**Distributed Systems Job Scheduler – Stage 2**

**Group Members**

Connor Wilson – 44946098

Peter Taylor – 45152969

Bradley Anderson – 45229678

**Introduction**

Stage two is about creating algorithms that allow the jobs to be allocated into different servers based on a certain functionality. The functionalities we have been given are First Fit, Best Fit and Worst-fit. The first fit algorithm will allocate the relative job to the first available server which has the correct number of cores, if this fails then the job will be queued for the first unavailable server that fits the core requirements. The Worst fit algorithm will allocate the job to the server with the most cores and is available. If none are available, then the job will be queued to the largest server. The first fit algorithm will allocate the job to the first server with the correct number of cores despite its availability.

**Design Considerations and Structure**

The common data structures used in each implementation are:

**ArrayLists**: Used to store the available servers that can have jobs scheduled to them. An Arraylist is used to be able to easily remove and update the list of servers, storing them as a string. For each string in the list, these will be broken down into an array of strings so that it’s possible to select individual bits of info for a server, such as core count, server size, server ID, etc.

There’s only one defined ArrayList that is used across all 3 implementations of the scheduling algorithm, that being serverList which is an array of strings that’s used to store the singular string that represents server info that’s been broken down into separate parts to be stored in the array. To fetch this list, each time a new job comes in, its required CPU cores, memory and disk requirements are used to call RESC Avail which prompts the server to return all available servers that meet the job requirements. One by one these servers are pushed onto serverList like so:

serverList 🡪 serverList(0) 🡪 small = serverList.get(0)[0]

0 = serverList.get(0)[1]

0 = serverList.get(0)[2]

270 = serverList.get(0)[3]

2 = serverList.get(0)[4]

4000 = serverList.get(0)[5]

16000 = serverList.get(0)[6]

In all three implementations, serverList is used globally to hold the list of servers pulled from the current server configuration.

In worst fit serverList gets searched through to find the server with the largest avaliable core count and pulls that out to be placed in BFFServer, an array that stores the best suited server for the job.

In first fit the last server in the list is grabbed and used as the best suited server, since first fit is the first available server with the bare minimum number of cores to handle the job, due to the fact we push servers onto serverList, the smallest servers should end up at the very end of the list.

In best fit, the list is searched to find the server with the smallest number of cores that still meets the requirements of the job being

**Arrays**: Arrays in this program are used across all 3 implementations to both store the jobs given to us from the server and the selected server chosen via the scheduling algorithm. Both arrays are arrays of strings to break up the details of the job or server and store them and that individual details of both jobs and servers can be used to make a call to RESC Avail or SCHD to schedule jobs.

There are two defined arrays, inputArrayString and BFFServer. Both arrays of strings are used to hold different things, inputArrayString holds the job that is sent by the server as an array of strings like how serverList stores each entry in its list. How these are stored is as such:

inputArrayString 🡪 JOBN = inputArrayString[0]

210 = inputArrayString[1]

0 = inputArrayString[2]

385 = inputArrayString[3]

1 = inputArrayString[4]

100 = inputArrayString[5]

800 = inputArrayString[6]

inputArrayString is used to call openServers which depending on scheduling algorithm used and passes it elements 4, 5 & 6 in the array so it can return the list of relevant servers that can handle the job in question. inputArrayString doesn’t change across the 3 implementations because the jobs remain the same regardless of scheduling algorithm.

BFFServer holds the selected server for the current job that is to be scheduled. It’s changed every time a new job comes in, since the openServers function is called to find the optimal server to assign the current job to. It’s an array of strings that is stored the same way as an entry in serverList is due to it being an array of strings representing an open server.

In worst fit Implementation BFFServer becomes the server with the largest number of available cores based on the search.

In first fit, BFFServer becomes the last server in the list of available servers, since this is ordered from largest to smallest and first fit requires smallest to largest.

In best fit, BFFServer becomes the server with the smallest number of cores still capable of completing the job.

There are a few other arrays used in each scheduling algorithm like max, min and ser which are temp arrays to hold the temporary best server while searches are still conducted however they are within functions such as bestFit, worstFit and firstFit

**Strings:** Strings are used throughout all scheduling algorithms to send the server messages using the .write command. There is only one string that really changes depending on scheduling algorithm and that is the input to the main function, this being the String type. Type is a string that is input as a command line argument to start the program running. When running the Client class using the command ‘java Client -a ff/wf/bf’ the final part of the argument influences what Type ends up being. Type is checked when openServer is called to determine what scheduling algorithm to use.

If the 2nd argument after -a is ff, first fit is used as the scheduling algorithm, so Type becomes ff.

If it’s wf, worst fit is used as the scheduling algorithm, Type becomes wf.

If it’s bf, best fit is used as the scheduling algorithm, Type becomes bf.

**Worst Fit Algorithm - Connor Wilson**

For the worst fit algorithm, we made a max variable and set it to the first server Then we loop through every server in our serverList variable. If the current item equals “.” Or “ “ then we skip that item. Knowing that the only items in the list now are servers, we check the 4th index of both the max and current servers for the core count, if the current is larger than the max core count we set the max as the current server. After doing this for the entire list, we return the worst fit server, which gets set in the schedule function

**Best Fit Algorithm – Bradley Anderson**

The client firstly requests the next job that needs to be scheduled, then the string representing the job is put into the inputArrayString array and used as an input for the openServer function to find all available servers. This list of available servers is stored in serverList, ready to be searched for the optimal server to schedule the job to. The best fit algorithm operates similarly to the worst fit algorithm, however rather than searching the list for the largest number of cores, it will check the list for the smallest number of cores that’s still able to handle the job’s core requirements. This is done using the bestFit function which will set up BFFServer with the optimal server to schedule the selected job to. Once the server that best fits the job is found, the job ID is pulled from inputArrayString[2] and the server type and ID pulled from BFFServer[0] & BFFServer[1] and used in the SCHD command to the server to schedule the job.

**First Fit Algorithm – Peter Taylor**

The first fit algorithm is the easiest of the 3 algorithms. This algorithm tells you to take the first server that you find that can run the process and send the job to that server. This, while it may not be the most efficient method at filling the servers it reduces the delay between a job being received and a job being sent to a server. In our code because we read in the servers in order all we need to do with them is to read the one that got stored first. The only trick that we need to do is to access the last element in the array list because we have been adding the newest server to the front of the list. This server is the first server we got and is the one we will send the job to.

**References**

Repository Link: https://github.com/Bradleyanderon/Comp\_335-Job\_Scheduler.git