Harsimrat Singh
2015CS50284

COL380
Assignment 1

Problem 1

An argument vector, input, is the input vector is passed to function calcPrefixSum with number of threads. No new data structure is made. In this, I have implemented work-efficient parallel prefix sum algorithm. There are total of log2(n) steps where n is size of input vector.

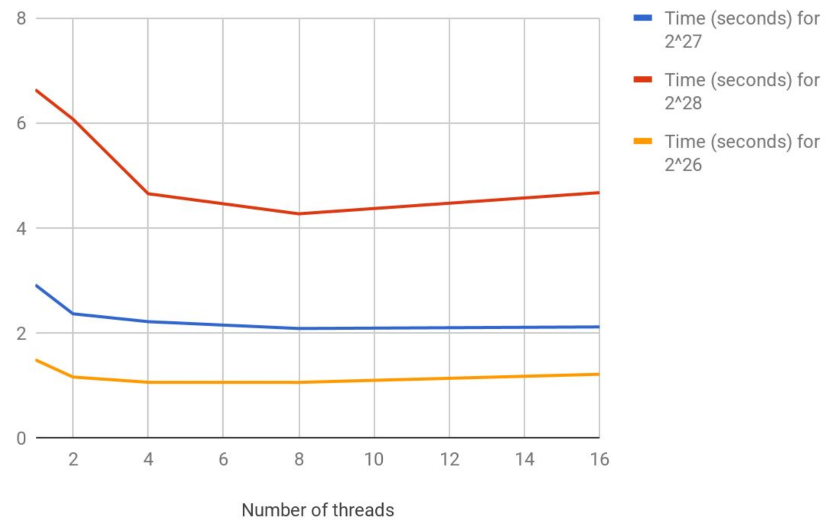
Now according to algorithm, input size should be power of 2. So I overcame padding by modifying my code. I have broken down algorithm into 2 parts - one for even indices and one for odd. Both have 2 nested for loops. First loop for both steps is of O(log n) and second loop iterates over vector element wise. Since we have to parallelize this algorithm, the only way to do this is divide inner for loops of both parts to threads.

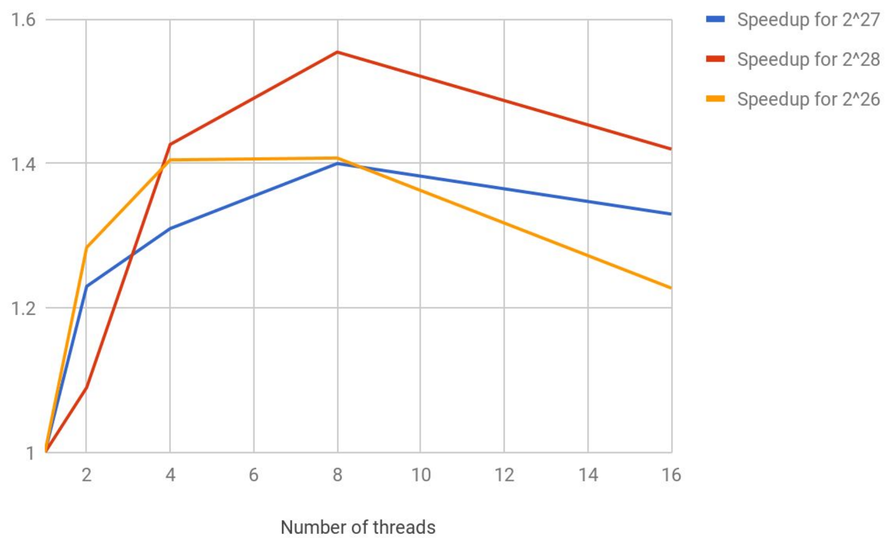
After completing the algorithm, I removed extra zeros that I had added. Since I have not created any extra vector as I am modifying the given input vector, so I return it in the end of function.

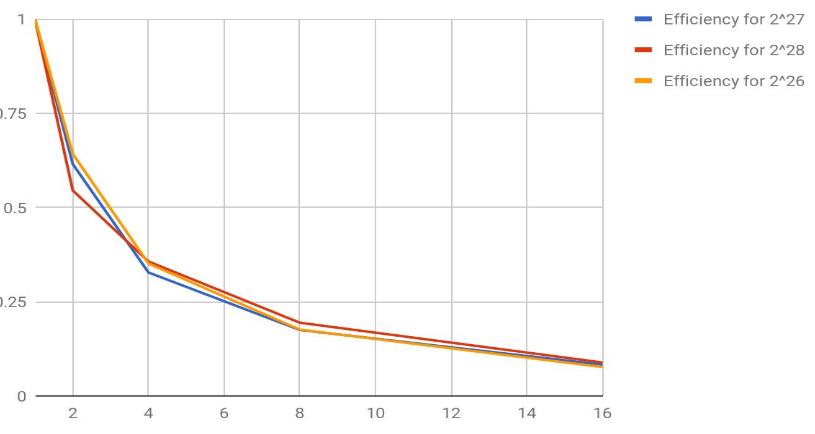
These are observation I get after running my code on input size of 134217728 i.e. 2^{27}

Number of threads	Time (seconds)
1	2.91
4	2.3
8	2.08
16	2.11

We can see that as we increase number of threads time taken by code decreases. Speedup increases as we increase threads as workload keeps on dividing and time taken is less. But as we go beyond certain point and keeps on increasing number of threads then time taken for creating and destroying threads takes more time than the whole process itself. So T_p start to increase and speedup starts decreasing.


8
6
4
2
0
2
4
6
8
10
12
14
16
Number of threads
Time (seconds) for 2^27
Time (seconds) for 2^28
Time (seconds) for 2^26


1.6
1.4
1.2
1
2
4
6
8
10
12
14
16
Number of threads
Speedup for 2^27
Speedup for 2^28
Speedup for 2^26


1
0.75
0.5
0.25
0
2
4
6
8
10
12
14
16
Efficiency for 2^27
Efficiency for 2^28
Efficiency for 2^26