Data Structures and Algorithms

Eric Osborne

MP4

Fall 2024

Analysis of Methods of Memory Allocation

Abstract

For this assignment we were tasked with designing and implementing several different methods of memory allocation. These functions were created to imitate the function of malloc and free. There were several different methods employed including coalescing and the fitting types of best-fit and first-fit. Combinations of each were tested to demonstrate their efficiency.

Coalescing

Coalescing involves combining blocks that are placed sequentially next to each other. This is used to decrease fragmentation on the stack which allows the software to utilize space more efficiently. Otherwise, we tend to be left with small blocks that cannot be used for large data storage.

Best-fit and first-fit

As the name implies first fit selects and splices the first available chunk of memory. While this may seem at face value to be less computationally expensive the data shows that it is better to find a block that is closest to the size which we need to access. This is how best-fit was implemented to find a block of data that is as close to ours as possible. This reduces needless fragmentation.

Testing Plan

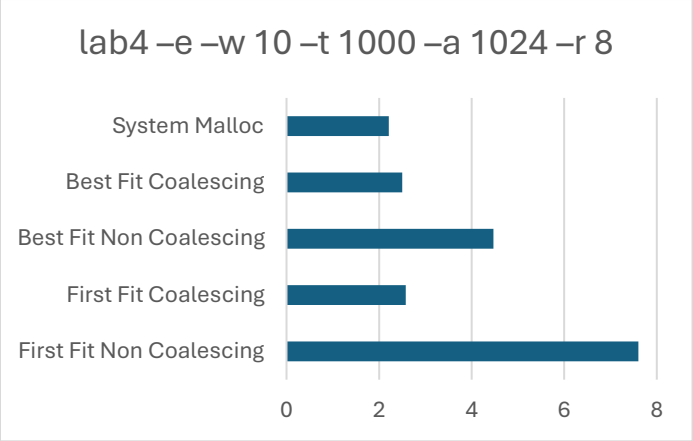
The testing plan for this implemented several unit drivers to test for edge cases and effectiveness of the data. While there are several different unit drivers already created the first unit driver I constructed, unit driver 3, did the following.

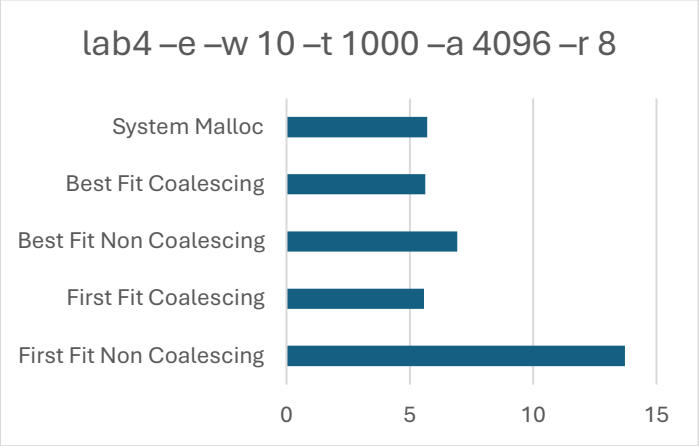
Unit driver 3 performs testing on large scale data sets. This includes several large arrays. Each array is written to and then freed. As with all unit drivers a mem_print() and a mem_stat() is called in between mem_alloc and mem_free to show the state of the free list and current memory allocation.

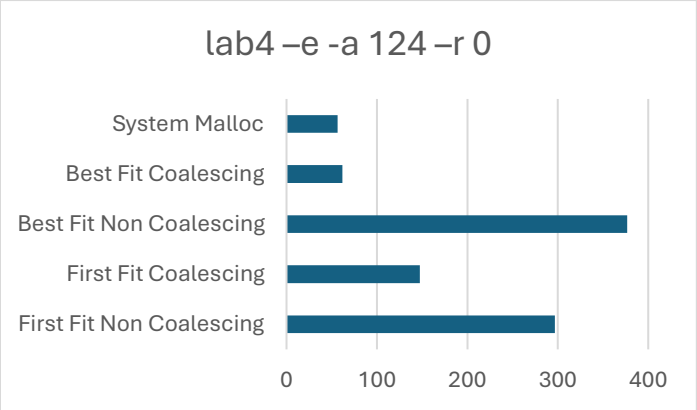
Unit driver 4 starts by allocating space for three different array types. It then writes values to each of the arrays, and then frees them. The arrays are then reallocated in the same order that they were allocated in the beginning. In doing so morecore should only be called in the first half which means that each fitting type should take an entire block from the free list instead of requesting memory from morecore.

Unit driver 5 tests a variety of different data types with malloc. This includes integers, floats, chars, floats, and structs.

Overall Results







Given the following results the combination outside of the system malloc that seems to have performed best is best-fit with coalescing. This was expected. As discussed in the overview of best-fit, best-fit finds the ideal block that matches ours. Given a large-scale data set it is not uncommon to find an exact match if we look long enough. This provides

the advantage of decreasing fragmentation. By far the worst combination was first-fit non-coalescing. Given these two approaches there is an extreme amount of fragmentation in the free list. This leads to longer roving times and less efficient use of memory. The one thing to take note of is that first fit with coalescing does somewhat lag behind the efficiency of best-fit coalescing with smaller data sets. However, as the size of the data set increases the efficiency of first-fit decreases leading to longer processing times due to greater fragmentation. The system malloc was of course the fastest in all cases as expected. Given its close processing time with best-fit coalescing it can be implied that malloc uses some form of best fit and coalescing which has some optimizations applied to it.