Parallelizing techniques - 2

Requirement 1

Given a sequence of *n* numbers, compute the sums of the first *k* numbers, for each *k* between 1 and *n*. Parallelize the computations, to optimize for low latency on a large number of processors. Use at most 2**n* additions, but no more than 2*log(*n*) additions on each computation path from inputs to an output. Example: if the input sequence is 1 5 2 4, then the output should be 1 6 8 12.

Solution

Split the work to be done in buckets of size 2 or LENGTH/NUMBER OF THREADS.
For each bucket, compute the linear sum.
After doing this in parallel, start the same number of threads with the same buckets, but each one waiting for the last value of the previous bucket to be computed.
After receiving it, it gives the opportunity of the next bucket to be computed. This is done using a class OneShotEvent, with a condition variable and a mutex and 2 functions, wait() and signal().

Hardware

Processor: Intel(R) Core(™) i7-8750H CPU @ 2.20GHz 2.21 GHz
RAM: 16 GB
System type: 64-bit OS
Platform: Windows 10

Tests

Sequence length	Number of threads	Time taken
100	2	0.0033644 s
100	50	0.0158159 s
80000000	1	3.51297 s
80000000	5	1.92736 s
80000000	300	1.68277 s

Requirement 2

Add *n* big numbers. We want the result to be obtained digit by digit, starting with the least significant one, and as soon as possible. For this reason, you should use *n-1* threads, each adding two numbers. Each thread should pass the result to the next thread. Arrange the threads in a binary tree. Each thread should pass the sum to the next thread through a queue, digit by digit.

Solution

Create n-1 threads and use n-1 producer-consumer queues to send the results from the additions one by one, at each step considering the carry so at the end we can simply add the element from the last queue to a stack and then print them in that order, that being the result of the addition of the n big numbers.

Hardware

Processor: Intel(R) Core(™) i7-8750H CPU @ 2.20GHz 2.21 GHz
RAM: 16 GB
System type: 64-bit OS
Platform: Windows 10

Tests

Number of big numbers	Number of digits (between …)	Number of threads	Time taken
20	300 and 1300	19	0.0056251 s
50	300 and 1300	49	0.0140456 s
200	300 and 1300	199	0.043062 s
3000	300 and 1300	2999	0.667698 s