Assignment 2 Report

Weiqi Wang
Faculty of Applied Science
Simon Fraser University
Vancouver, Canada
wwa97@sfu.ca

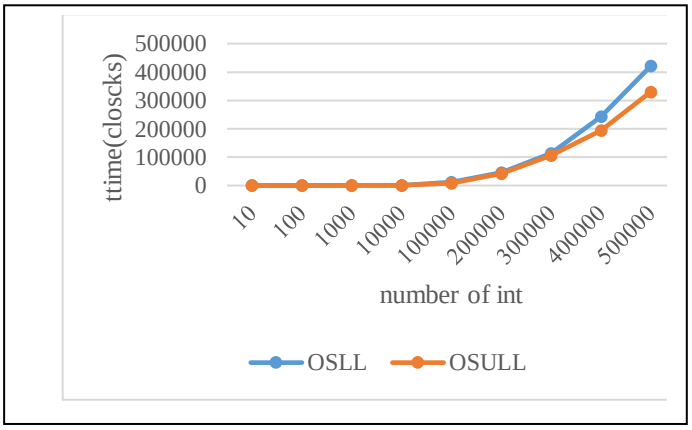
I. Data Inserting in Order

Data inserting in order has two tests, increasing order and decreasing order. These two tests aim to test the best and worst inserting performance for both OSLL and OSULL.

A. *Increasing order*

When inserting increasing order int numbers, the time for OSULL is slightly less than OSLL.

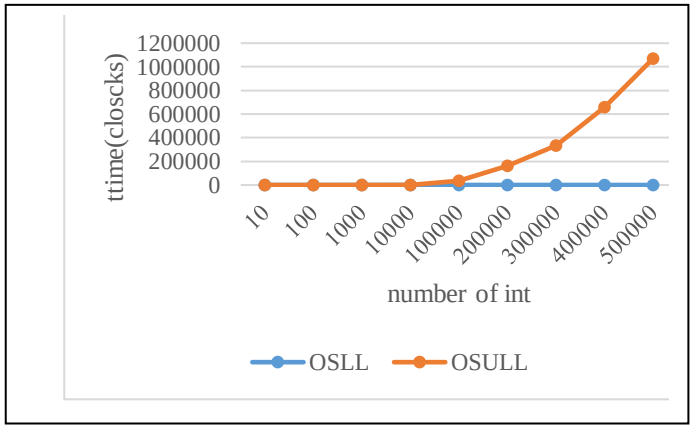
Number of Int	Time (clocks)	
	OSLL	***OSULL***
10	0	0
100	0	0
1000	1	1
10000	95	63
100000	11269	8256
200000	45034	42849
300000	113670	105817
400000	242334	194267
500000	421217	330387



B. *Decreasing order*

When Inserting decreasing order int numbers, the time for OSULL is much greater than OSLL.

Number of Int	Time (clocks)	
	OSLL	***OSULL***
10	0	0
100	0	0
1000	0	4
10000	1	340
100000	4	35612
200000	7	161709
300000	11	335878
400000	15	657999
500000	18	1069622



II. Data Removing in Order

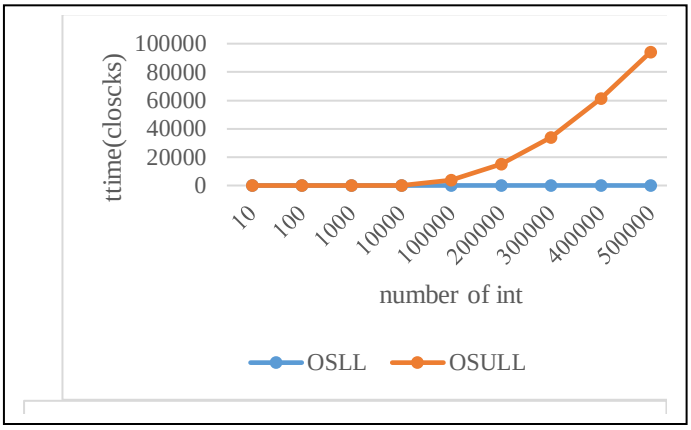
Data removing in order has two tests, increasing order and decreasing order. These two tests saim to test the best and worst removing performance for both OSLL and OSULL.

A. *Increasing order*

When removing decreasing in numbers, the time for OSULL is much greater than OSLL.

Number of Int	Time (clocks)	
	OSLL	***OSULL***
10	0	0

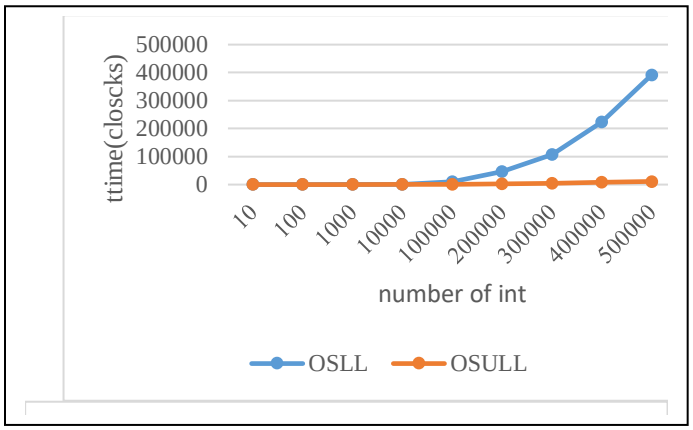
Number of Int	Time (clocks)	
	OSLL	***OSULL***
100	0	0
1000	0	0
10000	0	29
100000	3	3677
200000	4	14946
300000	6	33707
400000	9	61325
500000	11	94161



B. Decreasing order

When removing decreasing int numbers, the time for OSULL is much less than OSLL.

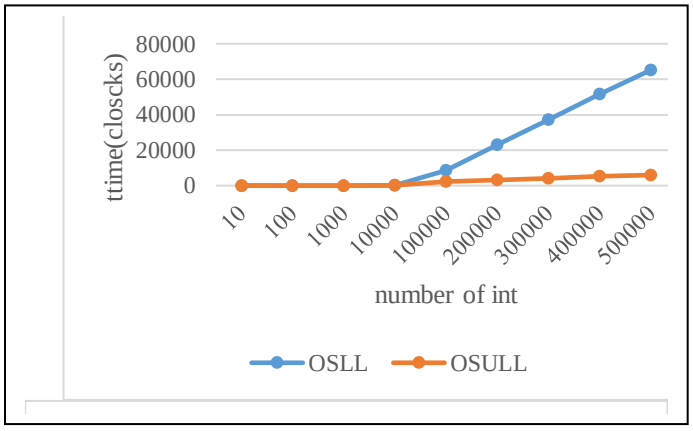
Number of Int	Time (clocks)	
	OSLL	***OSULL***
10	0	0
100	0	0
1000	1	0
10000	103	3
100000	11046	387
200000	46003	1795
300000	107681	4083
400000	224099	7354
500000	391787	11135



III. Data Inserting in Random Order

This test aims to test the inserting performance in real scenario for both OSLL and OSULL. When inserting random order int numbers, the time for OSULL is much less than OSLL.

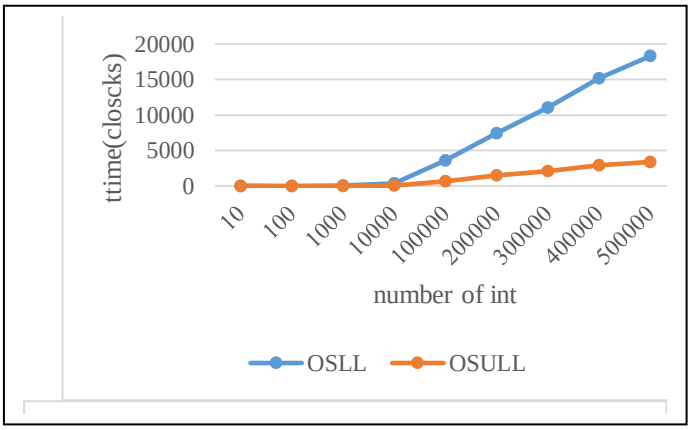
Number of Int	Time (clocks)	
	OSLL	***OSULL***
10	0	0
100	0	0
1000	0	2
10000	62	150
100000	8614	2180
200000	22932	3138
300000	37285	4199
400000	51651	5238
500000	65262	5981



IV. Data Searching in Random Order

This test aims to test the searching performance in real scenario for both OSLL and OSULL. When searching data in random order, the time for OSULL is much less than OSLL.

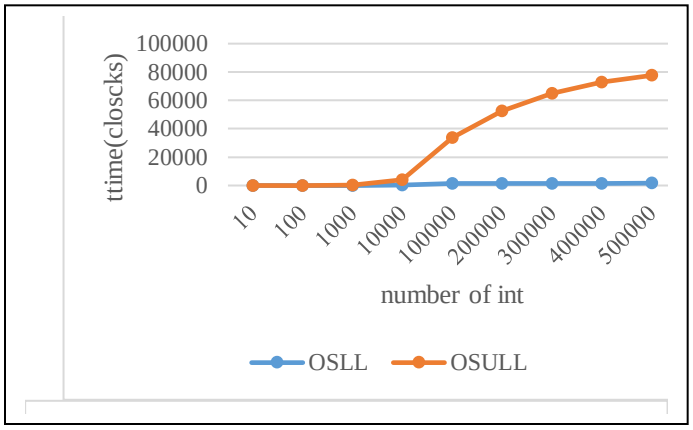
Number of Int	Time (clocks)	
	OSLL	***OSULL***
10	1	0
100	4	1
1000	35	7
10000	383	79
100000	3600	678
200000	7394	1475
300000	11030	2090
400000	15159	2930
500000	18283	3378



V. Data Removing in Random Order

This test aims to test the removing performance in real scenario for both OSLL and OSULL. When removing data in random order, the time for OSILL is much greater than OSLL.

Number of Int	Time (clocks)	
	OSLL	***OSULL***
10	0	0
100	4	61
1000	36	497
10000	322	4231
100000	1629	33793
200000	1630	52436
300000	1609	64886
400000	1641	73061
500000	1689	77625



VI. Problems in OSLL Class

OSLL class can't inserting number 0 to an empty OSLL object with int type. The cursor starts from the *cursor->next*, which is the *back* of the CSLL object. The *back* has initial value of 0, so the program won't jump in to the if statement *cursor -> data != item*, and the program won't insert 0 as the first item.

VII. Conclusion

OSULL has a better performance theoretically. However, in the seven tests I designed, OSULL do not have a better performance in overall. Algorithmic overhead for OSULL is a more significant factor that influence the performance of OSULL than the CPU cache module.