COMP3100 Group Project – Stage 1 Design Document

## Project title

Intelligent job dispatcher

## Group members

|  |  |
| --- | --- |
| Name | OneID |
| Fei Huang | 44129866 |
| Jiahui Lin | 45141916 |

## Introduction

When customers need to maintain thousands of batch tasks on a cloud platform, work efficiency will be an important reference index for them. The main task of stage 1 in this project is to design a mature client system. When the server publishes the corresponding work according to its own strategy, our client can complete the specified tasks accurately and efficiently, and without manual intervention throughout the process, 100% automated execution.

## System overview

The server will create the corresponding system.xml according to the current task flow. When the client and the server successfully communicate, the batch task will begin. The server will first send a JOBN command to the client, and then get the client's RESC request, the scheduled task will officially start. The server will send the server information to the client in turn. After the transfer is completed, the client will find the maximum server\_type and server\_ID of the current job through the allToLargest method, and then return to the server through the SCHD command. After the server receives the information and confirms it is correct, it will start the next schedule job, if any. Otherwise, the client will be notified, and the communication will end.

### Considerations

* If the server fails to accept the client's reply in the batch schedule job due to network communication, the client should be asked to resend the last failed message.
* If the server fails to accept the client's reply in the batch schedule job due to data recovery, the client should recalculate the largest server\_type and server\_ID with the job\_ID, and then resend the information.

### Constraints

* When the client receives the ERR command from the server, it should immediately terminate all current operations and close the communication with the server. Customers should check the log file in time to find the cause of the error.

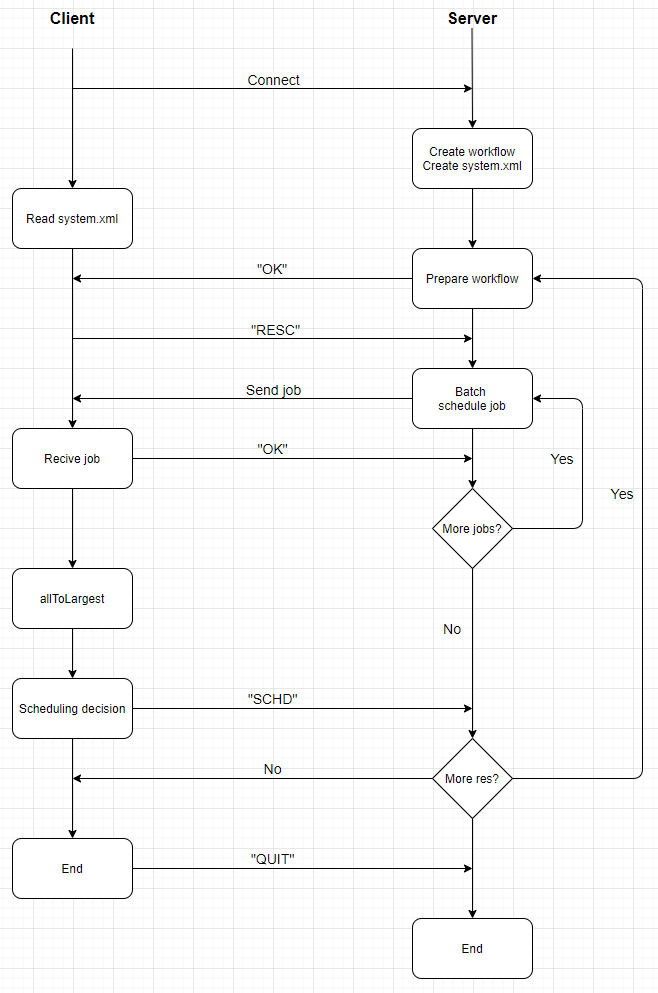


Figure 1 System overview

## System architecture

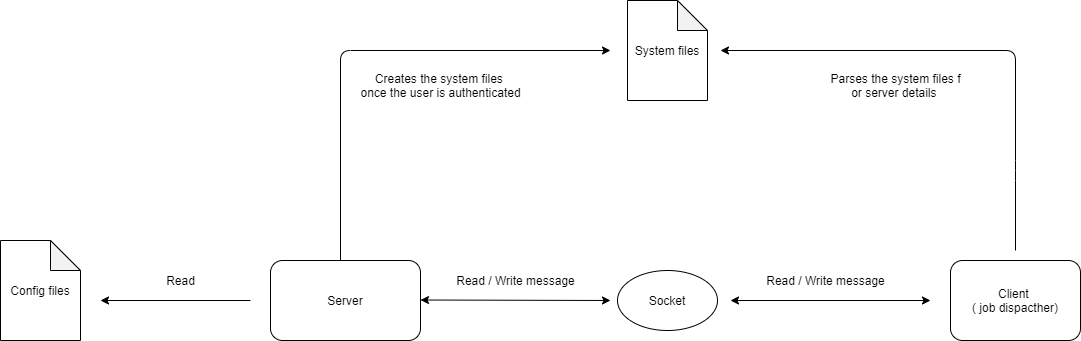


Figure 2 System architecture

As the Figure 2 described, the system is consisting of five components.

* 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.

* 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.

* Socket

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

* 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.

* 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.

## Implementation details

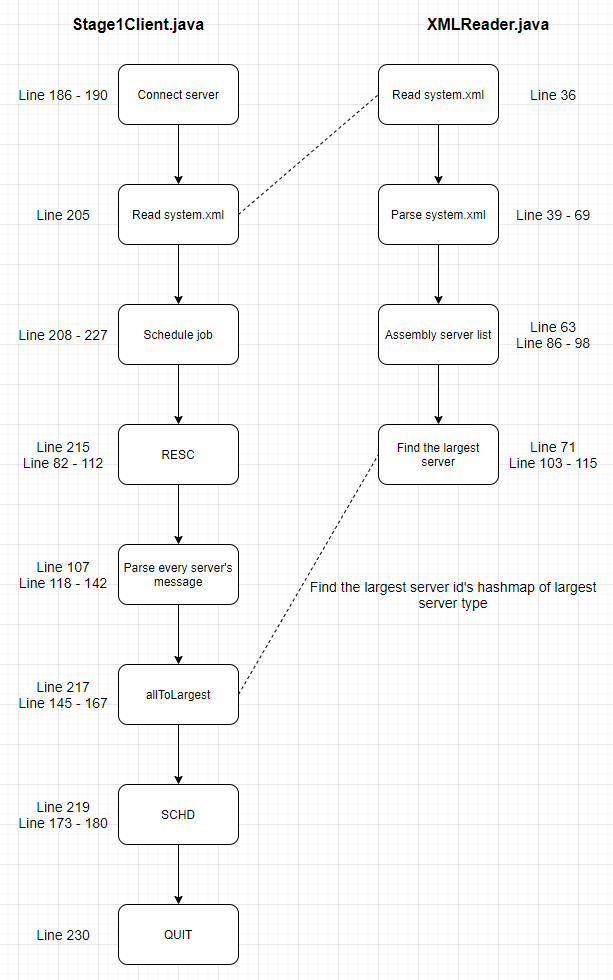


Figure 3 Implementation details

## References

* GitHub: <https://github.com/SnakeCN21/COMP3100-Group-Project>