**Stage 2: Baseline Algorithm Design Document**

**Project Title:** Distributed Systems (DS) Job Scheduler

**Group 51:** Lakshmi Priya Bhuphatiraju (45431957), Tonmoy Ahmed Jitu (4266278)

**Introduction**

The second part of the project was mainly about implementing design algorithms which when executed progresses how jobs are distributed to servers. In the previous stage all jobs had to be allocated to the largest server. With the new data from the XML file attributes form the “job” can be used to better optimize jobs to server depending on the design or type of algorithm we use.

The group decided to work on two algorithms, first-fit and worst-fit. The algorithms will determine how the jobs are allocated according to the needs of the client. Both the algorithm helps understand how server cost and efficiency varies from different types of job scheduling.

**Design consideration and preliminaries: common data structures used and how resource/server state information is checked**

Continuing on with our first design consideration of developing the job scheduler as an Object-Oriented Programme, we developed and build on the code by having different classes consisting of different functions. This allowed us to better develop each program and understand how each individual component worked. After a while of brain mapping, we found out how the classes will interact and fetch data from one another. We have five classes in total, Client, Electrode3100, Server, Jobs, and AlgorithmType.

The *Client* class basically the point from which we initiate the client-server. It isolates itself from all other functions and mainly focuses on choosing the type of algorithm and sending it to the right address with the right port number.

The *Server* class stores information of the server that we collect from the XML files which the server-side send us. The server side stores the value using the attributes from the XML file which allows us to build our own database of servers for job allocation. Compared to stage1 the main difference is that there is a new constructor which allows us to store more information of server which is then used for sorting the server according to *cpuCore.*

The *Jobs* class is more or less like the server. It is an independent class which is used to store jobs attributes that we receive from the server-side.

The *Electrode3100* class is the heart and soul of the algorithm. Compared to stage1 we updated the *run()* method which requests the server-side for job information. The class then uses the information to create its own list of jobs to dispatch to the server. The list of jobs is created to be used later on to dispatch jobs according to the server resource the job requires.

The *AlgorithmType* class is the new class which was introduced in this stage. The reason we decided to implement one whole class for this stage is, having a class allowed us to better implement and call different types of algorithms. The class allowed us to form an instance of the object in other classes and call its functions and methods. This simplified the code a lot and allowed us to develop on areas where we had issues. The class basically takes the two Array List that the Electrode3100 class build and goes through the list and compares values. This particular class also checks the availability of a server before dispatching jobs.

**Algorithm description (one sub-section per algorithm; the group member who implemented a particular algorithm needs to be clearly indicated)**

*First-Fit (Implemented by Tonmoy Ahmed Jitu)*

The idea of the first-fit algorithm is assigning jobs to server, from smallest to largest. For first-fit algorithm to assign jobs, it first needs to check whether the server is available or not. The availability of the server is depended on whether the server has enough memory, disk space and the CPU core count. In order to meet basic criteria of assigning jobs from the smallest to the largest, a function was created which was used to sort servers by *serverType*, this allowed the algorithm to filter out small and large jobs and send it to specific servers. This makes the algorithm efficient as the algorithm aims to reduce running higher server cost.

To explain the algorithm in more detail. As previously mentioned, the main function to design the algorithm was *sortServerID*. This function sorted all server from the Array List of servers according to the ID (smallest to biggest) and stored the servers in a new list. After sorting, the jobs were assigned to each server in the new server array list. Inside the algorithm another condition was used which checks while assigning jobs from smallest server to largest, if the server has enough resource to run the job. If the server cannot host the job, the job is then sent to the next active/available server.

In order to check if a server is active/available the algorithm checks for the server status and compares the value ranging from 0-4.

*Worse-fit (Implemented by Lakshmi Priya Bhuphatiraju)*

The worse-fit algorithm allocates jobs to the largest server available whenever the server becomes available. The server attribute *coreCount* is used to determines the type of server, which is then used to identify the size of the server as well. The algorithm also takes some idea from first-fit to determine whether a server has enough resource like memory and disk space for job allocation.

The worse-fit algorithm is unique in a sense that it has to calculate the *fitness* value. The *fitness* value is basically the different between *cpuCoreCount* from the jobs and *coreCount* of each individualserver. After calculating the fitness value by comparing each *cpuCoreCount,* the algorithm allocates the jobs to next available server with the maximum fitness value.

The worse-fit algorithm checks the availability of the server by requesting to check the resource from the ds-server by sending the RESC ALL command. The RESC ALL basically updates all the variables from ds-server to the client-side. The value that is of interest is the *state* of the machine which indicates whether the server is available to be allocated jobs.