CS322:Big Data

Final Class Project Report

Project (FPL Analytics / YACS coding): __YACS coding__ Date: __1/12/2020__

SNo	Name	SRN	Class/Section
1	Tejas Srinivasan	PES1201800110	5-'A'
2	Arjun Chengappa	PES1201800119	5-'A'
3	Lavitra Kshitij Madan	PES1201800137	5-'A'
4	Priyanka Gopi	PES1201801797	5-'B'

Introduction

Introduce your project.

Yeet Another Centralized Scheduler (christened with the same acronym provided) is a a master-slave architecture that is implemented on one machine but can be scaled to include many workers on many different machines. The 'master' is hosted on a dedicated machine and it schedules request from and allocates resources to 'workers' present in a cluster of machines (although it is all in the same machine in the current implementation).

The master consists of

A block diagram of the intricacies of the components -

Related work

Any background study material that you may have read and referenced

Reference books -

Core Python Applications Programming, 3rd edition by Wesley Chun

Operating System Concepts by Abraham Silberschatz, Peter B Galvin, Gerg Gagna

Operating Systems: Three Easy Pieces by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau (University of Wisconsin-Madison)

Computer Networks: A Top Down Approach by Jim. F. Kurose for socket programming concepts

Reference websites (articles/papers etc) -

RealPython and w3schools for Pythonic syntax and implementation of data structures and other concepts required.

GeeksforGeeks for concepts for implementation of sockets,locks,threads etc.

And of course, a lot of Stackoverflow :P

Design

Talk about the design of the system, algorithms used, and models implemented. Block diagrams are preferred wherever applicable.

Results

Discuss the results you got. What inferences could you draw from the results? Was any result unexpected? Any fine-tuning done to parameters so that the results changed?

The Results obtained are basically the results obtained by plotting the graphs in task 2 of the project -

Time taken to complete tasks vs Task IDs for each of the Scheduling Algorithms - (Can specify for a certain number of jobs in the requests.py file)

Random Selection -

Round Robin -

Least Loaded -

Problems

Since we had initially started off with the FPL project we would like to mention that the streaming and the usage of RDDs in PySpark and their conversion were a problem we spent a lot of time trying to solve but could not and hence moved onto the YACS scheduler project.

While we started working on the YACS, we found that the scheduler stops assigning tasks to the worker once all the slots were being utilized without checking for which ones were free.

We found a solution to this problem by having a separate thread on which the yeetacs centralized scheduler was run to read all the tasks from the queue and schedule them in order.

There was also initially the problem of correctly interpreting the log files after the master-worker exchange which proved to be an issue while plotting the graphs for Task-2 but this was quickly dealt with by the usage of 'dictionaries' instead of lists to store the tasks and job execution times.

PS - We had also faced various compile-time issues as in natural with a project and also run time issues with dying threads and unresponsive workers.

The time we found to work on the project satisfactorily was also insufficient.

Conclusion

Through this project we learnt a lot about the workings of a master-worker architecture, how it scales and how to make the implementation possible for workers located on a cluster of machines. Simulation of these tasks using concepts of socket programming,threads etc. gave us a lot of insight on the core subjects in computer science.

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.	