**Homework 1 Report**

# 1 Test Environment

I tested my code on department's ix server. It has two sockets with AMD Opteron 6376 on each socket. Each CPU has 8 cores, 16 threads. So, there are 32 hardware threads in total.

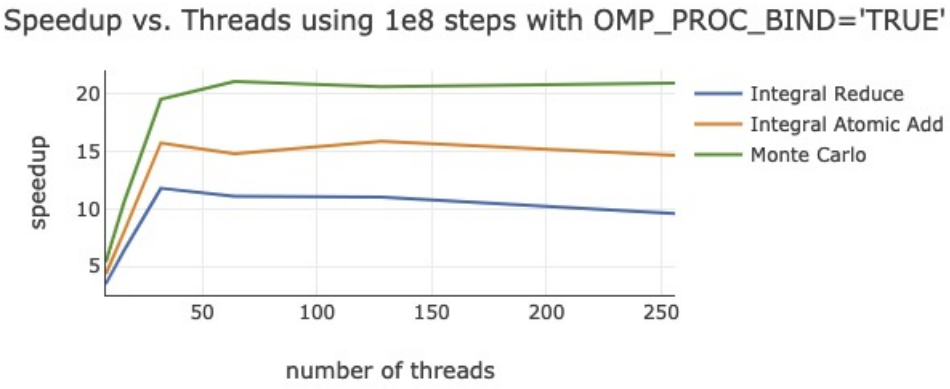
# 2 Test Results

| Steps | Accuracy (Pi is 3.1415926535) | Speedup (reduce) | Speedup (atomic add) |
|---|---|---|---|
| 1e6 | **3.1415**46524 | 1.30x | 1.95x |
| 1e7 | **3.141592**8536 | 5.79x | 8.93x |
| 1e8 | **3.1415926**736 | 11.17x | 15.09x |
| 1e9 | **3.14159265**56 | 11.99x | 16.07x |
| 1e10 | **3.14159265**50 | 13.00x | 16.09x |

Tabelle 1: Integral Method using 32 threads

| Steps | Accuracy (Pi is 3.1415926535) | Speedup |
|---|---|---|
| 1e7 | **3.14**11308000 | 16.00x |
| 1e8 | **3.1415**529600 | 18.45x |
| 1e9 | **3.14159**97440 | 23.29x |

Tabelle 2: Monte Carlo Method using 32 threads




# 3 Findings

- For both the integral method and Monte Carlo method, we can get better accuracy by increasing the number of steps, but there is a limit on how much accuracy we can get.
- Integral method has a better accuracy than Monte Carlo method in general.
- We can speedup the computation by increasing number of threads. However, once it passes the number of processors(hardware thread), it may hurt the performance. The figure above shows that we can achieve peak speedup at 32 threads for integral method, and 64 for Monte Carlo method.