ECE6101 Course Project
Distributed Batch Processing System
Option 1

In this project option, you are provided with a simple but functional batch processing system. The package (*project.tar.gz)* can be downlaoded from t-square. The source code is written in java and is thus OS independent. However, you are required to use Linux for this project. If you do not have Linux installed natively, you can either use the provided server or create a virtual machine on your own computer (highly recommended).

1. About of the given system
   Our batch processing system consists of three components: client, scheduler, and worker. The client is used to submit a user job to the scheduler. The scheduler will assign the jobs to the workers. The worker will carry out the actual computation.

   In the given source code you can also find two other directories: “common” and “jobs”. The “common” directory contains Job, JobFactory, and Opcode, which are shared by multiple system components. The abstract class common.Job describes the job model and should be inherited in the implementation of every job. A job in this system consists of a group of independent tasks. Similar to the SIMD (Single Instruction Multiple Data) programming model, all the tasks of one job share an identical task function, and they can work on different data based on their unique taskId. To code a job, the user needs to implement the task function and specify the number of tasks. The class common.JobFactory is a tool used to load the job class from a job file (.jar file). A user job is submitted to the system as a job file plus a class name, such that the system has to extract the job class from the received file before executing the job. The class common.Opcode packs the code words used in the communication among the system components.

   The “jobs” directory contains two user job examples. The jobs.Hello is a simple hello world example. It has two tasks, each will print a hello message and taskId. The jobs.Mvm is matrix-vector-multiplication example. This example uses four tasks to calculate the products of a matrix and four different vectors.

2. Required features
   In this project, you are asked to implement four features based on the given source code. Most of the coding work are expected to be done within the scheduler and the worker. You are not allowed to change the job model. The given client and its interface to the scheduler should also be good enough (thus no edits are expected).

   - Non-blocking request handling

   In the given system, the scheduler takes a brute-force method that it can handle only one job request or worker registration request at a time. In your system, the scheduler should be able to handle multiple such requests in non-blocking manner, such that multiple jobs can be served simultaneously if multiple workers are available.

- Parallel job execution

  In the given system, the scheduler assigns all the tasks of one job to a single worker. In your system, the scheduler should be able to assign the tasks to multiple workers when necessary. For example, a job with four tasks can be served by two workers in parallel. Please also beware that a worker in our project is designed to represent one node in a real distributed system. Typically, all those nodes have identical computation capability. So, in our project, each worker is allowed to serve only one task at a time.

- Fair job scheduling

  In the given system, the jobs are handled in a First-In-First-Out manner. In your system, a fair scheduling algorithm should be developed. The objective is to let all jobs get, on average, an equal share of resources over time. When there is a single job running, that job can use all the workers. When multiple jobs are presented simultaneously, each job is expected to get roughly the same share of CPU. Unlike the given FIFO scheduler, fair scheduler lets small jobs make progress even if they are sharing the system with large jobs. For the design of such scheduler, you can refer to http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-55.pdf or similar resources about Hadoop Fair Scheduler. You don't have to follow every bit of Hadoop's design, you are encouraged to create your own fair scheduler.

- Fault tolerance

  In the given system, jobs will fail if the any part of the system is malfunctioning. Your system should be able to survive a sudden worker failure. Specifically, if some of the workers are killed (simulating a node failure), your system should be able to keep serving jobs with a shrunk pool of workers. When the killed worker revives, it can be registered again into the system.

3. Scoring process

   Your project will be scored based on the report and the executable code. In the report you should describe how are the required features implemented and how are the experiments designed to test the features. We will also run your system on our server with the configuration of eight workers. A non-disclosed set of jobs would be used to verify the correctness and fairness of your system. We would also test the system's throughput and fault tolerance capability.

4. Miscellaneous

   The access to our server will be provided shortly. The rules regarding the usage and port assignment would also be posted.

   An homework will be posted on Tsquare for your to submit your team formation and the your choice of the project option.

   For questions about the project please contact Jiadong Wu (jwu65@gatech.edu). Please include “ECE 6101” in the title of emails. For common questions we would post response via Tsquare.