Parallel array sum speedup report

Author: Fahad Nayyar, IIIT-Delhi

Introduction

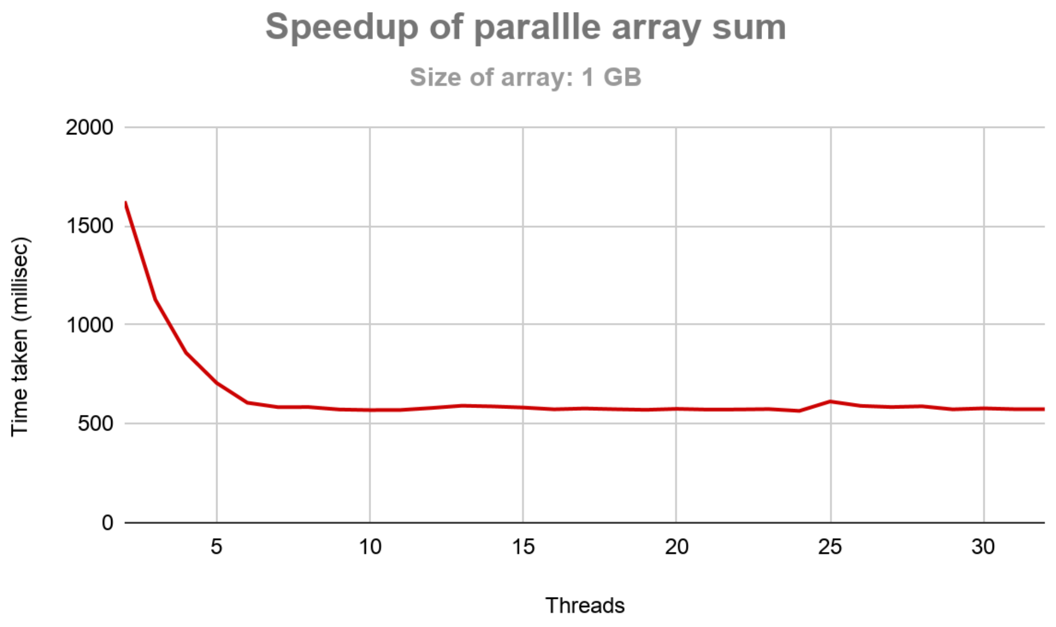
This report contains the results and discussions about the evaluation of the parallel array sum using the APIs of ParallelSum class. The experiment was conducted by addition of a 1 GB array using threads varying from 1 to 32. The aim is to verify the runtime complexity of **O(M/N+Log(N))**, where **M** is the size of input_array and **N** is the number of threads.

Machine used for experiment

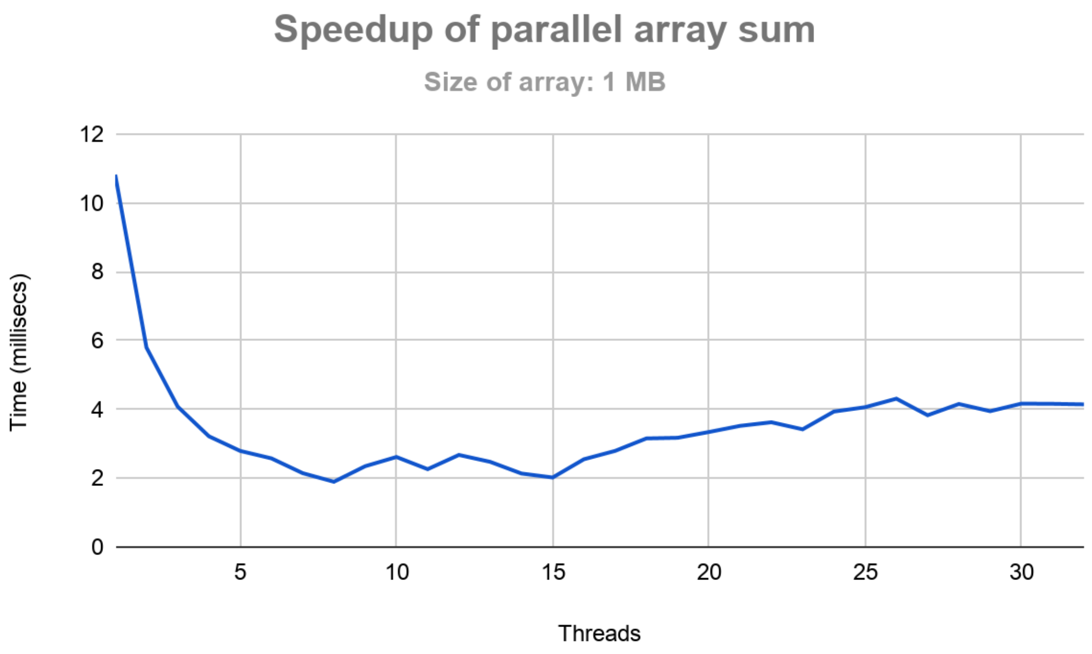
- Processor: **Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz**
- CPU cores used: **16**
- OS: Ubuntu **19.04**
- Linux Kernel version: **5.0.0-38-generic**

Speedup graphs:

Speedup for 1 GB array sum



Speedup for 1 MB array sum



Conclusion

The **O(M/N)** factor is observable from the speedup graphs by the hyperbolic function plots. The speedup flattens for 1 GB array and slightly decreases for 1MB array. This might be due to hyperthreading (using more number of threads than the actual number of cores present in the machine) and thread launch overheads.
The **O(log N**) factor is not observed in the graphs because the number of threads is significantly less than the size of the array (N << M). For the O(Log N) factor to dominate it must be that M/N < Log N => **M < N Log N.** This factor can be observed if the number of threads used is almost similar to the size of the array (and very high number of CPU cores in the machine).
To summarize I was able to observe the O(M/N) speedup.

<u>Possible Improvement:</u> Parallel array sum can be a very good case for GPUs (massively parallel threading programming models). In that case we can just do the parallel reduction (bottom up addition of tree) to get **O(log M)** time by using M/2 number of threads (half of the size of the array). But this would change the input_array (in place). But the runtime O(Log M) would be significantly better than O(M/N) observed using pthreads on multi-code CPUs.