**COMP3100 Group Project – Stage 1 Design Document**

1.     Project title

 Intelligent job dispatcher

2.     Group members

Student name: Jiahui Lin, Student ID: 45141916

Student name: Fei Huang, Student ID: 44129866

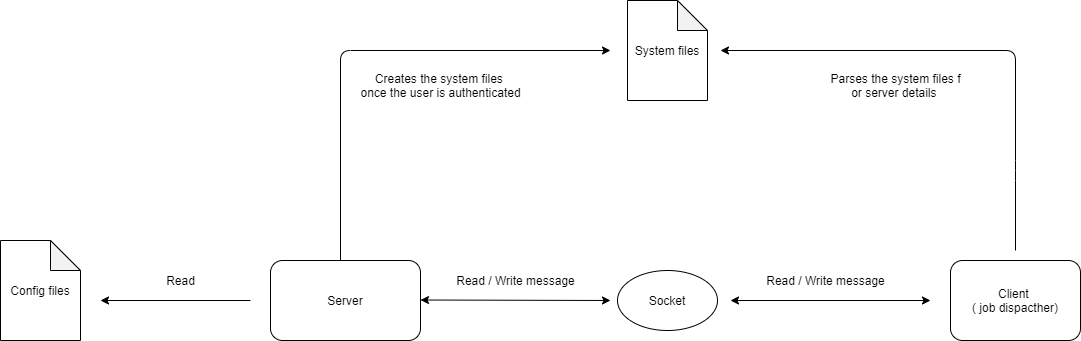
3.     Introduction:

Having a job scheduler is not sufficient enough for a server to deal with the execution of the process (jobs). The main role of the scheduler is to select and queue the jobs to the system and ensures the processes are executed in order. However, during the execution phase, an OS also need to take account of the CPU allocation of each task and be aware of context switching of the  tasks in order to maximize the system performance.

4.     Aims:

Within this project, our goal is to design and implement a client simulator that mimics the behavior of the job dispatcher that handles the CPU allocation of the selected process (job) by the scheduler, moving the process from the waiting state into executing state. Furthermore, the simulator needs to be capable of sending the message to the server and receive and recognize the system command being delivered to the client-side.

5.     Background:

5.1 System overview

5.2 Design consideration

* Acknowledgment of the message being received is vital for the communication between server and client, sender / receiver should hold on and wait for the acknowledgement from the other end of connection to be received before moving further.
* Introduce system runtime exception, once the invalid message is received, the client should halt the communication, terminate the connection with the server.

5.3 System constraint:

* Communication between the server and client is restricted under a preformatted flow, sending a invalid message wouldn’t terminate the connection but the communication cycle would be halted because of not getting any feedback from the other end of the connection. In other words, the server or client is not responsive to invalid messages but it would be halted without informing the other end of the connection what is going on.
* The connection between server and client can only be achieved if they are under the same local network and using the same port for communication.

5.4 System architecture

    As the above image described, the system is consisting of five components.

* 1.     Configuration file

The config file can only be accessed by the server, containing the information that server should consider when creating the jobs, such as execution time, workload and termination status.

* 2.     System files

System files would be created inside the local directory once the user is authenticated successfully. Including the details of servers within a cluster, such as server type, core size, memory size, disk availability and etcetera.

* 3.     Socket

Socket is the middleware between client and server, acting like a postman to deliver messages to the destination.

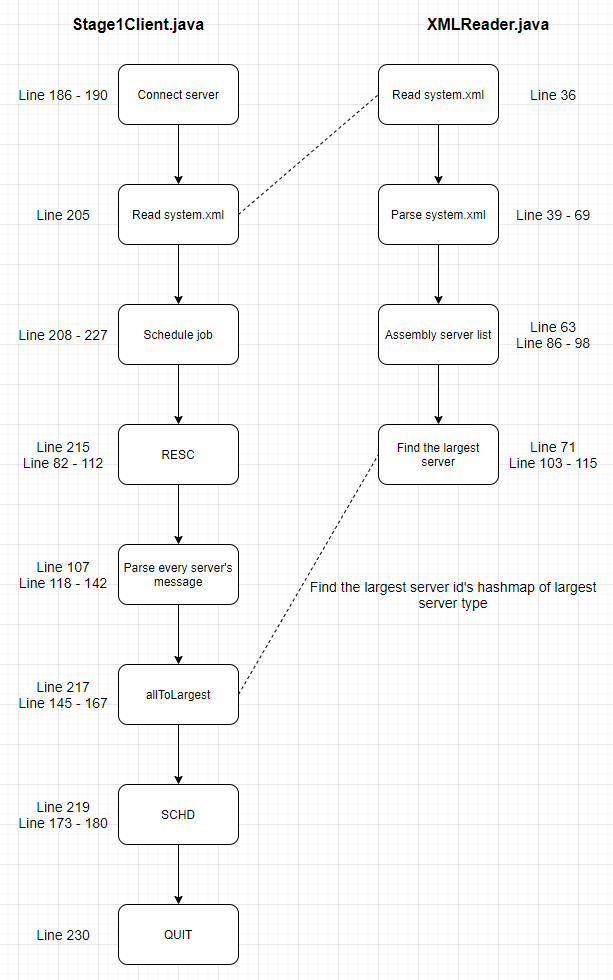
* 4.     Server

Server is designed to process the request sent by the client. It’s capable of authenticating a user’s identity, read and reschedule the job allocation within the config file. Halts the connection once invalid message is received for the purpose of security.

* 5.     Client

Client reads and parsers the system file created by the server. Sends and receives the validated message (command) from / to the server side through the socket. Halts the connection once invalid message is received for the purpose of security.

6.     Implementation details

****