PREFIX SUM REPORT

Name: Udayin Biswas
Entry Number: 2014CS50296
Machine Specifications:2.5 GHz I5 core(quad core), 4 gb ram

For calculating the Prefix Sum, it had to be done in O(log n) phases at O(n) cost.
This calculation was done in a specific manner as follows:
We are provided with an input vector of the numbers and suppose n threads.
Assumption: n is 2 to the power of k, k>0.
Data structure used was a vector(/array) which we can assume to take the shape of a tree when computation is performed in log n phases.
First, we have to sum up certain elements at each stage, with log(n) (to the base 2) stages.
At stage 0, every 2 neighbour elements at a gap of 1 get summed up together .
a[1]=a[0]+a[1]
a[3]=a[2]+a[3]
a[5]=a[4]+a[5]
a[7]=a[7]+a[6]

At stage 1, every 2 elements at a gap of 2 get summed up.
a[3]=a[3]+a[1]
a[7]=a[7]+a[5]

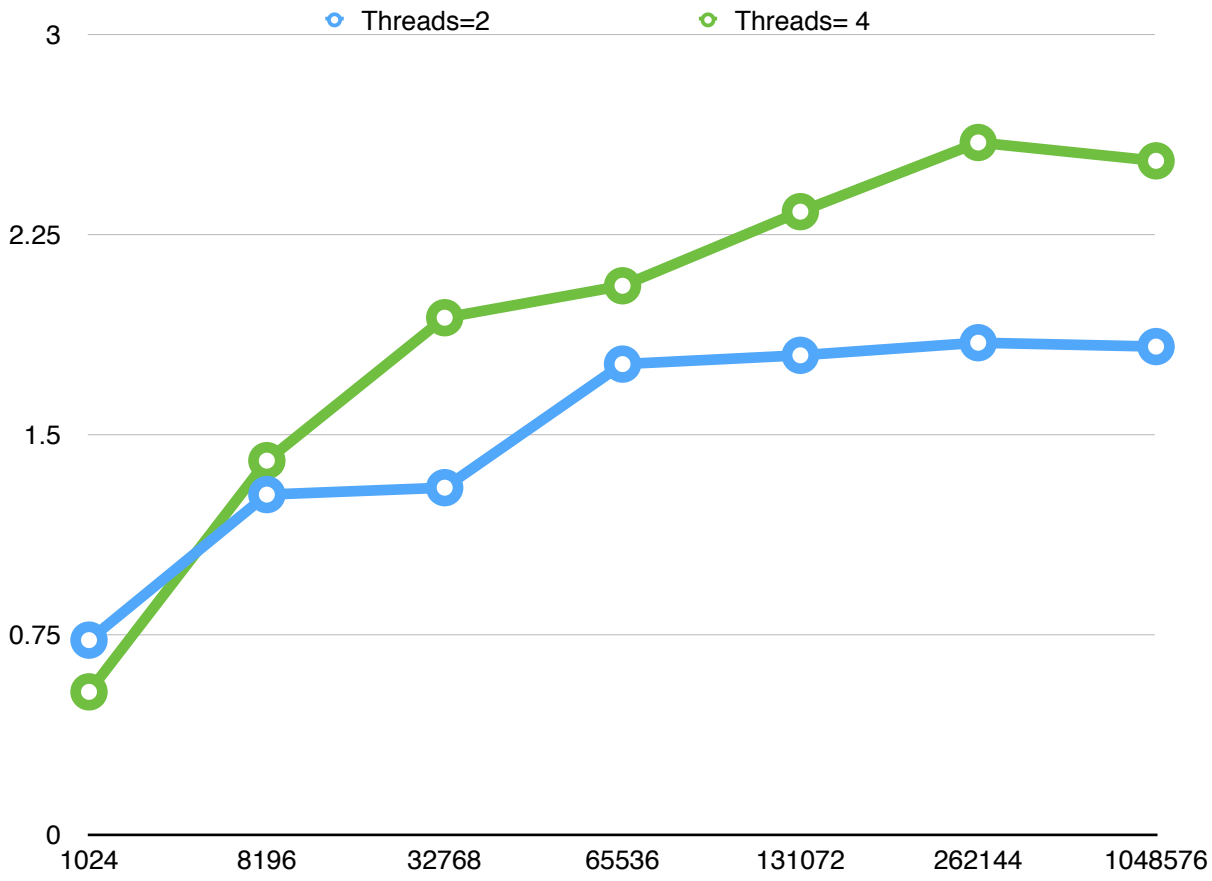
At stage 2, every 4 elements at a gap of 4 get summed up.
a[7]=a[7]+a[3]

….
and so on.
So total we can see we have log(n) phases, and the computation cost halves at each increasing stage.

Eg: 1,2,3,4,5,6,7,8
->Stage 0: 1,3,3,7,5,11,7,15
->Stage 1: 1,3,3,10,5,11,7,26
->Stage 2: 1,3,3,10,5,11,7,36
Now next stage,
we have to traverse back from a[7] to rest of the elements in a manner.
first: the last element is set to 0.
-> 1,3,3,10,5,11,7,36
At each stage now,left child value = temp, left child value=right child value, right child value=temp +left child value. log n such phases again.
Eg above- 1,3,3,10,5,11,7,0
Stage 0: a[3]=0,a[7]=9+0=9 -> 1,3,3,0,5,11,7,9
Stage 1: a[1]=0,a[3]=3,a[5]=9,a[7]=14 -> 1,0,3,6,5,9,7,14
and so on,
we finally get 0,1,3,6,10,15,21,28
Shift by 1 to the left and calculate the final value by adding 8 to 28.
=>1,3,6,10,15,21,28,36. So the basic strategy was to first sum up elements region wise so that we can get the total at the end. then we set it 0 so that 0 can travel to the left. Meanwhile the right child is the sum of itself and the left child value at each stage. The left child carries the 0 to the left most of the vector. This strategy will help us get prefix sum.

Parallelisation was done on the 2 for loops to divide elements of array among the threads, at each stage. Each thread takes the next consecutive element at a gap of n threads for computation so that the data is divided equally among the threads.

Speedup vs problem size



Efficiency vs problem size