Aldo Tali

Section 1

Umut Akös

Section 2

Operating Systems

CS342 Project 2 Report

Fall 2018-2019

Aldo Tali

21500097

&

Umut Akös

21202015

12.11.2018

Purpose of report:

The following is a report that tries to give insights and answers to the third part of the second project in Operating Systems course given in Bilkent University Fall 2018. The project itself is composed of two parts, one for testing the use of POSIX semaphores and the other the use of synchronization in threads. Below we show the results of some tests that are related to the second part of the project. Each test is run 6 times in order to take into consideration the standard deviation in runtime for the case being studied (example operating system related delays).

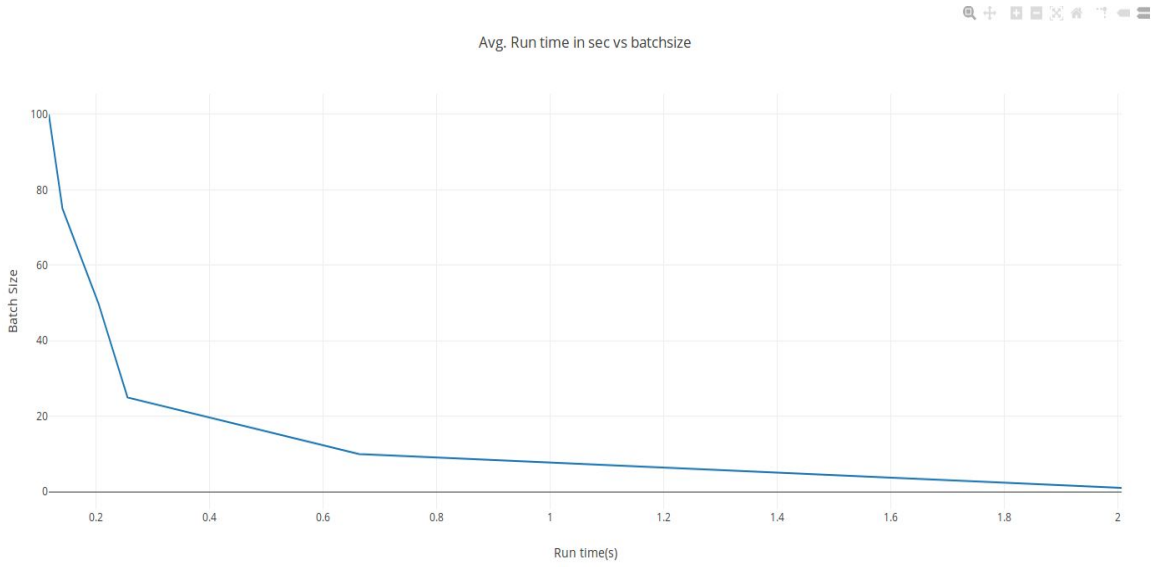
Testing batch sizes effect

Below we give the values recorded for several batch sizes in the multithreaded tests. The tests were run with an input of 3 files each containing 250000 numbers. These were kept constant to check purely the effect of the batchsize.

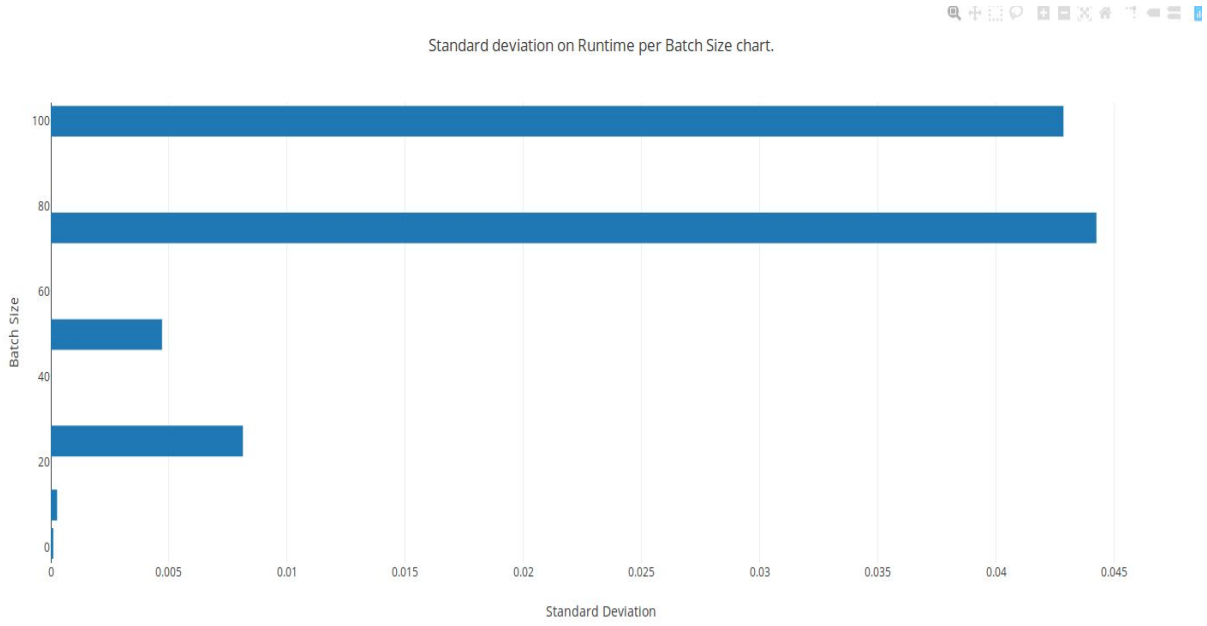
Table 1.0 Values for Batch Size and Run Time

Batch Size	Run1 (sec)	Run2 (sec)	Run3 (sec)	Run4 (sec)	Run5 (sec)	Run6 (sec)	Mean (sec)	S.Deviation
1	1.8296	1.7531	1.8808	1.8485	1.9797	2.0466	2.0063	0.106551234
10	0.4163	0.2239	0.3113	0.4128	1.3108	1.3038	0.6633	0.504030152
25	0.5336	0.1758	0.1533	0.1758	0.1537	0.3412	0.2555	0.153750663
50	0.1158	0.1424	0.2664	0.2505	0.2871	0.1624	0.2041	0.072478852
75	0.1335	0.1362	0.1317	0.1357	0.1663	0.1420	0.1409	0.012922074
100	0.1000	0.1294	0.1066	0.1564	0.0981	0.1107	0.1168	0.022361007

Avg. Run time in sec vs batchsize



Standard deviation on Runtime per Batch Size chart.



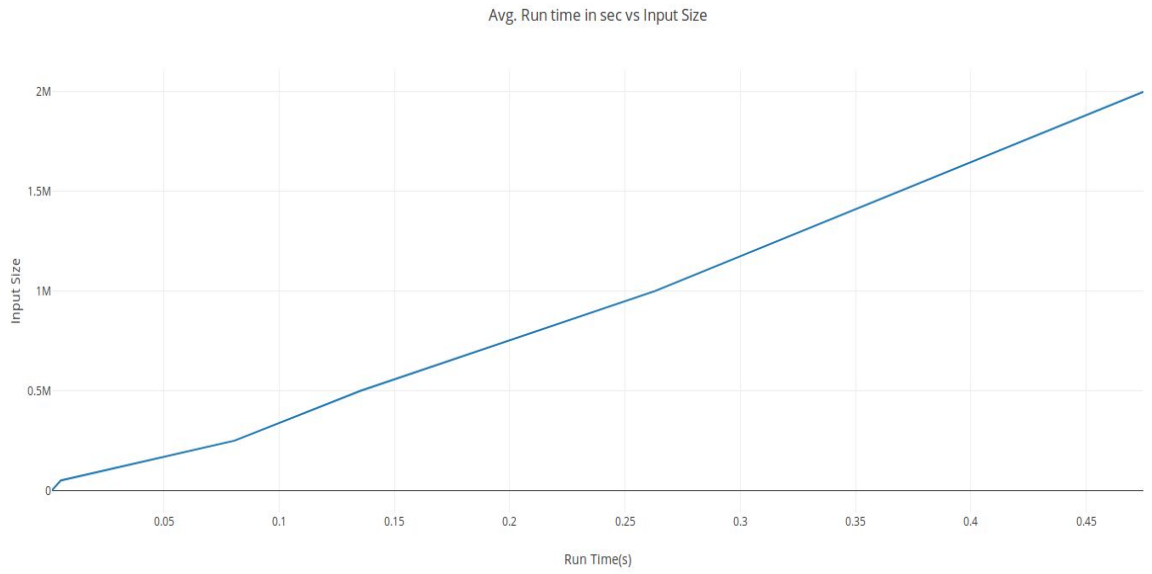
Testing input sizes effect

Below we give the values recorded for several input sizes in the multithreaded tests. The tests were run by keeping the batch size constant to B = 25. There is no particular assumption on the choice for this exact value on batch size, however we did see it fit to use it based on the results of the runs from Table 1.0 (example: It is the one that produced a somewhat an in between time for the runs).

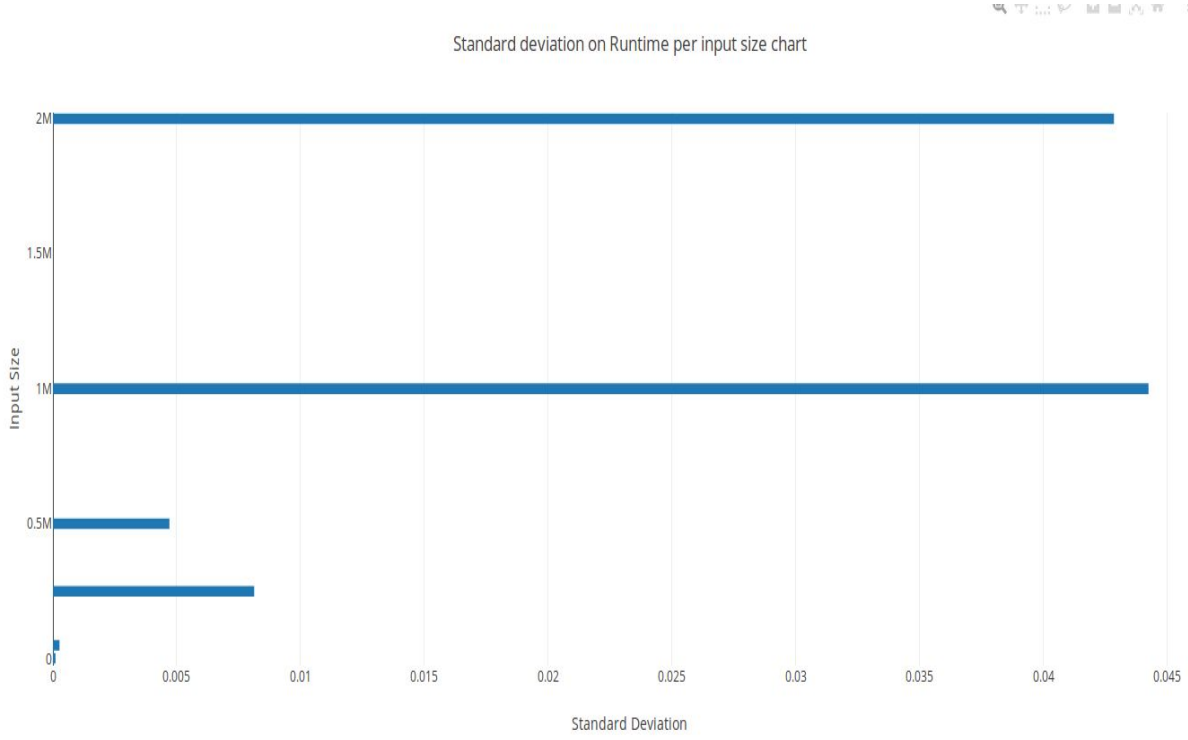
Table 2.0 Values for Input Sizes and Run Time

Input Size	Run1 (sec)	Run2 (sec)	Run3 (sec)	Run4 (sec)	Run5 (sec)	Run6 (sec)	Mean (sec)	S.Deviation
1000	0.00151	0.0015	0.0016	0.0014	0.0017	0.0016	0.0015	0.000104003
50000	0.0058	0.0051	0.0055	0.0054	0.0053	0.0051	0.0053	0.000265832
250000	0.0731	0.0883	0.0862	0.0893	0.0712	0.0762	0.0807	0.008126848
500000	0.1442	0.1344	0.1372	0.1342	0.1313	0.1321	0.1353	0.004704324
1000000	0.2577	0.2139	0.2570	0.2637	0.2399	0.3453	0.2629	0.044231772
2000000	0.4962	0.4514	0.4572	0.4117	0.5312	0.5012	0.4748	0.042836452

Avg. Run time in sec vs Input Size



Standard deviation on Runtime per input size chart



Insights:

As seen from the graph plotted by the values of Table 1.0, the average run time in seconds decreases as the batch size increases through the runs. It is important however to state that in this case we kept the input file size constant so that we would see this effect. One possible reason why batch size affects the run time of the application in the following way could be that in the case when the batch size is small then there is a larger possibility of the operating systems making more frequent context switches. So a lower batch size means that less number of files are passed on the globally stored linked list atomically so this translates into more code sections that are not blocked. The increased number of context switch adds more overheads for the operating system has well. So this means the faster execution time in higher batch size is something to be expected. This data is stable because as seen from the Standard deviation graph on different runs the going away from the average run time is only about 0.05 seconds and the relative change from one standard deviation to the other is even less. This reasoning supports the argument posed.

Regarding the second part where we kept the batch size fixed but gave different input sizes in the files, we can see that on the **Avg. Run time in sec vs Input Size** graph computed by the values of Table 2.0 we can see that the input size increase will increase the

run time in a linearly looking fashion. This result is also reasonable because the more numbers there are to read and process the more time the runtime should take. In all this we should mention that, one could tweek the values of the batchsize to try to get better execution times for larger input sizes. The standard deviation here follows a similar trend to the standard deviation of the values in Table 1.0 which accounts for extra small times that the operating system itself delays the run.