Michael Morris

ECE 2230-1

MP4 Performance Analysis

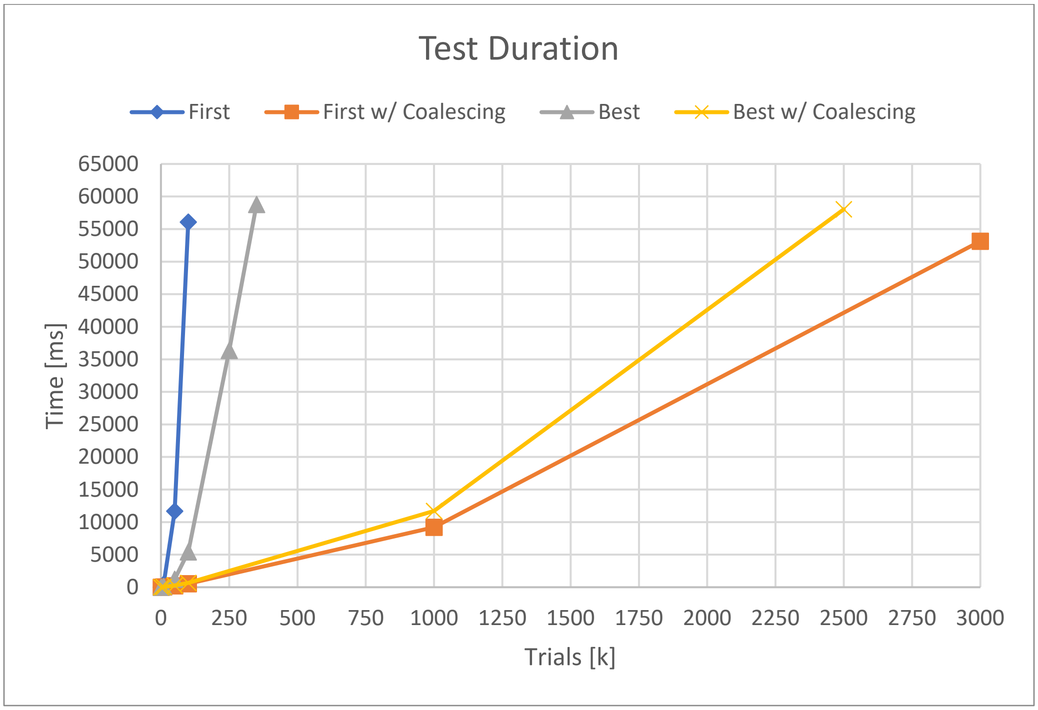
10/26/2023

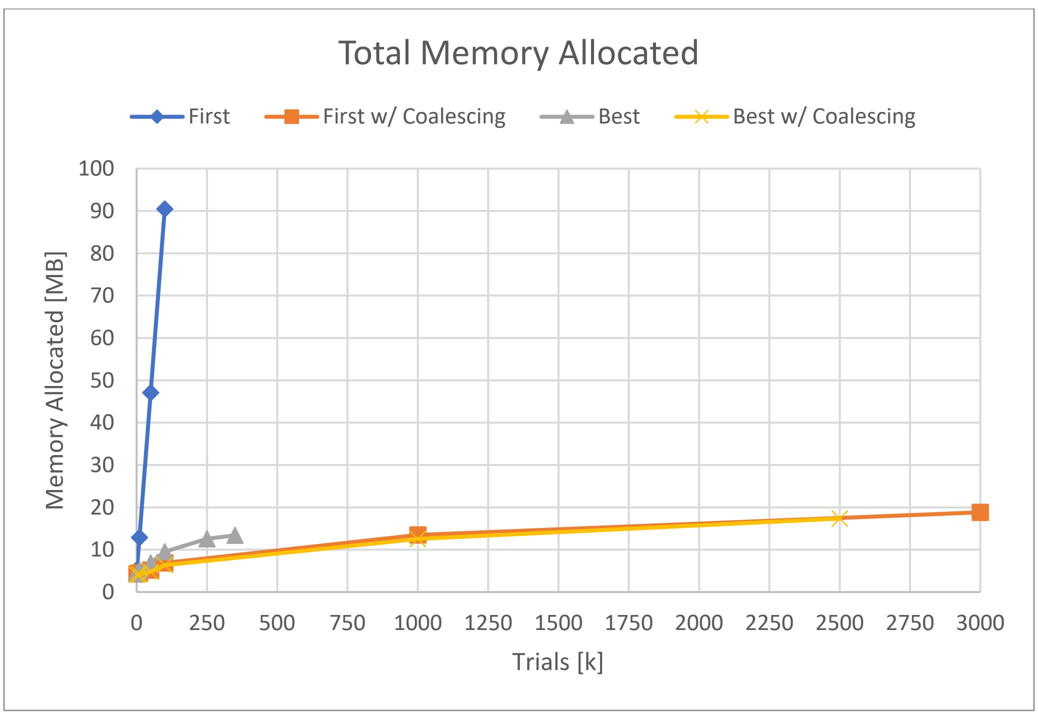
In MP4 we used a one-way circular linked list to hold the contents of our free memory data. This was simple to implement but may have yielded slightly lower performance compared to a two-way linked list. In order to allocate memory in the best-fit manner, each free memory block needs to be evaluated and compared to the other blocks to determine which size is actually the best. If a block has the exact amount of space required for the allocation, the search is stopped and the roving memory pointer points at the correct spot. If the exact free space is not found, then the rover pointer cycles through list until the correct block is selected. Next, a temporary memory pointer is created to cycle through the free memory again until the memory block just before the selected block is found. By doing this, I was able to adjust the next pointers and stored memory block units to accurately show the free space available after the memory block needed was removed. With a two-way linked list, this would have been accomplished by simply moving the new temporary rover back one block and would have saved time.

Using a two-way linked list would have required an additional pointer for the previous memory block to be stored in each free block, however. This would have taken up more space in the header and would have left less free space available for the user. This tradeoff might not be noticeable in simpler programs that don't require many memory allocations, but programs that use many thousands of allocations would see a large increase in memory consumption. This free space / performance tradeoff needs to be addressed an a per-use basis when implementing either a one or two-way linked list for this purpose.

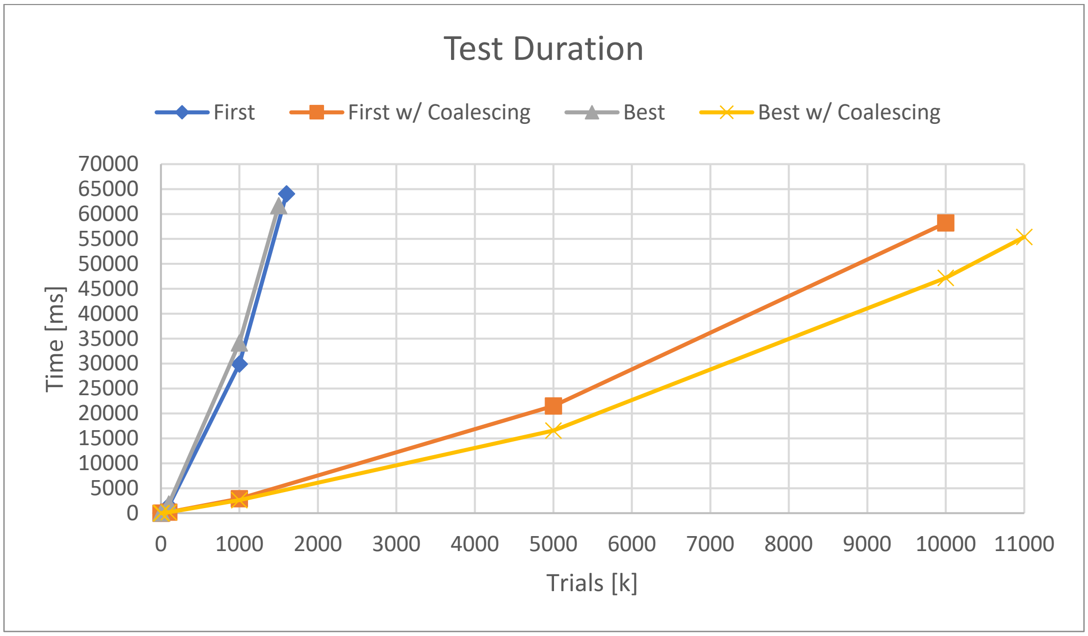
As is seen in the two graphs below, there were noticeable differences between the first and best-fit methods used in this MP. Comparisons between the first-fit block finding and best-fit methods shows that the non-coalescing first-fit method performed worse in both speed and total memory usage. This is because the first-fit method inevitably creates many small blocks which later can only be used by equal or smaller memory allocations. The best-fit method performed approximately 10 times better than first-fit when running 100k trials – best-fit finished in 5437 ms while first-fit finished in 56089 ms. Although best-fit scans each member of the free list (worst case scenario) if it finds a block that is exactly the size as was requested the loop breaks and the program continues. This, combined with the fact that it produces less fragmentation overall, lead to both an increase in speed and total memory usage. Interestingly, the best-fit method memory usage appears asymptotic. This shows that after a certain number of trials, a stable state is achieved where the probability that there is an exact fit for a new allocation request approaches 100%. This is most likely due to the nature of the Equilibrium driver's maximum array size and average range parameters combined with the equal probability that it will allocate new memory or free existing memory.

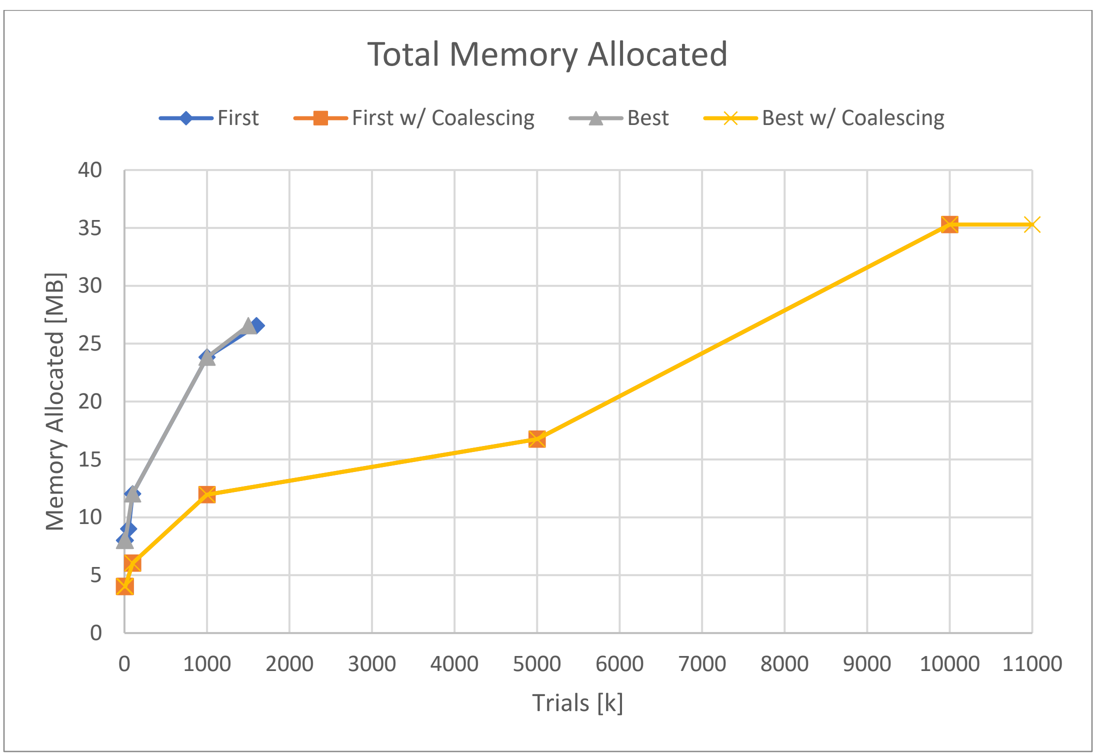
Both first-fit and best-fit methods are drastically improved when using coalescence compared to their non-coalescing counterparts. As memory is freed back into the free list, adjacent memory blocks are combined into a single block which allows for significant reduction in total memory used as well as overall speed. When this option is enabled, the first-fit method performed better overall as it did not have to assess each free list member and simply took the first available free memory block of adequate size. Compared to non-coalescing methods, these coalescing search functions were able to perform roughly 10 times as many trials in ~60 seconds and exhibited an asymptotic relationship between the total memory used and total trial number.



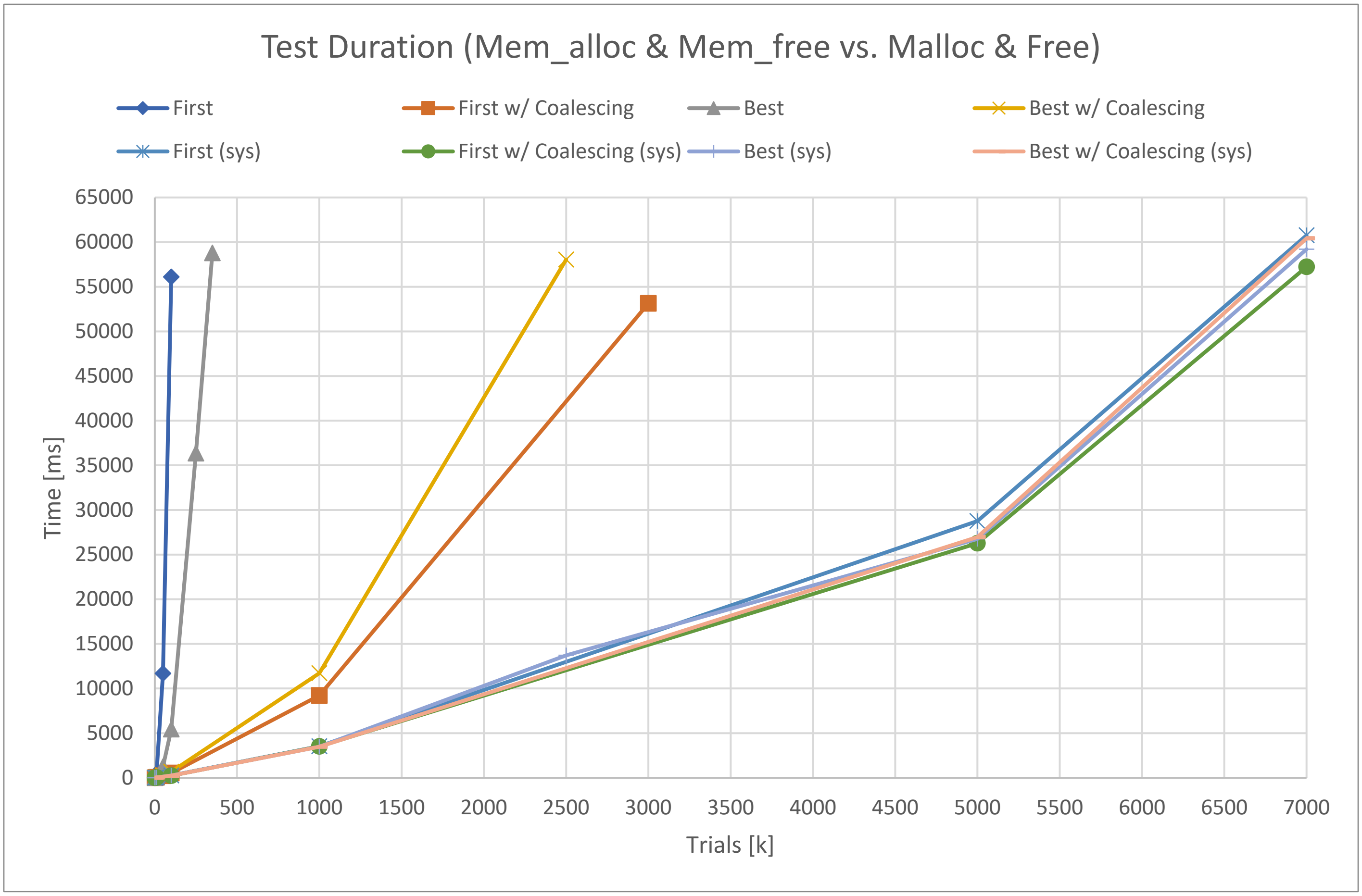


The results below show that when using allocations that are always the same size (-r 0), performance for all fit methods improves considerably. This is because each the search function will have to travel few blocks to find a suitably sized block, and less pages will have to be requested overall compared to tests with a nonzero range.





Testing my Mem_alloc() and Mem_free() functions against the system Malloc() and Free() functions resulted in the graph below. It is obvious from this that the system functions are significantly more efficient than my new functions.



Data:

			Warmups	Avg array size	Range					
			1000	1024	255					
	Trials (thousand)	Time	Sbrk Calls	Pages Allocated	Blocks	Total Memory (bytes)	Total Memory (MB)	Min	Max	Avg
First	1	6.937	744	1354	1180	5545984	5.2890625	16	5136	2540.53
First	10	440.527	1714	3293	7441	13488128	12.86328125	16	5136	1812.92
First	50	11683.4	6094	12051	31039	49360896	47.07421875	16	5136	1590.34
First	100	56089.6	11651	23162	60584	94871552	90.4765625	16	5136	1565.98
First w/ Coalescing	1	3.233	713	1106	2	4530176	4.3203125	4530160	4530160	4530160
First w/ Coalescing	10	44.495	662	1004	2	4743168	4.5234375	4743152	4743152	4743152
First w/ Coalescing	50	227.595	822	1324	2	5423104	5.171875	5423088	5423088	5423088
First w/ Coalescing	100	536.476	1038	1756	2	7192576	6.859375	7192560	7192560	7192560
First w/ Coalescing	1000	9219.6	1885	3450	2	14131200	13.4765625	14131184	14131184	14131184
First w/ Coalescing	3000	53136.8	2570	4820	3	19742720	18.828125	262144	19480560	9871352
Best	1	5.746	603	1128	1834	4620288	4.40625	16	5136	2520.61
Best	10	104.447	698	1318	3913	5398528	5.1484375	16	5136	1379.99
Best	50	1233.76	921	1764	9800	7225344	6.890625	16	5136	737.35
Best	100	5437.01	1257	2436	16310	9977856	9.515625	16	5136	562.86
Best	250	36358.1	1647	3216	29547	13172736	12.5625	16	5136	445.84
Best	350	58787.7	1754	3430	34426	14049280	13.3984375	16	5136	408.11
Best w/ Coalescing	1	3.162	687	1054	2	4317184	4.1171875	4317168	4317168	4317168
Best w/ Coalescing	10	50.685	690	1060	2	4341760	4.140625	4341744	4341744	4341744
Best w/ Coalescing	50	279.595	779	1223	2	5009408	4.77734375	5009392	5009392	5009392
Best w/ Coalescing	100	665.318	1005	1629	2	6672384	6.36328125	6672368	6672368	6672368
Best w/ Coalescing	1000	11709.9	1865	3208	2	13139968	12.53125	13139952	13139952	13139952
Best w/ Coalescing	2500	58037.1	2542	4442	3	18194432	17.3515625	217088	17977328	9097208

			Warmups	Avg array size	Range					
			1000	1024	0					
	Trials (thousand)	Time	Sbrk Calls	Pages Allocated	Blocks	Total Memory (bytes)	Total Memory (MB)	Min	Max	Avg
First	1	15.368	1025	2049	2050	8392704	8.00390625	4080	4112	4095.99
First	10	83.311	1025	2049	1132	8392704	8.00390625	4080	4112	4095.99
First	50	502.767	1153	2305	2306	9441280	9.00390625	4080	4112	4095.99
First	100	1460.2	1539	3077	3078	12603392	12.01953125	4080	4112	4095.99
First	1000	29900.1	3050	6099	6100	24981504	23.82421875	4080	4112	4096
First	1600	64050.4	3399	6797	6798	27840512	26.55078125	4080	4112	4096
First w/ Coalescing	1	1.818	515	1029	2	4214784	4.01953125	4214768	4214768	4214768
First w/ Coalescing	10	18.892	515	1029	2	4214784	4.01953125	4214768	4214768	4214768
First w/ Coalescing	100	201.915	773	1545	2	6328320	6.03515625	6328304	6328304	6328304
First w/ Coalescing	1000	2902.68	1531	3061	2	12537856	11.95703125	12537840	12537840	12537840
First w/ Coalescing	5000	21504.4	2145	4289	3	17567744	16.75390625	417792	17149936	8783864
First w/ Coalescing	10000	58215.3	4518	9035	4	37007360	35.29296875	1966080	17891328	12335781
Best	1	9.138	1025	2049	2050	8392704	8.00390625	4080	4112	4095.99
Best	10	132.244	1025	2049	2050	8392704	8.00390625	4080	4112	4095.99
Best	100	1833.3	1539	3077	3078	12603392	12.01953125	4080	4112	4095.99
Best	1000	34125.4	3050	6099	6100	24981504	23.82421875	4080	4112	4096
Best	1500	61685.8	3399	6797	6798	27840512	26.55078125	4080	4112	4096
Best w/ Coalescing	1	1.844	515	1029	2	4214784	4.01953125	4214768	4214768	4214768
Best w/ Coalescing	10	16.479	515	1029	2	4214784	4.01953125	4214768	4214768	4214768
Best w/ Coalescing	100	189.093	773	1545	2	6328320	6.03515625	6328304	6328304	6328304
Best w/ Coalescing	1000	2615.1	1531	3061	2	12537856	11.95703125	12537840	12537840	12537840
Best w/ Coalescing	5000	16568.3	2145	4289	3	17567744	16.75390625	417792	17149936	8783864
Best w/ Coalescing	10000	47192.6	4518	9035	4	37007360	35.29296875	1966080	17891328	12335781
Best w/ Coalescing	11000	55391.8	4518	9035	4	37007360	35.29296875	1966080	17891328	12335781

malloc and free		
Warmups	Avg array size	Range
1000	1024	255
	Trials (thousand)	Time
First	1	2.767
First	10	24.772
First	100	248.523
First	1000	3530.07
First	5000	28746.7
First	7000	60758.5
First w/ Coalescing	1	2.724
First w/ Coalescing	10	23.962
First w/ Coalescing	100	242.403
First w/ Coalescing	1000	3519.41
First w/ Coalescing	5000	26273.2
First w/ Coalescing	7000	57225
Best	1	2.924
Best	10	23.067
Best	100	255.807
Best	1000	3481.57
Best	2500	13703.5
Best	5000	26745.8
Best	7000	59202
Best w/ Coalescing	1	2.967
Best w/ Coalescing	10	22.828
Best w/ Coalescing	100	244.611
Best w/ Coalescing	1000	3473.3
Best w/ Coalescing	5000	26952.9
Best w/ Coalescing	7000	60416.8