Esty Nass, Batsheva Tanenbaum, Layala Kleinkaufman

Final Submission

Our load balancing program will have a user enter a task which the client socket reads and sends to the master. The master’s client- master thread then reads the task and randomly decides whether the task is of type A or type B, and then checks if the appropriate slave is available. (If it’s a type A task it will be checking if slave A is available and if it’s type B task it will check if slave B is available.) If the appropriate slave is available it will send it to the shared array list of waiting tasks. If not, it will check to see if the other slave is available and send it to its waiting tasks array list. If both are unavailable, it loops again to check to see if any slaves are available. For example, if the user enters 2 + 2 as a task and it is assigned type A, the master will check if slave A is available. If so, it is added to waitingA’s array list. If slave A is busy, master checks to see if Slave B is available. If so, it is added to waitingBTaskA’s array list. However, if both slaves are unavailable it will loop again until either slave is available.

If an A task gets added to waitingA’s array list, Master slave A thread will remove that task and send it to slave A to calculate the answer. Slave A will then send it back to master and the thread will add the results to the shared finished array list. Because it was a task of type A and slave A was available, the answer was calculated quickly. However, if slave A is unavailable the task would be sent to slave B’s method 2 which takes a longer time to calculate the results. (Slave B also has a method 1 that calculates tasks of type B quickly). Same for a B task if Slave B is available the answer is calculated quickly by its method 1 and if not its calculated slower by Slave A’s method 2.

Once the shared finished array list is not empty (contains the results) the client master response thread will read values from the list. Each task has a method getID() which tells us which client sent the request so we know where to return the answer to. The client master response thread will call finished.get(0).getID(). If it has the same ID as the current client, it will display the answer to the client and remove it from the finished array list. If it has a different ID, it will remain in the finished array list and the other client master response thread will read it.