Bilkent University
Computer Engineering
CS 342 – Operating Systems
Fall 2016

Project 2

Gülsüm Güdükbay (21401148)

Assigned: October 21, 2016
Due Date: November 5, 2016

For this project, in order to test and analyze the benefits of multithreading, several inputs types, numbers of inputs and numbers of threads were selected. The well-known hypothesis in using multiple threads is that as thread number increases, execution time decreases. However this is not the case in real life. There is a significant memory allocation and deallocation bottleneck in multithreaded programs. This bottleneck is because of the fact that the allocator has to search hard for larger heaps in order to allocate the continuous regions that are requested. Besides the memory allocation and deallocation bottleneck, another topic that affects the performance of a multithreaded program is the context switching payoff. Because of the fact that all of the priorities of threads are the same, the scheduling done by the operating system is typically like round robin scheduling. Each thread has a short turn and after that turn is ended, another waiting thread is executed. This creates a problem, because each time when a context switch happens, the register states are saved so this affects the performance. Also, saving a thread's cache state, which can consume a large memory, causes another significant overhead. That is why, since the caches are small, when they are full, eviction happens and when several threads do this, performance is affected negatively.

To demonstrate the above information, two inputs from the sample input folder given by the teaching assistant were used: sample 2 and sample 7. For sample 2, for one input file, several numbers of threads were experimented. For sample 7, several different numbers of input files along with several different numbers of threads were experimented.

Number of Threads	Execution Time (s)
5	0.019357
10	0.017014
15	0.016758
20	0.020430
25	0.016328
30	0.017478
35	0.019319
40	0.015020
45	0.020467
50	0.057357

Figure 2 – Table of Execution Time for Several Numbers of Threads

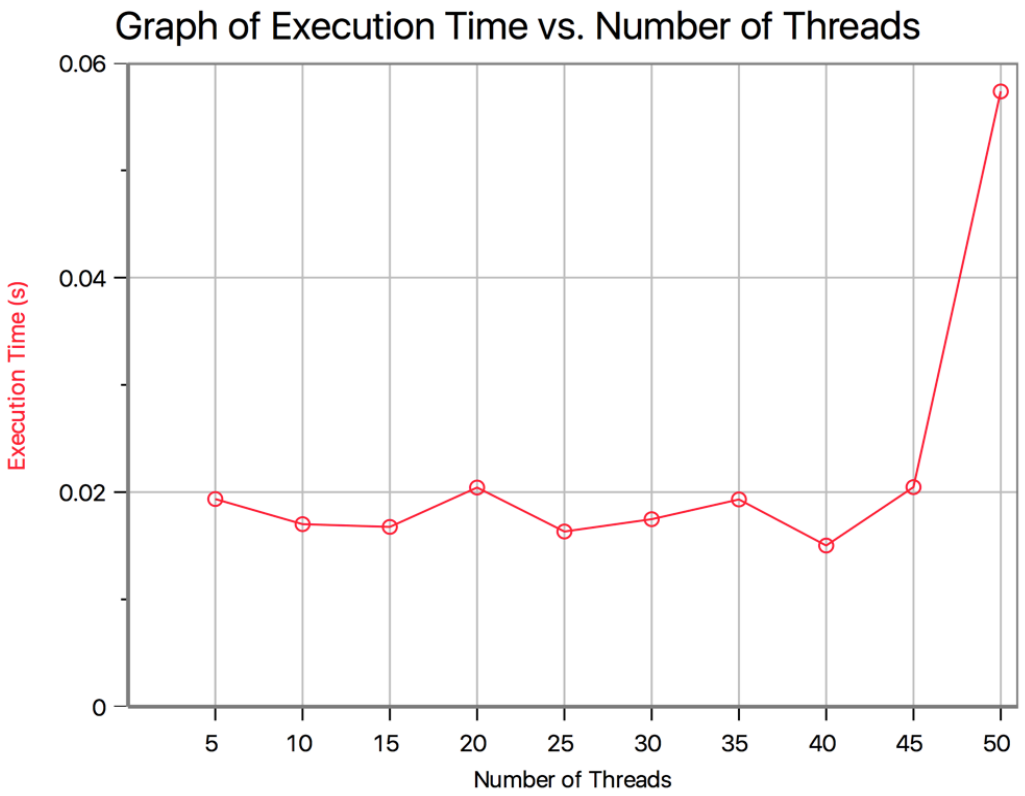

Figure 1 – Graph of Execution Time vs. Numbers of Threads for Sample 2

As seen in the graphs, for one input file the execution times are almost the same, excluding the anomaly in 50 threads. The anomaly is almost five times the other execution times, showing the bottleneck of creating a lot of threads. The optimum performance is seen with 40 threads and 25 threads. Better results are seen in the next example.

Thr	In	1	2	3	4	5	6	7	8	9	10
1		0.058016	0.053048	0.093812	0.083964	0.116597	0.099405	0.107622	0.171562	0.135899	0.181914
5		0.053561	0.054528	0.096638	0.082700	0.103025	0.084815	0.089049	0.116052	0.086122	0.090475
10		0.056035	0.072916	0.116565	0.078556	0.098168	0.094018	0.092933	0.153262	0.095544	0.093737
20		0.066302	0.076368	0.123491	0.089446	0.129784	0.103568	0.097862	0.150606	0.114544	0.114132
30		0.070325	0.078578	0.103661	0.072114	0.123678	0.110172	0.112990	0.108669	0.127810	0.093157
40		0.063196	0.071286	0.116744	0.079281	0.119429	0.075937	0.114386	0.063381	0.120691	0.036397
50		0.028534	0.024724	0.053419	0.064823	0.049602	0.045199	0.042473	0.059312	0.065313	0.057089

Figure 3 – Table of Execution Times for Each Number Of Input and Number of Threads for 7th Sample Input File

The table above represents the execution times for 10 different input file numbers given to the program, along with the 7 different numbers of threads, in seconds. As seen above, the execution times a specific number of files with different number of threads are not linear, as expected. This is mainly because of the payoff of creation of threads. For each number of files, there is an optimum value of execution time for a specific thread number. This is seen much better in the graph in the next page.

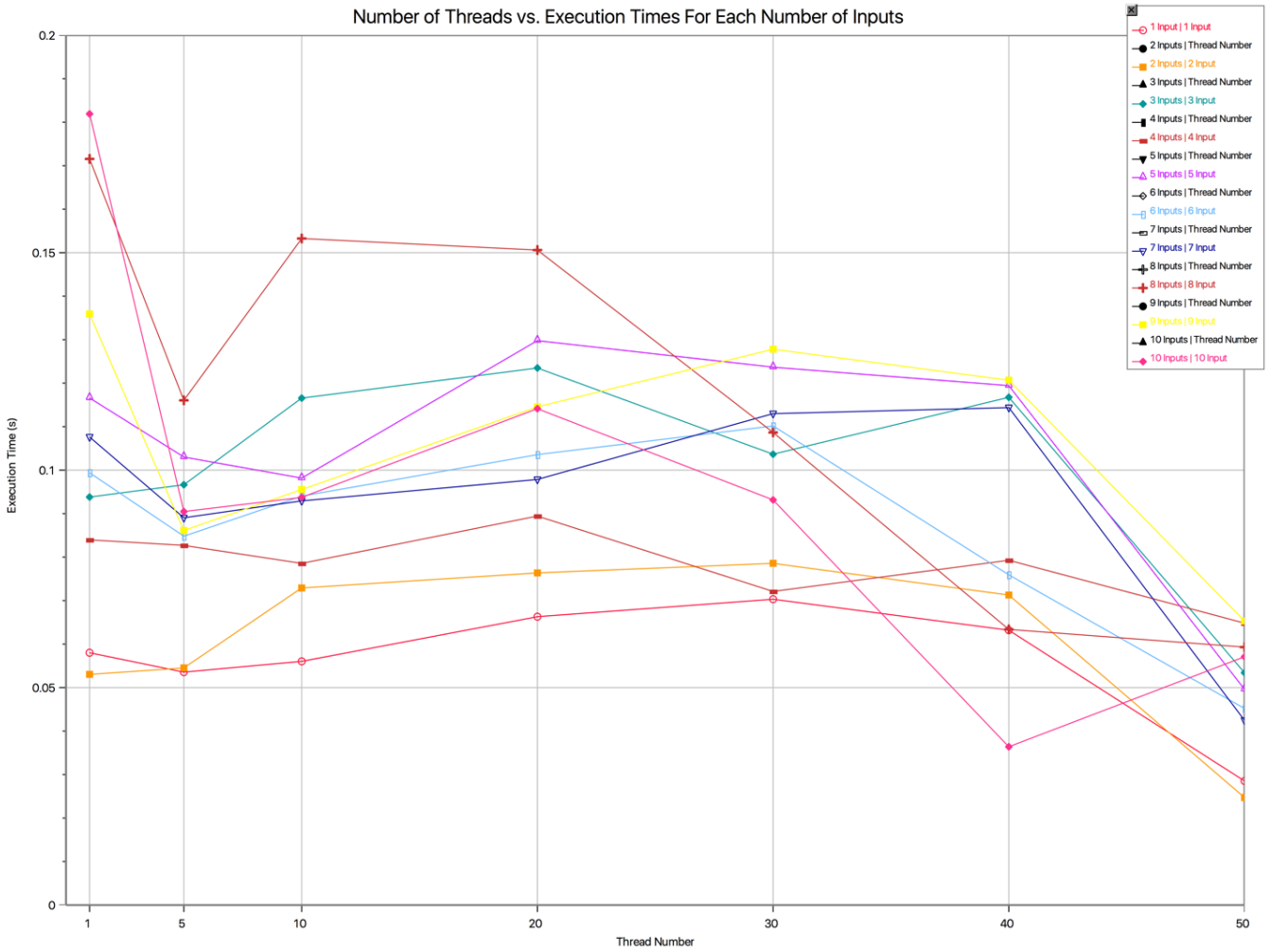

Figure 4 – Graph of Number of Threads vs. Execution Time for Each Number of Inputs
(The inputs are from sample inputs 7th file)

As seen on the graph, every number of inputs has an optimum execution time for a specific thread number. This is same for every input number with thread number 50, but excluding that, when we look at the intermediary thread numbers, 40 has a higher execution time rate than 30 and 20 thread numbers for 5, 6, 7 and etc. number of inputs. This shows that the hypothesis of increasing the number of threads decreases execution time is not always true in real life cases. The bottlenecks affect the linearity of the relation significantly.