CS322:Big Data

Final Class Project Report

Project (FPL Analytics / YACS coding): _YACS_ Date:

SNo	Name	SRN	Class/Section
1	Nagabhushan M Hegde	PES1201801460	5 D
2	Vishwas Badiger	PES1201801609	5 F
3	Goutham S	PES1201801687	5 D
4	Shripad Vernekar	PES1201801932	5 D

Introduction

YACS,a centralized scheduling framework.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 process makes scheduling decisions while the Worker processes execute the tasks and inform the Master when a task completes its execution.

Related work

We weren't aware of many important concepts required to do this project like threading, locks for synchronisation and practical use of socket programming. So we had to look into many online resources in order to understand the above concepts.

Design

The three scheduling algorithms we used to pick a machine for a task are: random, round--robin and least--loaded.

- random : The master chooses a machine or worker at random and then it checks if the machine has free slots available if yes, it launches to that machine else chooses another machine at random this continues until a free slot is obtained.
- round-robin : The machines are ordered based on worker id and the master picks the worker in round robin fashion that is if the machine does not have a free slot the master moves on to the next worker id in the ordering.The process is continued until a free slot is found.
- least-loaded : The master looks at the state of all machines and checks which has the most number of free slots and launches the task on that machine(worker) , if none of the machines have a free slot available the master waits for 1 second and repeats the process.

master.py

First sockets are created.Total of 5 sockets are used one for request, accept message and other 3 to send messages to workers. 3 Threads were used to complete this process.First thread was used to accept the message from the request file,second thread is used to update the slots in the worker based on the messages received from the worker and last one is used to launch the reduce tasks.

- accept_message : It accepts the job requests from the requests file and then increments the job count by 1 and later jobs are added to the scheduling pool.Here 2 dictionaries are used as data structures to implement scheduling pool i.e. one for map tasks and other for reduce tasks. The map tasks are sent to the scheduling algorithms functions which selects a worker for a particular task based on available slots.Here the scheduling algorithm is given as user input.To implement scheduling pools locks are used.
- assign_tasks : It reduces the total number of tasks by one and slots count is reduced by one for that worker selected.Connection for selected worker is established and message is created.Created message includes job id,job type , duration and start time and then time log is created using a dictionary.This message is encoded and sent to selected worker.Here locks are used for reducing the slots.
- reduce_tasks : It check whether all the map tasks for a particular job is completed if yes,it sends reduce tasks to the scheduling algorithm function.
- update_tasks : It receives message from worker. This message includes job id,job type, task id, work id, begin time, end time.Number of slots for that worker is increased by one, time taken to complete the task is calculated by taking the difference of end and start time.Checks for which task has been completed i.e. map or reduce tasks and that completed task is removed from the scheduling pool if no other tasks are there to complete then total job time is calculated and job count is reduced by one.

  If all jobs are completed, then those calculated time is appended to the file for *further analysis*.

worker.py : Here we have used 2 threads, one for listening for task launch messages from the master and other to simulate the execution of tasks and to send updates to the master about the task completion.

- worker : This functions listens to the task launch messages from the master,decodes that message records begin time and then it is added to the execution pool, locks are used here.Execution pool contains the worker id and message sent from master.
- simulate_execution : Here 1st check if execution pool is empty if yes, then waits until task is added to the execution pool else decrements duration and if duration becomes 0 then it records end time and message update is sent to the master and that particular task to be removed is stored in list, after this at the end those tasks are removed from the execution pool, locks are used while removing the tasks.

Results

The results discussed below are for the 10 job requests.These were calculated in analysis.py file.

Random scheduler :

Job	**Task**
mean : 23.029	mean : 7.718
median : 19.631	median : 5.508

Round-robin scheduler :

Job	**Task**
mean: 30.416	mean : 10.748
median: 32.351	median : 9.778

Least-loaded scheduler :

Job	**Task**
mean: 21.682	mean : 7.528
median: 23.089	median : 5.70

Problems

We faced a lot of problems in threading and socket programming, so we went through many of the resources to get to know about them and we spent a lot of time understanding the concepts and after all these we faced problems in synchronizing the process.

Conclusion

We learnt about threading, socket programming and how to resolve the race conditions and how master and workers work in YACS.

EVALUATIONS:

SNo	Name	SRN	Contribution (Individual)
1	Nagabhushan M Hegde	PES1201801460	accept,worker,document

2	Vishwas Badiger	PES1201801609	Scheduling and analysis
3	Goutham S	PES1201801687	assign,reduce,update
4	Shripad Vernekar	PES1201801932	Worker,document

(Leave this for the faculty)

Date	Evaluator	Comments	Score

CHECKLIST:

SNo	Item	Status
1.	Source code documented	Done
2.	Source code uploaded to GitHub – (access link for the same, to be added in status)	https://github.com/agasthya36/BD_YACS
3.	Instructions for building and running the code. Your code must be usable out of the box.	run master.py syntax: python3 master.py config.json scheduler (roundrobin,leastloaded,random) run worker.py syntax: python3 worker.py port_num worker_id Then run request.py syntax: python3 request.py number_of_jobs Then run analysis.py