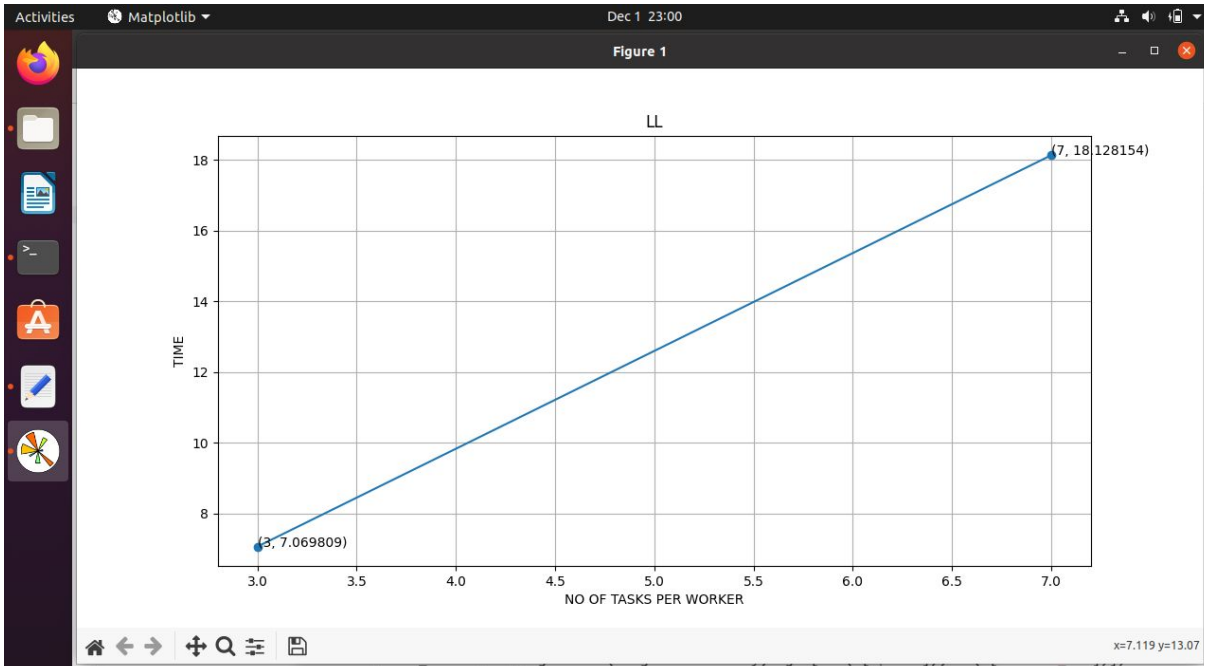
The working of workers consists of 3 different stages:

1) Listening for requests: The worker receives task execution messages from the master at the port, and keeps a record of the task information using log files
2) Task performance: The worker now spawns a thread to begin executing the task that has been assigned to it by the master. It also keeps decrementing the time allotted for the task until it reaches 0.
3) Updating the master: The worker must now update the master on completion of the task, and update the log files with the task completion information.

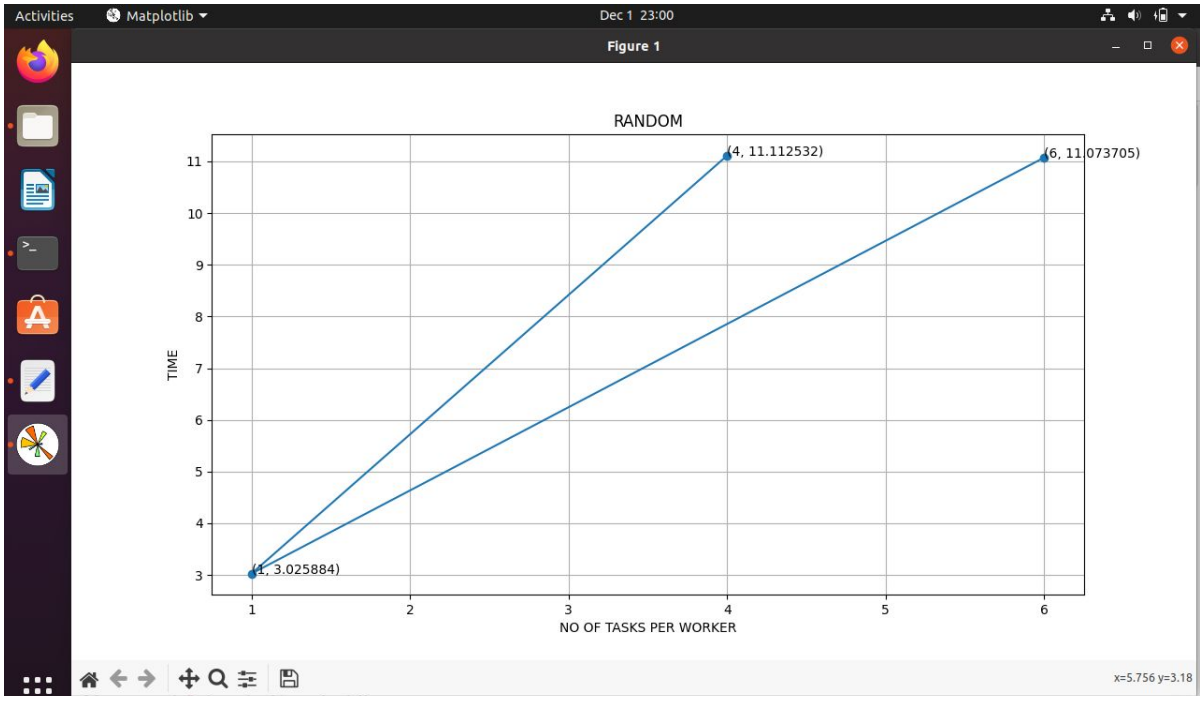
Results

```
Master LL

Median Job Completion 7.070238
Mean Job Completion 5.744102666666667

WORKERS LL

Median Task Completion 3.024811
[18.128154, 7.069809] [7, 3]
Mean Task Completion 2.5197963000000003

Master RANDOM

Median Job Completion 5.042403
Mean Job Completion 5.4007656666666675

WORKERS RANDOM

Median Task Completion 2.008711
[11.073705, 3.025884, 11.112532] [6, 1, 4]
Mean Task Completion 2.292011

Master RR

Median Job Completion 8.148912
Mean Job Completion 6.483476

WORKERS RR

Median Task Completion 1.032112
[6.113539, 11.198076, 10.168403] [3, 4, 4]
Mean Task Completion 2.4981834545454547
```

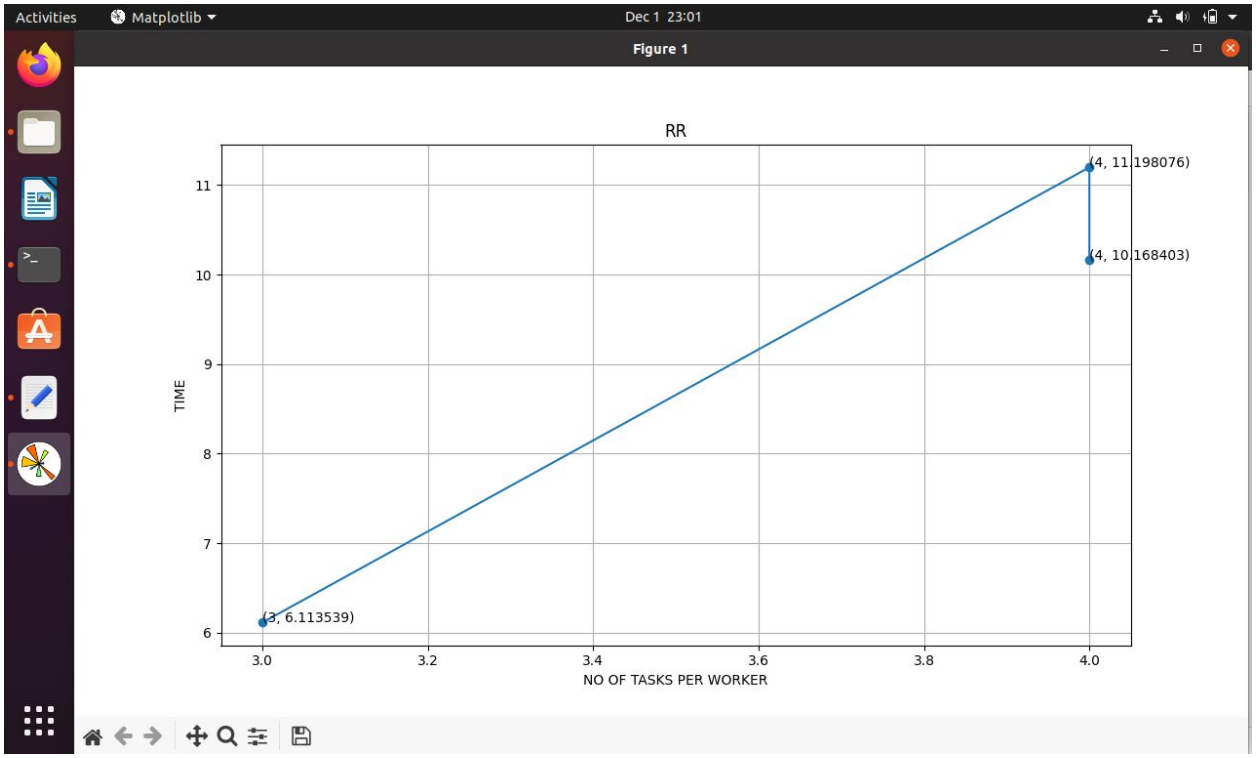
The above figure shows the mean and median job and task completion times from both the master and the workers.



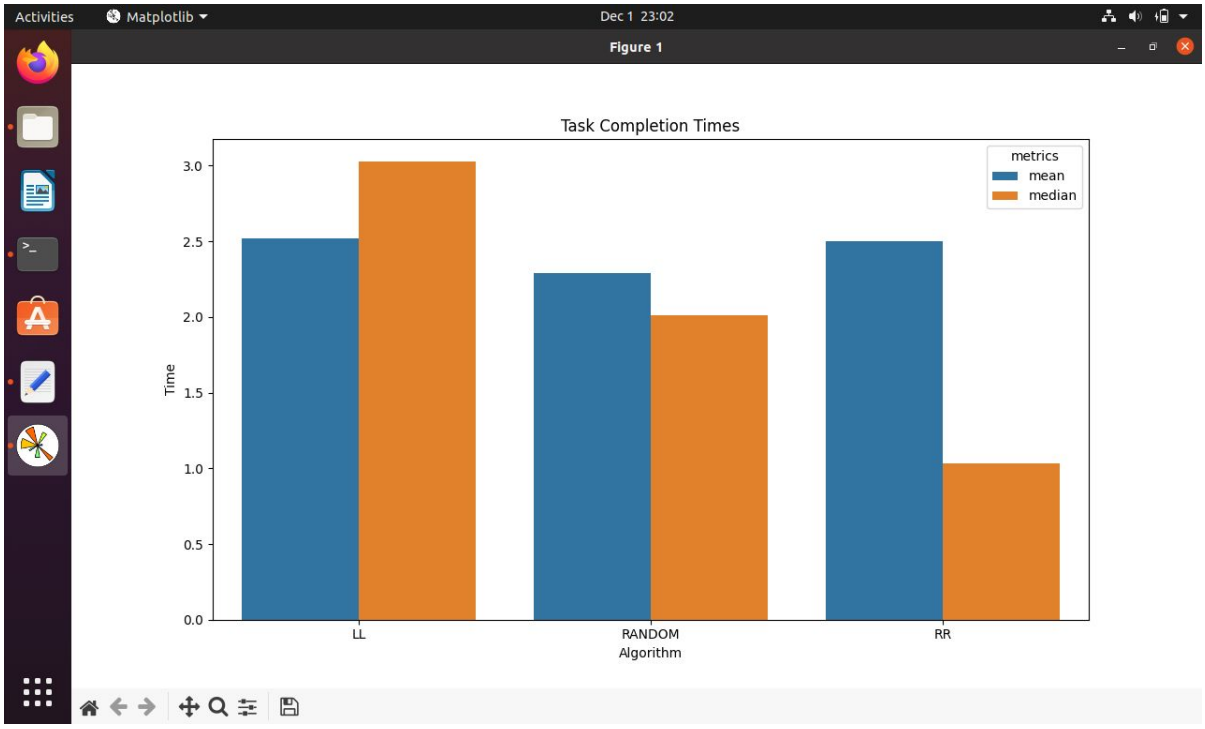
For Least loaded Algorithm it can be seen that task requests were sent to two workers one with 3 tasks which took a total time of 7.07 secs and another that ran 7 tasks for 18.12 secs.



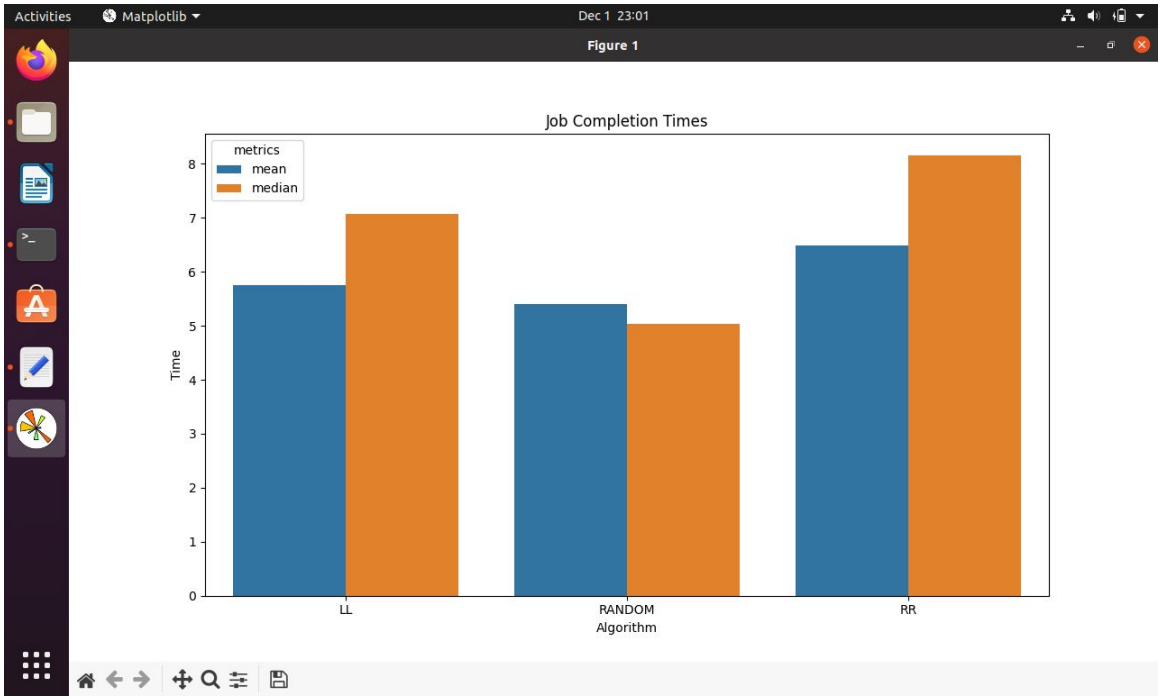
Three workers received requests from the master and from the line chart the total number of tasks and the time to complete those can be visualized.



It can be seen that the workers are assigned the tasks almost equally because of the circular scheduling manner in Round Robin algorithm.



The mean and median task completion time for each of the scheduling algorithms is visualized using a grouped bar chart which shows that RANDOM algorithm worked well for the particular worker configuration and job requests.



From the above bar graph it can be seen the mean and median job completion times for the job requests is better when scheduled using Random algorithm when compared with the others.

Problems

1. A task was scheduled many times which was solved by having a variable to check if a task was scheduled
2. Some of tasks were marked as scheduled even without sending a request to the worker, this was solved by spawning a thread which ran a function to schedule the tasks in the queue.
3. Parsing the log files for calculating the mean and median job and task completion times was difficult.

Conclusion

Centralized schedulers play a pivotal role in Big data, where workloads consist of numerous jobs from different applications. Building a scheduling framework that properly manages and allocates the resources of the cluster is imperative to provide good performance. Working on an abstraction of these schedulers brought out the importance and complexity involved in managing threads, inter-task dependencies, and appropriate scheduling techniques, The process of signalling between workers and the master demanded an understanding of sockets and socket programming. All of this helped us appreciate the architectural prowess of distributed processing frameworks like Hadoop, YARN.

EVALUATIONS:

SNo	Name	SRN	Contribution (Individual)
1	Devika S Nair	PES1201800372	LL Algo
2	K M Mitravinda	PES1201801872	Random Algo
3	Prathima B	PES1201801889	RR Algo
4	Ramya C P	PES1201801554	Base Structure of Master, worker

(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.	