CS322:Big Data

Final Class Project Report

Project (FPL Analytics / YACS coding): YACS Coding Date: 1st December, 2020

SNo	Name	SRN	Class/Section
1	Harish S	PES1201801965	H
2	Suhas R.	PES1201800186	C
3	Vikshith Shetty	PES1201801555	C
4	Prateek Nayak	PES1201800054	C

Introduction

Yet Another Centralized Scheduler (YACS) is as the name suggests a centralized scheduler running on the Master of the distributed system that is responsible for scheduling tasks on the worker nodes. In this implementation, we present a YACS implementation that can schedule task on the workers using the Round Robin Scheduling, the Least Loaded Scheduling and Random Scheduling and compare how they stack against one another in terms of performance and load distribution

Related work

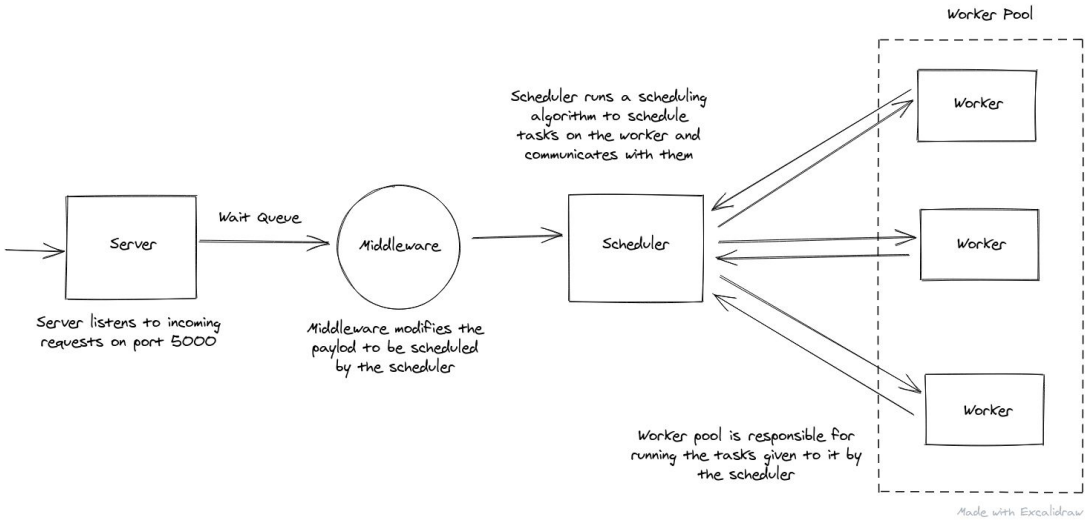
The idea of centralized scheduling is old but to understand the system design better, we referred the the sides of our Big Data course (UE18CS322) and the [Jeff Dean's paper covering Map Reduce published in 2004]

Design

The design is very simple for this minimal implementation. There sits a master on port 5000 listening for any incoming requests. When a request is heard, the payload is retrieved and converted from a JSON string to a Python Dictionary. This dictionary is put on a wait queue which a middleware listens on. The middleware breaks down the request payload into scheduler payload and puts it on the scheduler queue. The scheduler listens to the scheduler queue for any jobs and schedules them on the workers as and when they come based on predefined algorithms:

- Random Scheduling
- Round Robin Scheduling
- Least Loaded Scheduling

The block diagram illustrates the flow of request within our YACS implementation



Results

The result we got are as follows

Result 1:

For the Master

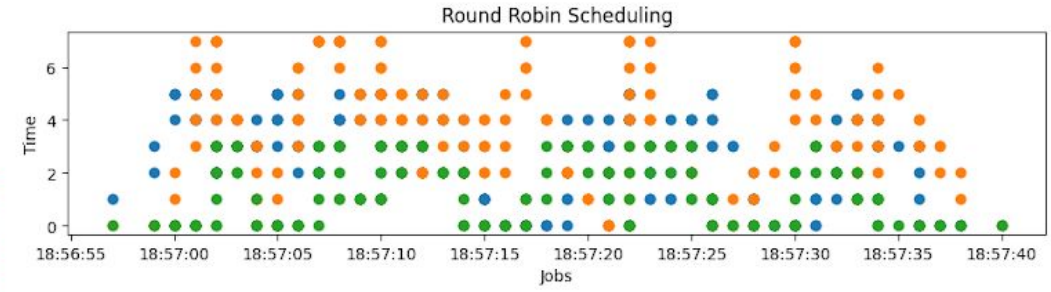
Algorithm	Request Count	Mean	Median
ROUND-ROBIN	5	6.40932	6.007197
LEAST-LOADED	5	6.61091	6.009166
RANDOM	5	6.40958	6.007993
ROUND-ROBIN	10	6.90843	7.5073805
LEAST-LOADED	10	7.00949	7.508435
RANDOM	10	7.43805	7.834915
ROUND-ROBIN	15	7.42021	8.008216
LEAST-LOADED	15	7.67147	8.009575
RANDOM	15	7.34679	7.009864
ROUND-ROBIN	20	7.15577	7.0099395
LEAST-LOADED	20	7.10098	7.507414000000001
RANDOM	20	7.91305	8.009386500000002
ROUND-ROBIN	25	6.56921	7.007503
LEAST-LOADED	25	7.58613	8.008325
RANDOM	25	7.51797	8.007431

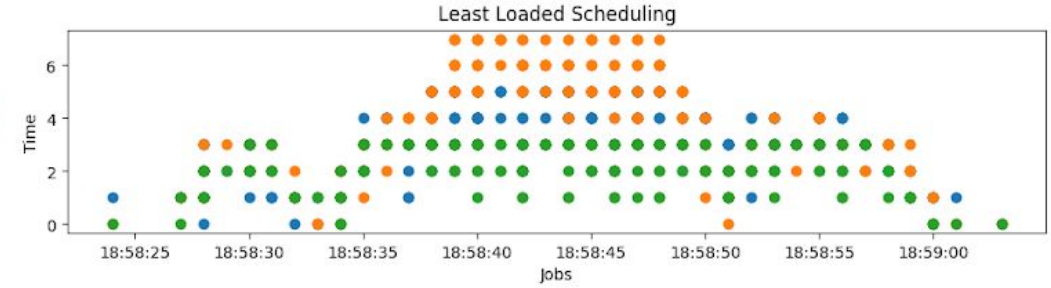
For the Worker we have

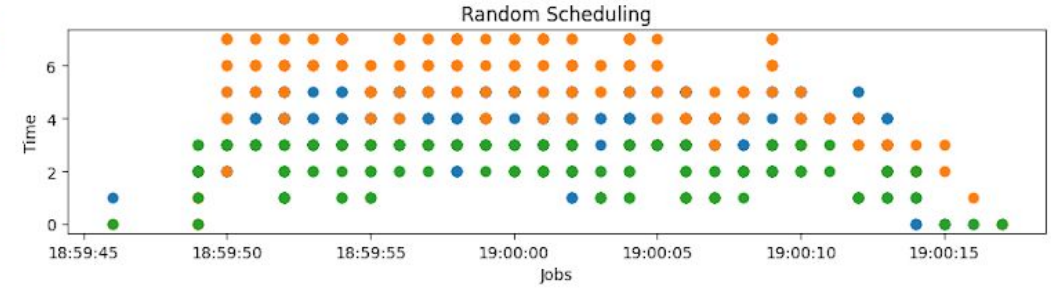
Algorithm	Request Count	Mean	Total Jobs	Median
ROUND-ROBIN	5	2.55784	115	3.002083
LEAST-LOADED	5	2.77475	22	2.5018979999999997
RANDOM	5	2.77459	22	3.501647
ROUND-ROBIN	10	2.71259	45	4.001491
LEAST-LOADED	10	2.71289	45	3.002386
RANDOM	10	2.71241	45	3.0022595
ROUND-ROBIN	15	2.65354	69	2.5012350000000003
LEAST-LOADED	15	2.65344	69	2.002219
RANDOM	15	2.65368	69	2.501437
ROUND-ROBIN	20	2.51754	93	2.002008
LEAST-LOADED	20	2.51745	93	2.002082
RANDOM	20	2.51731	93	3.003263
ROUND-ROBIN	25	2.57156	114	2.002686
LEAST-LOADED	25	2.57159	114	3.000764
RANDOM	25	2.57151	114	3.001528

Result 2

The plot of number of tasks running on each worker for each algorithms is as follows







Problems

Some of the problems we faced were:

- Using Asyncio to handle requests asynchronously
- Evaluating correctness of algorithm
- Analyzing logfile using regular expression

Conclusion

Some of learning we have taken away from the project are:

1. Asynchronous programming is better when it comes to event driven program
2. We learnt the basic syntax of asyncio and hacks to maintain and cleanup the task pool
3. We learnt the difficulties of asynchronous code with respect to race condition and data races they can cause
4. We briefly learnt how task scheduling works on a distributed system with a centralized master
5. Listening to class can help one undertake projects better.

EVALUATIONS:

SNo	Name	SRN	Contribution (Individual)
1	Harish S	PES1201801965	Analysis, Scheduling
2	Suhas R.	PES1201800186	Scripting, Worker Architecture
3	Vikshith Shetty	PES1201801555	Scripting, Scheduling
4	Prateek Nayak	PES1201800054	Server Architecture, Task Synchronization

(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.	