3. ARRAY SUM

Joey Welvaadt

HOGESCHOOL UTRECHT 1734098

https://github.com/AI-S4-2023/v2hpp-herkansingsopdrachten-JoeyWelvaadt1999

1. Code

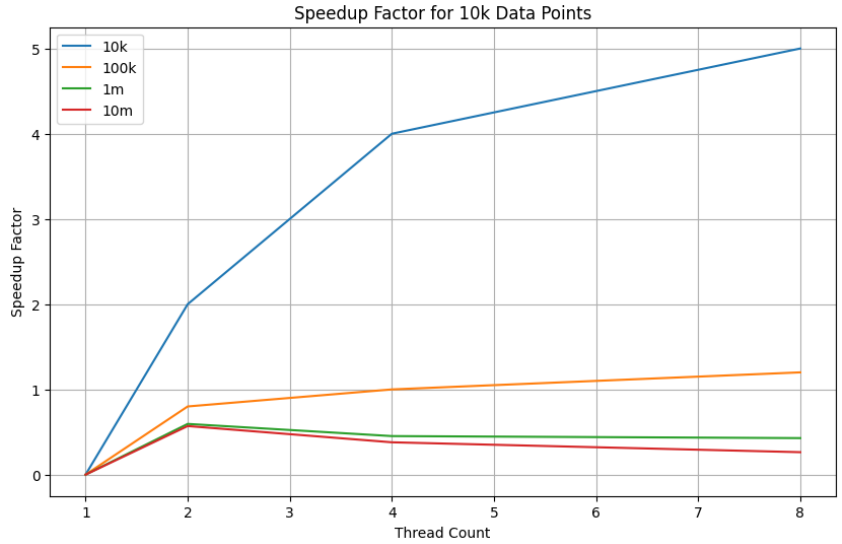
Initieel kreeg ik als resultaat van de omp time function steeds 0 seconde te zien. Met deze reden heb ik ervoor gekozen om de arraySum functie 100 keer te draaien en het gemiddelde van de tijd te nemen als waarde voor mijn tabel.

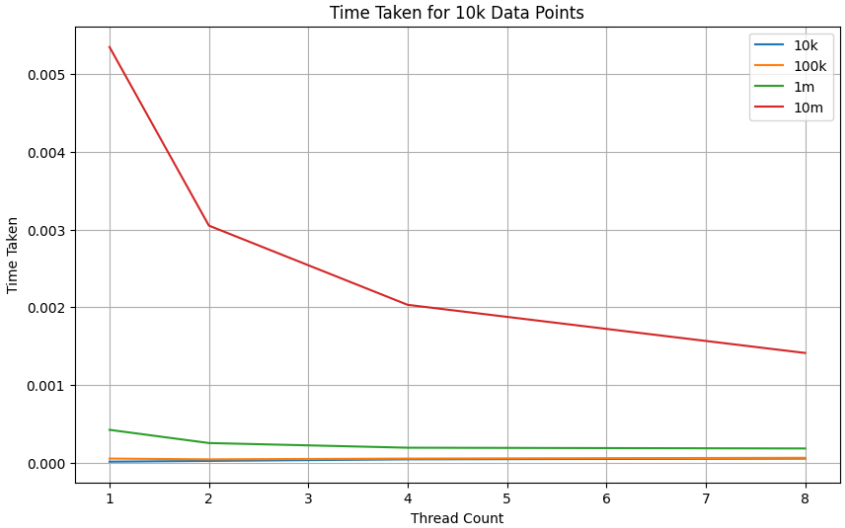
```cpp
int main(int argc, char *argv[]) {
    std::vector<int> thread_counts = {1, 2, 4, 8};
    for (int threads : thread_counts) {
        omp_set_num_threads(threads);
        double sum;

        if (argc != 2) {
            cout << endl << "*** Usage: arraySum <inputFile>" << endl << endl;
            exit(1);
        }

        input_array a = readArray(std::string(argv[1]));

        double startOpenMP = omp_get_wtime(); // start OpenMP timer

        int runs = 100;
        // Run 10 times to get a more accurate average time
        for (int i = 0; i < runs; i++) {
            sum = sumArray(a);
        }

        double endOpenMP = omp_get_wtime(); // end OpenMP timer
        cout << std::fixed << std::setprecision(10);
        cout << "Number of threads: " << threads << endl;
        cout << "Elapsed time (OpenMP): " << (endOpenMP - startOpenMP) / runs << " seconds." << endl;
        cout << "The sum of the values in the input file '" << argv[1]
        << "' is " << sum << endl;
    }

    return 0;
}
```

```cpp
double sumArray(input_array a) {
    unsigned int i;
    unsigned int numValues = a->size();
    double result = 0.0;

    #pragma omp parallel for reduction(+:result)
    for (i = 0; i < numValues; i++) {
        result += a->at(i);
    }

    return result;
}
```

2. Results

Ik heb ervoor gekozen om hieronder de resultaten weer te geven in beide een tabel en een plot. In de plot valt mij op dat hoe groter de bestanden zijn hoe minder de uiteindelijke speed up factor is tussen verschillende aantallen threads.





File Size	Data Points	Thread Count	Execution Time (seconds)	Speedup Factor
10k	10,000	1	0.0000099993	0.0
10k	10,000	2	0.0000199986	2.0
10k	10,000	4	0.0000399995	4.000230016
10k	10,000	8	0.0000499988	5.000230016

100k	100,000	1	0.0000499988	0.0
100k	100,000	2	0.0000399995	0.8000092002
100k	100,000	4	0.0000500011	1.000046001
100k	100,000	8	0.0000600004	1.200036801
1m	1,000,000	1	0.0004200006	0.0
1m	1,000,000	2	0.0002500010	0.5952404762
1m	1,000,000	4	0.0001900005	0.4523820476
1m	1,000,000	8	0.0001799989	0.4285714286
10m	10,000,000	1	0.0053500009	0.0
10m	10,000,000	2	0.0030499983	0.5700930443
10m	10,000,000	4	0.0020300007	0.3794393193
10m	10,000,000	8	0.0014100003	0.2635514136