Bilkent University CS 342 Project 3 Report

BILKENT UNİVERSİTY

EGE TURAN – 21502441

CS 342 – SEC1

FALL-2019

Table of Contents

1. Experiment Data

1.a

Experiment is made using ctime and values are in milliseconds.

Number of Allocations	First-Fit	Best-Fit	Worst-Fit
100	0.325	0.48	0.62
1000	17.2	23.41	28
10000	3244	3678	4734

1.b

Same experiment using 2 threads.

Number of Allocations	First-Fit	Best-Fit	Worst-Fit
100	0.294	0.37	0.57
1000	16.8	22.65	25
10000	2899	3423	4454

2. Explanations and Observations

We have used Virtual Machine while testing and implementing the project. Virtual machine uses multi-processing while computing. Our virtual machines use 5 CPU to compute. This is was effective when we were using multithreads for our experiments. Thanks to multithreading and using multiple CPUs, we take more synchronized run times for our experiments.

2.a First Fit

In our project we used three different approach (first, best, worst) so their running time differ because of their working ways. In order to find the desired memory block, we need to loop through our emptyList(we used linked list for that). As we expected, the fastest algorithm was First-fit approach because can use "break;" statement to quit from while loop so that we don’t need to check other empty places. Quitting loop when we find first place to allocate, provide less running time. On the other hand, we realized that there is trade-off between running time & memory usage in first fit approach because most of the time, takes much more memory than requested amount. It can take 10000 bytes of memory block even if request is just 20 bytes.

2.b Best Fit

Best fit approach takes much more time as compared with first-fit approach. The reason is that we need to loop through all emptyList to find best match. We determined the most proper memory block with lowest internal fragmentation that means reducing wasted space within each allocated block. We compared every (holeSize - objectSize + sizeof(struct hole)) to find minimum. Even if best-fit takes considerable amount of time, it can provide much more efficient memory allocation.

2.c Worst-Fit

Finally, worst-fit approach -as its name indicates- was the worst algorithm we used because it is searching all empty list and allocating the maximum block size from that list. It's running time is most of the time same with best-fit because both of them searching all the list. Unfortunately, worst-fit approach is not as fast as First-Fit approach nor as effective as Best-Fit approach with respect to memory allocation.