Design and implementation of the Descending First Fit algorithm

Jonathan Cawood 45887454

Introduction (1/2 page)

This project is centrally based upon the understanding of job scheduling and algorithms of job scheduling which through the utilization of algorithms can control in which manner the server scheduled jobs and therefore will have differing effect on the productivity of the server. With successful completion of the project the development and implementation of a new scheduling algorithm will outperform one or more of the current traditional algorithms which are, First Fit, Best Fit and Worst fit. The way in which the new algorithm will out preform the traditional algorithms is to be determined with the use of measures. These measures can be quantitative metrics such as turnaround time, resource utilisation, and rental cost, additionally a qualitative analysis of the advantages and disadvantages will be assessed in determining the overall performance of the newly designed algorithm.

Problem definition (1/2 page)

The designed algorithm is derived from the traditional First Fit algorithm, which takes a list of servers which are sorted in ascending order based on the capacity of the server, and scans through the list of servers and the first server that can take the job and has no waiting jobs, the scheduling decision is made that that server is the First Fit server to take the job. The newly designed algorithm has similar concept to the traditionally First Fit algorithm however the key difference is that fact that the server list is reversed and therefore in descending order regarding capacity, this will affect the process by there being a great bias towards the largest servers. Therefore, this alteration will result in a lowered turnaround time as the algorithm will be able to quickly assign jobs to the largest server, however this will impact the rental cost as the larger servers are more expensive, additionally the utilisation of the server will not be as productive as possible as the smaller servers will no be used, and much of the work will be to the largest servers. The justification as to why this approach was taken is to try and optimise the time the algorithm is searching for the right server, with the idea being that most of the jobs will be sent to the largest server, this will be the first server compared in the server list, therefore reducing turnaround time.

Algorithm description (1 page)

**Descending First Fit (DFF)**

**Input:** A String Array *job*, a Server Array *serverlist*, serverlist is in ascending order based on core count.

**Output:** A server in which the job is to be scheduled on.

**Process:** reverse the given serverlist, recognise the corecount needed from the job, search in the new server list, with each server search check satisfaction with server state being idle, if so then check capacity is greater than or equal to the corecount needed. The first instance these conditions are meet record the server values, then output these the desired server.

The purpose of Descending First Fit is to maximize the turnaround time, by having the available servers in descending order limits the time wasted on checking smaller servers, intern having an increased likelihood of jobs being scheduled to the largest servers.

With relation to ds-config01—wk9.xml the servers “tiny”, “small” and “medium” once passed to the algorithm will be arranged “medium”, “small”, “tiny”. Then checks the corecount of the first job. Then scans the serverlist, firstly checks if the first server is idle, if so, checks the corecount and if the server corecount is larger or equal to the job is scheduled to the server. If the server is not idle or does not have the required corecount then the next server is analysed. In the event there is no server capable of running the job, either no idle servers or corecount is not large enough the job is skipped, as the outputted server will be empty.

**Worked example.**

With “ds-config01—wk9.xml” having job types “short” requiring 1 core, “medium” requiring 2 cores and “long” requiring 3 cores, taking into consideration their population Rates we can have 2 short jobs, 2 medium jobs and 1 long job. Additionally, having servers “tiny” with core count of 1, “small” with a core count of 2, and “medium” with a core count of 4. We can detail the process in which the algorithm goes through in the scheduling decision. Let’s assume that all servers are idle and waiting for a scheduled job.

The algorithm is passed a server array in ascending order being “tiny, small, medium”, within the algorithm this is reverse and is stored as “medium, small, tiny”. the algorithm is also passed the first job sent by the server which is a “long” job, from the job the core count is found to be 3 to be used in the comparison. the next step is a for loop the size of the server list which sequentially goes through the list check the compatibility for each server, the first field is check is the availability of the server where if the server state is “idle” meaning not currently on a job and is running then the next condition is checked, if not the next server is analysed. the first server is the medium server which is idle, so this condition is met, the next condition to be met is the core count, and the medium server has a capacity of 4 and the long requiring 3 this is condition is met. As these conditions are all meet the algorithm is able to return the server to the call, to then be able to schedule, this process is repeated for all remaining jobs, however the medium server is longer idle therefore will not meet conditions of the algorithm so will be skipped.

Implementation (1/2 page)

In the implementation of the newly developed scheduling algorithm, data structures were used in assistance, two classes were defined, a server class to be able to store information of servers, and a job class to be able to store the right information about to be scheduled jobs, both these classes have getters and setter functions enabling the setting of variables and the retrieval of values. Additionally, the function called Descending First Fit (DFF) for the new algorithm was created to be called enabling modularisation, within the function was the process in which the new algorithm would function and the actions in which would produce a more efficient process meeting guidelines for this task. DFF was called upon the detection of a job to be scheduled, with the output used to create a SCHD string to then be sent to the server. Helper functions were created to enable pure modularisation of coding and therefore allow for ease of understanding with reviews taken into consideration. These helper functions consist of a readMSG function which taken the input stream passed through a buffered reader and converts the bytes into a String using the readLine() command, a sendMSG function which when passed a String converts the string to bytes and flushes the message to the server. Additionally, a parsing function which when parsed a string will return an array of strings which is the original string delimited by a blank space, this allowed the breaking of received messaged from the server, allowing for extraction of variables.

Evaluation (2 pages)

With analysis of the traditional algorithms FF, BF, WF logs were created from the ds-server communication when running the simulation with the three different algorithms to give a comparison to the new designed Descending First Fit (DFF). For the baseline test the config file “ds-config01—wk9.xml” was utilised when testing all algorithms. As the intended implementation of Descending first fit algorithm was specifically designed to lower turn around time, this is the focus of evaluation.

Regarding the average turnaround time, which is calculated upon the completion of scheduling all jobs. The mathematical process in which average turnaround is calculated is the starting time subtracted from the completion time, or alternatively the sum of the waiting time and execution time, this gives the total time taken which is then divided by the number of jobs to get an average turnaround time.

**Pros and Cons**

In evaluation of the DFF algorithm the positive attributes of the newly designed algorithm is that it will improve the turn around time of scheduling, this is because of the implemented biased towards the largest servers with the largest servers being frequented first, eliminating the time wasted in checked the capacity of smaller servers, the little time wasted checking the capacity of the smaller servers is compounded over all the jobs and could sum to a fairly large total with multiple jobs to be executed.

The DFF algorithm will encounter limitations with regards to its ability to process multiple larger sized jobs, the order in which the jobs is scheduled will have an impact on the performance and therefore the turnaround time of the DFF algorithm, as if a moderate sized job is to be scheduled first, this will take the largest server possible, then if a larger then moderate sized job is to be scheduled which requires the largest server this will cause longer waiting times and therefore negatively impact the turnaround times. This can be avoided with other algorithms as the largest server is not utilised allowing the larger than moderate job to be scheduled efficiently on the largest server.

Conclusion (1/4 page)

In conclusion, the designing and development of the Descending First Fit (DFF) algorithm, has been somewhat unsuccessful, as the implementation of the algorithm was not successful however the designed and logical process behinds the process is sound, and with the theoretical analysis and reasoning are sound in being able to achieve the requirements and is believed that the correct implementation would lead to visible results.

References (1/4 page)

Cawood. J. Comp3100-Stage2-45887454 [online]. GitHub. Https://github.com/jonathancawood/Comp3100-Stage2-45887454

Lee. Y. (2021). ds-sim[online]. GitHub . Available at: [https://github.com/distsys-MQ/ds-sim/]