**Cluster Computing Simulation**

**Moaz, 175888**

**A screenshot of a computer

Description automatically generated**

**Design Decisions**

**Separating Tasks in 3 array lists**

* + - It made it easier for each worker to get the tasked assigned to it.
    - But it consumed more memory
      * This could be solved by clearing the original array and save the number of tasks as integer for comparison later

**Sending Tasks as lists to workers instead of task by task**

* + - It’s the right choice as the Switch will be waiting anyway to receive all tasks form all workers as it needs to send all tasks back to the client as a single list
    - But if one worker took more time than the rest of the workers it will delay the whole thing but it’s the same if we send tasks by task as we will still wait for them all to put in one array and send to the Client

**Disadvantages**

**Worker failure**

* + - If a worker failed for some reason the whole system won’t work as the Switch code is centered around having 3 exact workers to distribute tasks.
    - But this can easily be solved by adjusting the distributing function to work with more than 3 workers or even less
      * It can distribute the tasks based on the list of workers connected, it also takes into consideration the possibility of a new worker joining or a current worker leaving while the system is working.

**A list of tasks from and to the Client**

* + - If the client were to send task by task instead of a list at a time, it would make a huge difference as the user will not have to wait for the whole tasks to finish to see the results.
    - The current approach can be useful for tasks that are somehow dependent on each other like Sorting a list and then getting the first element of the sorted list
    - But here the tasks aren’t related, and the user would prefer to get the results as fast as possible and not wait for the whole thing to finish

**Client Application**

* Starts Client Server on port 2000
* Registers Switch and connects to it at port 1099
* Receive the solved tasks and displays them
* Generates tasks
  1. Picks one of the 3 types of tasks randomly
  2. Based on the type, it generates parameters
     + PRIME || FACTORS
       - It generates a number between 1, 50
     + SUMS
       - It generates a starting number between 1, 10
       - It generates an ending number between 10, 1000
       - It generates a step between 1, 10
     + Note: all these numbers can be adjusted as long as it follows the functions rules
* Add the task to the tasks array list with the id attached to it
* Send the tasks array list to the **Switch**

**Switch Application**

* Connects to the client
* Starts Switch Server
* Gets tasks from Client
* Distribute tasks
  1. Makes a separate array list for each worker
  2. Uses round-robin to distribute the tasks to each array list
* Gets Results from Workers
  1. Gets the results from the worker
  2. Append the results to the solved array list
  3. Check if the size of the array list is qual to the size of tasks array list
  4. If they are equal, it sends the solved array list back to client
* Send tasks to workers based on their ids

**Worker Application**

* Connects to the switch
* Starts Worker Server
* Gets the tasks from the switch
* Solve tasks
  1. Looping over tasks
  2. Checking the task type
  3. Call the correct function to solve it with the correct parameters
* Send Solved tasks to the switch