High Performance Computing
Assignment 3

Jahnavi Suthar(201301414)   Deeksha Koul(201301435)

Prefix Sum(Exclusive Scan)

We have to calculate the prefix sum of the given array.

$$\text{prefix sum}[i] = \begin{cases} 0 & , i = 0 \\ \sum_{k=0}^{i-1} input[k] & , i > 0 \end{cases}$$

Complexity of the Problem(Serial)

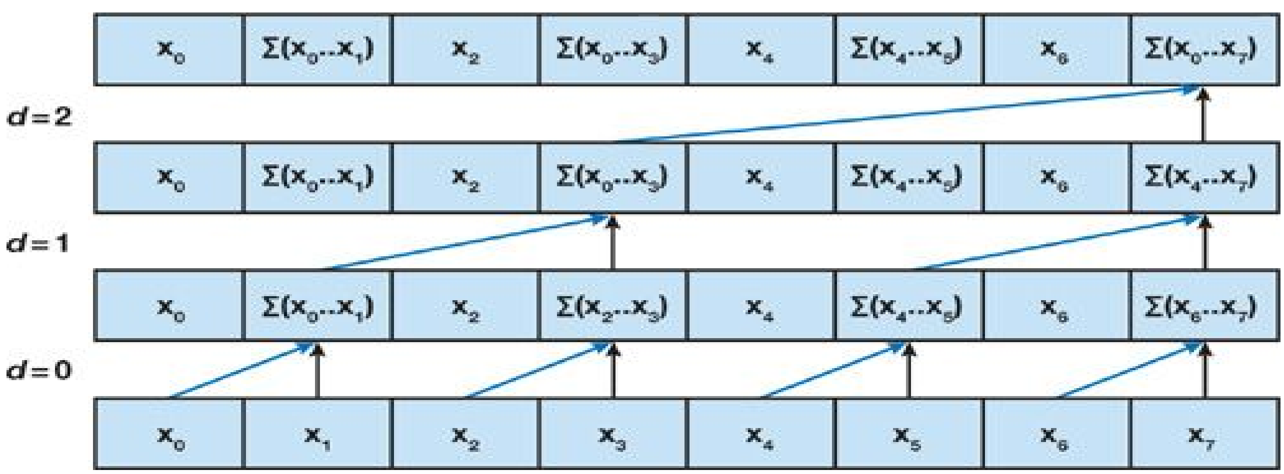
Serial complexity of the problem is O(n), where n is the input size.

Span of Problem:

Parallel Span is O(logn), which is the height of the binary tree that we traverse twice.

Optimization Strategy:

Blelloch Algorithm is used for the parallel implementation of the problem. The algorithm is divided into two phases: Up sweep or Reduce Phase and Down sweep.



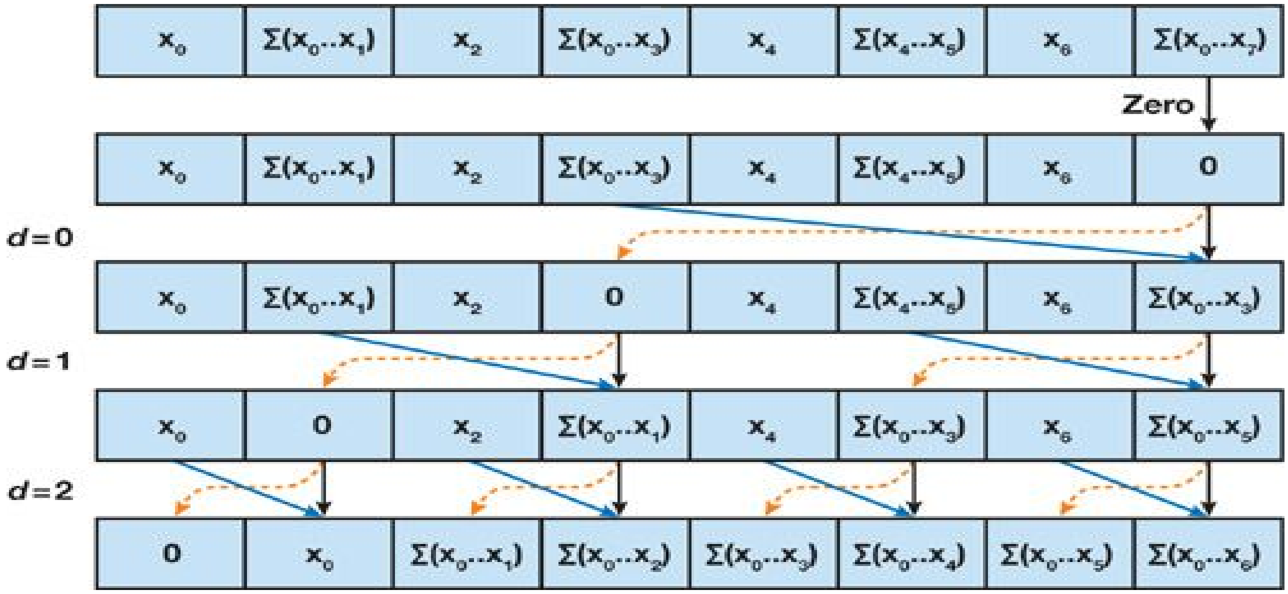
Up Sweep

Up sweep :

In the reduce phase, we traverse the tree from leaves to root computing partial sums at internal nodes of the tree. At the end of this phase, the last element contains sum of all the elements of the array.

Down sweep:

We start by inserting zero at the root of the tree(last element of the array), and on each step, each node at the current level passes its own value to its left child, and the sum of its value and the former value of its left child to its right child.



Down sweep

Hardware Details

CPU Model:INTEL(R) Core i5-4590

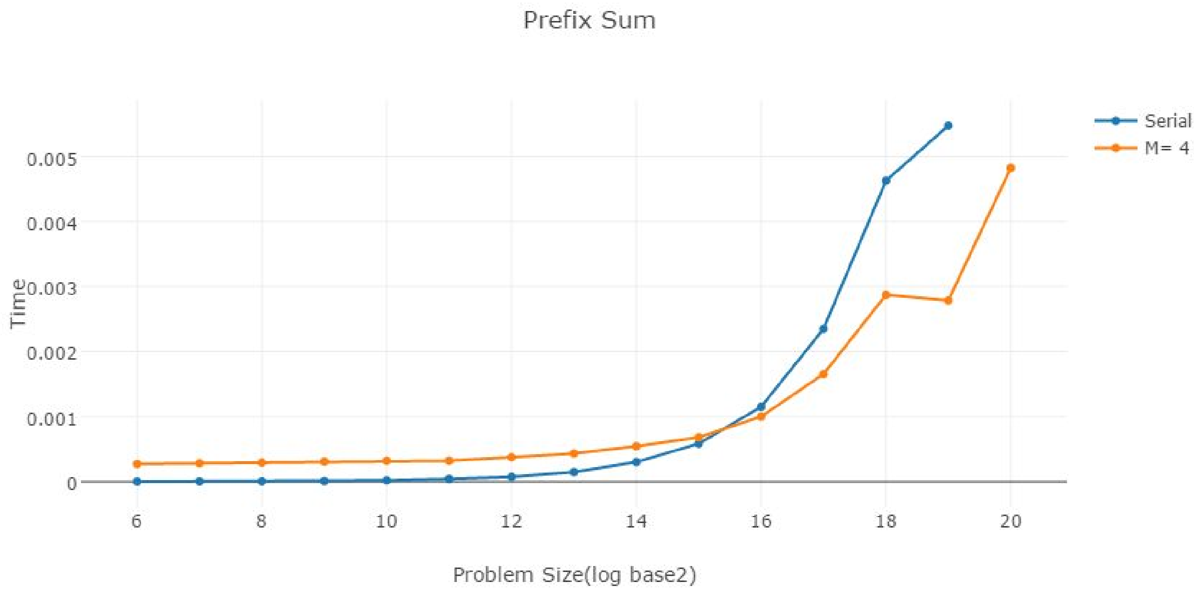
No. of cores: 16

Memory:7.6 GiB

Compiler:gcc

Optimization flags if used: None

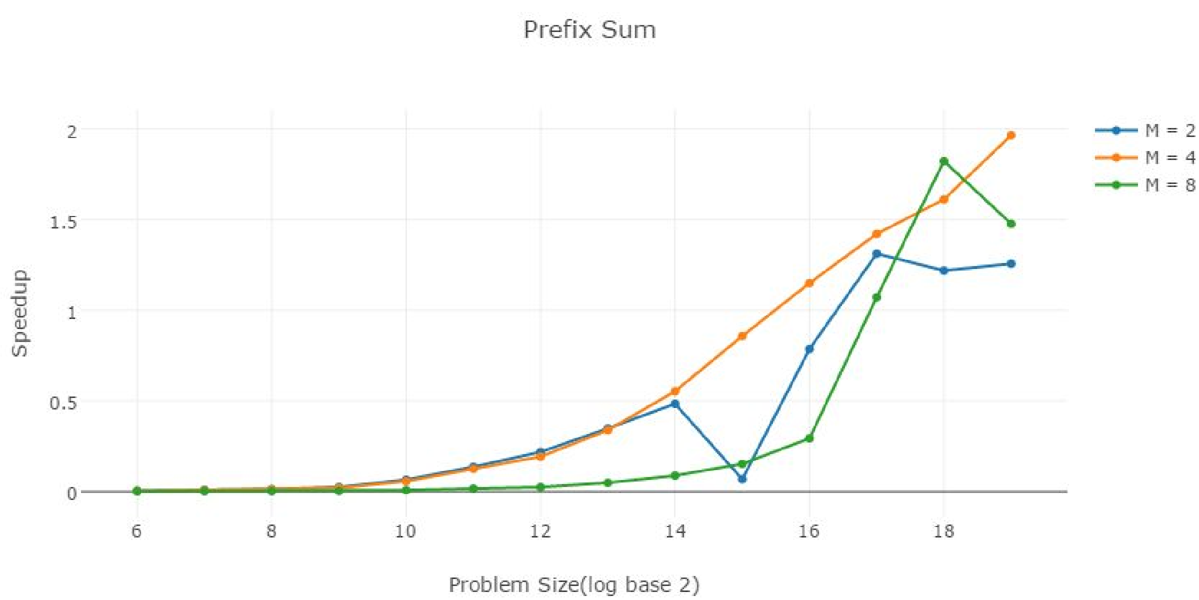
Precision: double



Problem Size vs Time(Serial and Parallel)

Observation:

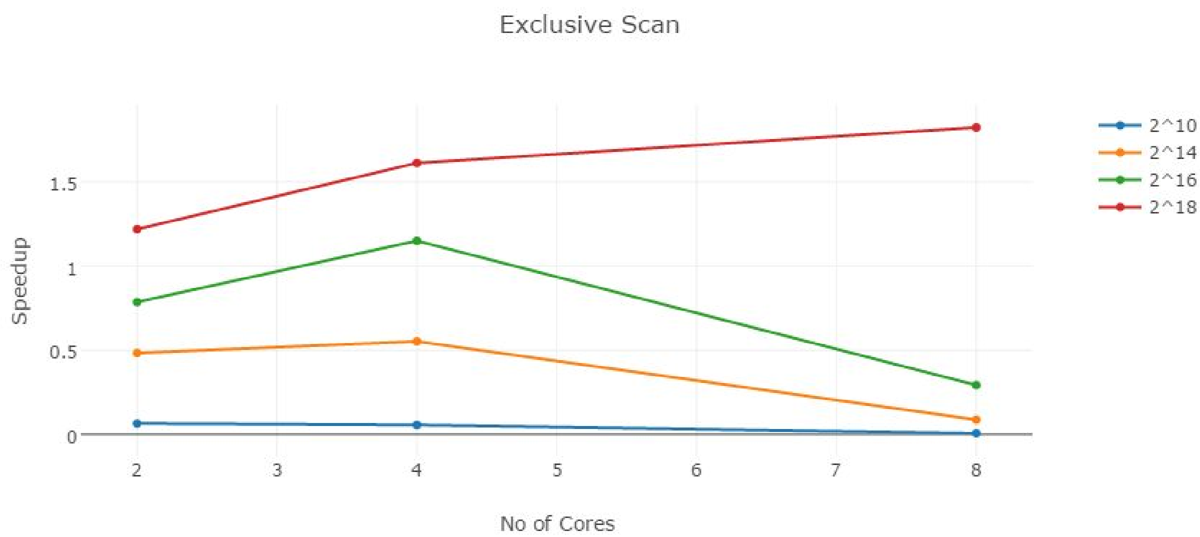
For small input sizes(upto 2^{15}), the parallel code takes more time than the serial code because of the overhead of dividing work among the threads surpasses the reduced parallelized time. For the larger input sizes parallel code performs better.



Problem Size vs Speedup for 2, 4 and 8 cores

Observation:

With the increase in the problem size, speedup increases. For larger input sizes, the reduced parallelized time compensates for the overhead.For the small input sizes, 2 cores perform better than 4 cores, but with the increase in problem size, 4 cores perform much better than 2 cores.



No of cores vs Speedup

Observation:

For a particular number of cores as the input size increases (becomes greater than 2^{16})then the speedup tends to increase that is parallel code becomes more efficient as compared to serial code .As number of cores increases from M=2 to M = 4 the speedup tends to increase for larger value of the input parameter but its more efficient for M=4 as compared to M=8 . (possibly due to overheads)

Problems faced due to parallelization:

1. For input ($N>2^{20}$) the code output shows Segmentation Fault,primarily because of the memory bound constraints.
2. For some larger values of N (2^{17}) the code performs better for M=4 than for M=8 , the possible reasons might be that the number of overheads overpower the amount of time saved due to parallelization.
3. The code shows correct value for only those N that can be represented in power of 2 as the up sweep and down sweep treats array as a binary tree and sum the numbers in pair , it is better to give input in form of 2^x.

Inclusive Scan

We can convert the exclusive scan to inclusive scan using array of size 1 more than the input size. At the end of the up sweep phase the last element would be containing the sum of all the elements of the array, we copy that value to another location and perform the down sweep.