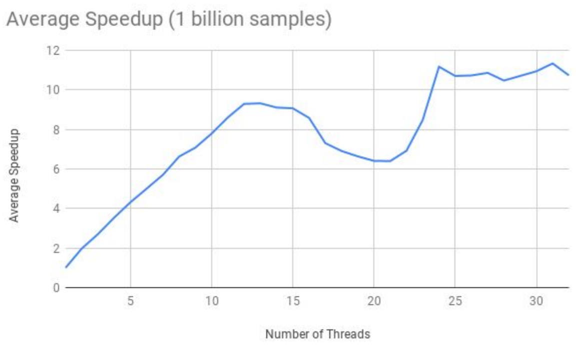
CS547 Project 1 Report

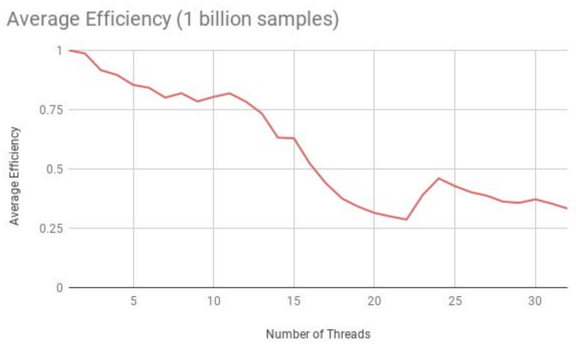
All experiments were run with 1 billion samples and the following `sbatch` configuration:

```
#SBATCH --cpus-per-task=24
#SBATCH -N 1
#SBATCH --mem=8G
```

I first implemented two integration techniques: Trapezoid and Monte-Carlo techniques. When I compared their runtimes, they were very similar (see `graphs/tz-mc.png`), so for the rest of this report, I will focus on my results for the Trapezoid method

The following graphs show the average speedup/efficiency across 3 runs (see `graph/` for larger images).





The experiments were run on a single node in the `openhpc` cluster. Each node has 12 physical cores with 2-way hyperthreading on each core, with a total of 24 logical processors. If we look at the speedup graph, the speedup increases up to 12 threads. This is because when we run 1 to 12 threads, each thread has its own physical core while decreasing the number of samples that each thread computes. Between 13 and 21 threads, the speedup decreases because there are cores with 2 hyperthreads executing per core.

There is a strange increase in speedup between 22 and 24 threads. As discussed with the professor, this may be due to uneven load balancing, where, due to the uneven distribution of threads across physical cores, some threads may complete earlier than the threads on other cores, which leaves some cores idle.

From 24 threads and on, there is not much increase in speedup. This is because there are more threads than processors available in the node, and the threads must now context-switch between each other to complete. Any minor increase in speedup after 24 threads may be due to uneven load balancing.

Prior to these results, I had issues with false sharing with passing a shared `sums*` array to each thread. Even though each thread only modifies their own array value, each write causes other threads to reload the `sums` array from memory. Other minor changes were to set variables to `const` whenever possible, to allow compiler optimization (in particular, I set the integration parameters to `const` when passed into the thread functions).