“All To Largest” Job scheduler

Group Members: Aydin Sumer (45437009), Jonathan Bui (), Jonathan Skirtun ().

Introduction (1/2 pages): What this project (focusing on Stage 1) is about, including the goal of the project and Stage 1.

This project is a client-side job schedular with the goal to connect to a job server and schedule all jobs to the first one of the largest server type, which is the one with the highest core count in the server list.

System overview (1/2 pages): high-level description of the system (both client-side simulator and server-side simulator with the focus being your client-side simulator), preferably, with a figure (your own, not one in ds-sim User Guide) showing the workflow/working of the system.

\*insert diagram of system. Here or references? \*

Sending and receiving messages with the server:

Handshake protocol:

Gets command, getting the server list, largest server algorithm:

The system will use the GETS command to have the server send a list of server data. It will then parse this data into an array of server objects and sort in ascending order first by core count then name. It will then loop backwards from the end of the server to find the first of the largest core count server.

Scheduling and received jobs:

Design (1 page): design philosophy, considerations and constraints, functionalities of each simulator component focusing on the client-side simulator.

Implementation (2 pages): brief description of any implementation specific information including technologies, techniques, software libraries and data structures used. How each of components/functions of your simulator is implemented including who is in charge of which function(s) and how they have led the design and development.

Aydin Sumer oversees getting, storing, and sorting of server data to find the largest server. The goal of this component is to read the list of servers sent from the Gets command from the server. Then store the server’s type and core count in a class. Hold each server class in an array and sort using the arrays.sort utility in java. To do this a compare function is implemented in the server class and overridden such that it can return the compared value of the core count and the compared value of the server type if core counts are the same. So that the list of servers is sorted in ascending order of both name and core count. Then the program loops backwards from the end of the array to find the first of the largest as the list is in ascending order.

References:

https://github.com/a758/Group34ATL