CS322:Big Data

Final Class Project Report

Project (FPL Analytics / YACS coding): YACS coding **Date: 1/12/2020**

SI No	Name	SRN	Contribution (Individual)
1	Abhishek Shyam	PES1201801743	
2	H M Thrupthi	PES1201801987	
3	Hemanth Alva R	PES1201801937	
4	Sreekanth R Gunishetty	PES1201801467	

YACS: Yet Another Centralized Scheduler

Introduction

At a high level, a big data strategy is a plan designed to help oversee and improve the way acquire, store, manage, share and use data within and outside of the organization. Big Data workloads consist of multiple jobs from different applications. A scheduling framework is used to manage and allocate the resources of the cluster (CPUs, memory, disk, network bandwidth, etc.) to the different jobs in the workload.

YACS is the centralized scheduling framework consists of one Master, which runs on a dedicated machine and manages the resources of the rest of the machines in the cluster. The other machines in the cluster have one Worker process running on each of them. The Master process makes scheduling decisions while the Worker processes execute the tasks and inform the Master when a task completes its execution.

Related work

Earlier studies already demonstrated the superior convergence rates of YACS compared to modern commercial 3D solvers. [1] Whereas, with the most recent revision of YACS the emergence of spurious modes, when solving for eigenmodes with an azimuthal mode number m 0 and utilizing higher order basis functions for the field or geometry discretization, could be successfully suppressed.

The paper on Scheduling algorithms summarizes the state of the real-time field in the areas of scheduling and operating system kernels.[2] Given the vast amount of work that has been done by both the operations research and computer science communities in the scheduling area, they discussed four paradigms underlying the scheduling approaches. The four paradigms are: static table-driven scheduling, static priority preemptive scheduling, dynamic planning-based scheduling, and dynamic best effort scheduling.

Design of YACS

In the YACS, the framework consists of one Master, which runs on a dedicated machine and manages the resources of the rest of the machines in the cluster. The other machines in the cluster have one Worker process running on each of them. The Master listens for job requests and dispatches the tasks in the jobs to machines based on a scheduling algorithm. e. The Master is informed of the number of machines and the number of slots in each machine with the help of a config file.

The Worker processes listen for Task Launch messages from the Master. On receiving a Launch message, the Worker adds the task to the execution pool of the machine it runs on. The execution pool consists of all currently running tasks in the machine.

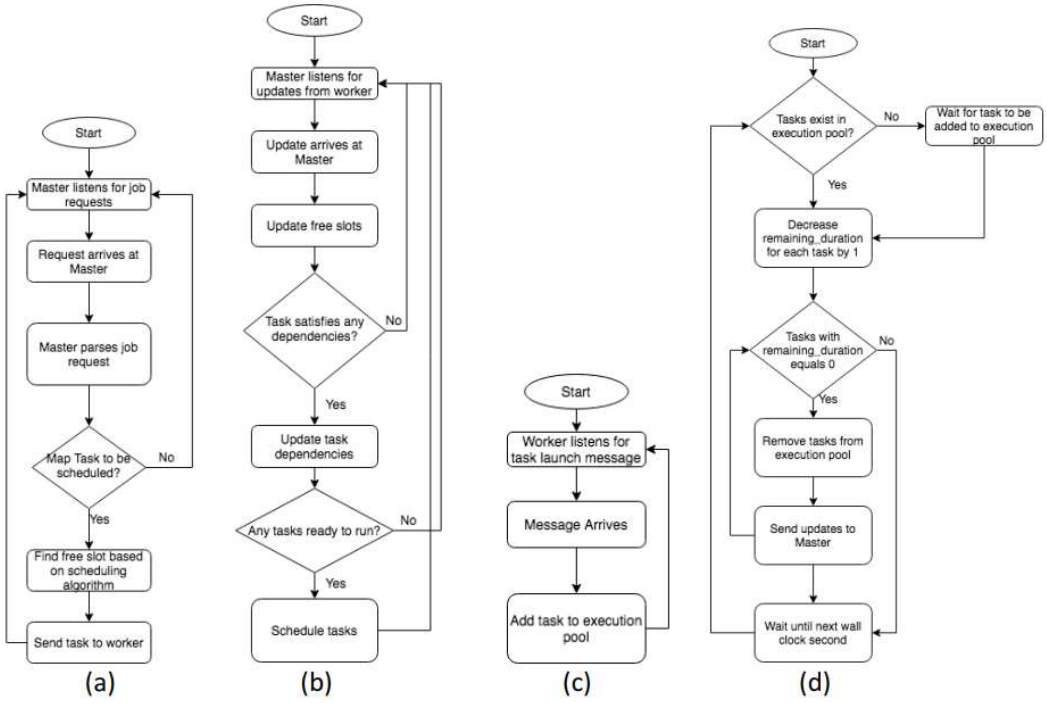
Implementation of YACS Model

In the functioning of YACS simulated framework that get created, there is 1 Master process and 3 Worker processes. All the processes run on the same PC, however, they must behave as though they are on separate dedicated machines.

All jobs have 2 stages only.

- The First stage consists of map tasks
- The Second stage consists of reduce tasks.

The reduce tasks in a job can only execute after all the map tasks in the job have finished executing. There is no ordering within reduce tasks, or within map tasks. All map tasks can run in parallel, and all reduce tasks can run in parallel.



Each Worker's machine will be configured with a fixed number of slots. The Master needs to keep track of the number of slots available in each machine. The Master and the Workers need to maintain a log of important events.

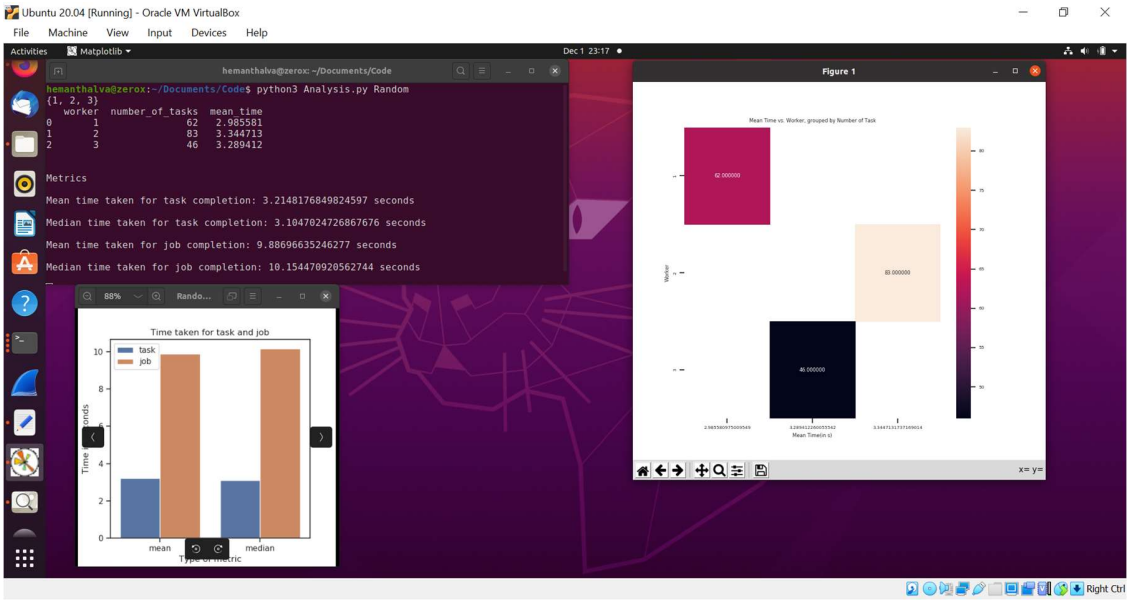
The Master will need to have at least 2 threads in order to listen for job requests and updates from Workers. Each Worker will need to have at least 2 threads for listening for task launch messages from Master and to simulate the execution of the tasks and send updates to the Master. The Worker processes listen for Task Launch messages from the Master. On receiving a Launch message, the Worker adds the task to the execution pool of the machine it runs on. The execution pool consists of all currently running tasks in the machine.

When a task completes execution, the Worker process on the machine informs the Master. The Master then updates its information about the number of free slots available on the machine.

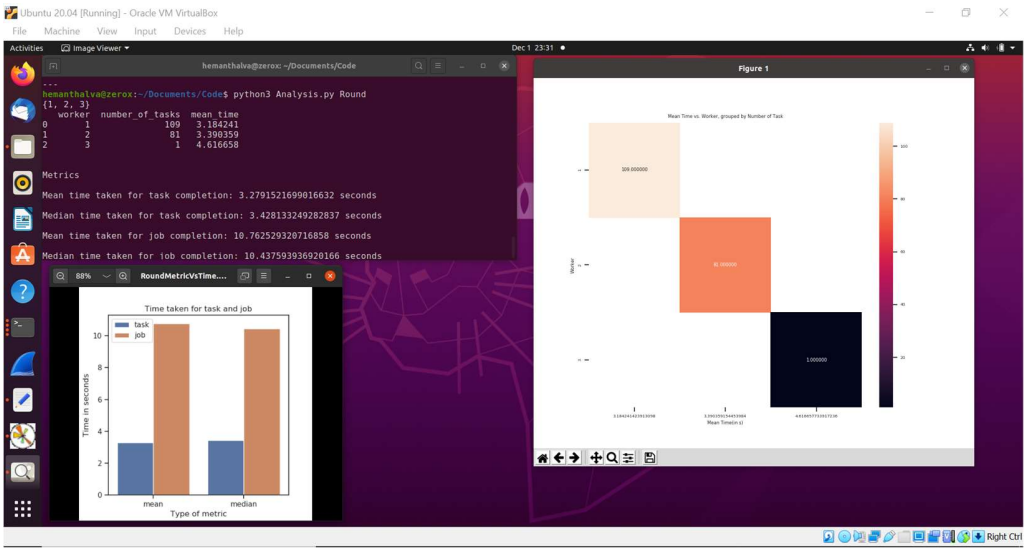
Scheduling Algorithms

The Master listens for job requests and dispatches the tasks in the jobs to machines based on a ***scheduling algorithm***.

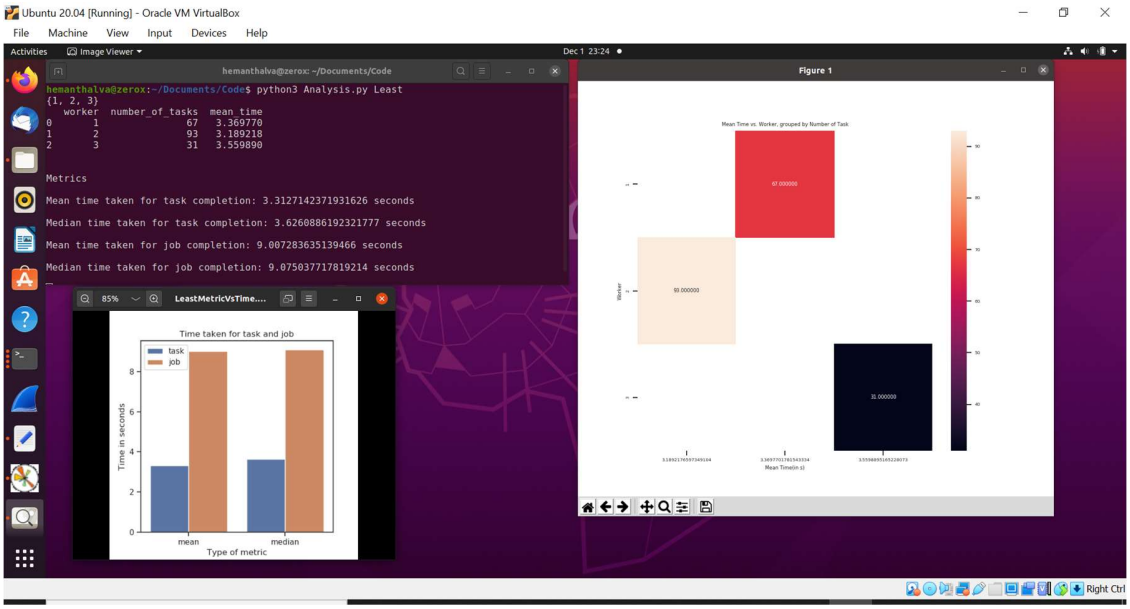
1) **Random:** The Master chooses a machine at random. It then checks if the machine has free slots available. If yes, it launches the task on the machine. Else, it chooses another machine at random. This process continues until a free slot is found.



2) **Round Robin**: The Master chooses a machine at random. It then checks if the machine has free slots available. If yes, it launches the task on the machine. Else, it chooses another machine at random. This process continues until a free slot is found.



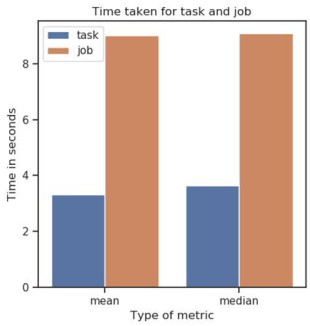
3) **Least-Loaded:** The Master looks at the state of all the machines and checks which machine has the most number of free slots. It then launches the task on that machine. If none of the machines have free slots available, the Master waits for 1 second and repeats the process. This process continues until a free slot is found.



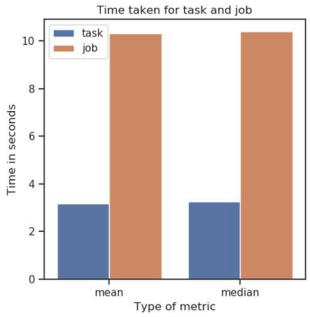
Results

Scheduling Algorithm	Mean time for Task Completion	Median time for Task Completion	Mean time for Job Completion	Median time for Job Completion
Random	3.21	3.10	9.88	10.15
Round Robin	3.27	3.42	10.76	10.43
Least-Loaded	3.31	3.62	9.00	9.07

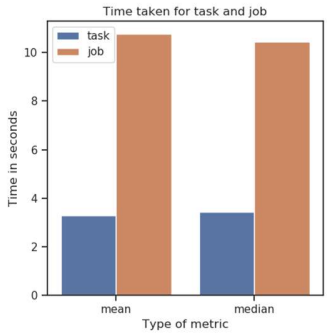
Least-Loaded Vs Time



Random Metric Vs Time



Round Robin Vs Time



Problems

Problems are faced during the update of number of free slots in the machine. The Worker processes didn't get listened for Task Launch messages from the Master. Then, the taskLaunchSocket used to connect localhost with appropriate port to resolve the problem.

Conclusion

We have learnt an implementation of YACS Model and its importance. The Scheduling Algorithms has processed on YACS and obtained various mean time and median time for both Job and Task completion, which impacted on us to choose the appropriate algorithm for given real time applications.

References

1) Isbarn, B. D., et al. "Cavity Characterization Studies With the Latest Revision of YACS." *9th Int. Particle Accelerator Conf.(IPAC'18), Vancouver, BC, Canada, April 29-May 4, 2018*. JACOW Publishing, Geneva, Switzerland, 2018.
2) Ramamritham, Krithi, and John A. Stankovic. "Scheduling algorithms and operating systems support for real-time systems." *Proceedings of the IEEE* 82.1 (1994): 55-67.

EVALUATIONS:

SNo	Name	SRN	Contribution (Individual)
1	Abhishek Shyam	PES1201801743	
2	H M Thruthi	PES1201801987	
3	Hemanth Alva R	PES1201801937	
4	Sreekanth R Gunishetty	PES1201801467	

(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.	