**Efficient Task Scheduler for Distributed Systems – Algorithm Design Document**

**Group members: (Name, Student Id)**

* Joshua Archer, 45235074
* Wei Xuan Leow, 44317484

**Introduction:**

This stage of the project is focused on designing and implementing algorithms. Once connected to the server it is dispatched a list of servers that can be used for running jobs. Once the list of servers is completely sent, the server waits on a response where the Client is then in charge of the decision making. The Client is in control of which server is most appropriate to complete the job on. Based on 3 different algorithms; first-fit, best-fit and worst-fit, the client decides which server is responsible for running this task and by doing this sends the name of the server back to the main server. From there the server schedules it.

**Design Considerations and Preliminaries:**

This project consists of one encapsulating java file which within has multiple sub-classes in charge of various things, such as connection. More specifically 2 of the classes within are bestFit() and firstFit() which are called upon when the user issues either ff and bf respectively, since there are 2 members of our group. A way of storing the server state information had to be implemented.

Before initiating the algorithms, a few steps are introduced as preliminary steps.

The system.xml file is read and saved in a ‘local’ copy on the client. Then when a job is given by the server the clients response is to ask for the server states.

When a command is typed in as an argument, the client checks whether it is “ff” or “bf” if so then the corresponding algorithms are set, and the methods are invoked.

**Algorithm description:**

* firstFit: (By Wei Xuan Leow, 44317484)

The client reads System.xml and saves the server types into an ArrayList “types”. It is ordered from smallest to largest type. A RESC command “RESC type [types[i]]” is sent to the server starting from the smallest type then using an if/else statement to search for the server with sufficient cores, memory and disk to run the job. The server also has to be idle, inactive, or active. If a server is found, the job is scheduled to the server. Else, a bigger type of server will be searched. This algorithm is the fastest out of the rest as it is only concerned with finding the first server that is available with sufficient resources to run the job.

// Emma to complete

* bestFit: (By Joshua Archer, 45235074)

bestFitServer is set to an extremely large number and so is minAvail.

For each server that appear in System.xml, an individual server is checked if it has enough resources available to run the current job. If not, then the next server in the list is checked. At the same time directly after, the server checks whether the difference in cores between server and job/how well it fits (‘fitnessValue’). If the current fitness value of the server is less than ‘bestFitServer’ or the current fitness value and the bestFitServer is identical, the minAvail times are checked. If the current server has a lower time then both variables are replaced with the current ones, else nothing happens.

This is then repeated for all servers in system.xml. Once all servers have had a chance at being more fit for the schedule, the winning one gets the current job scheduled on it.