Runtime Analysis

Nicholas Gammel, Olivia Brague, Timmy McKirgan, Dustin O'Brien, Kyle Tranfaglia

COSC 420 – 001

Professor Yaping Jing

Analysis of OMP

a)

Number of Threads (N) = **2**

Serial Runtime (Ts) = **0.201965s**

Parallel Runtime (Tp) = **0.156684s**

Speed up (S) = Ts / Tp = 0.201965 / 0.156684 = **1.28899568558**

Efficiency (E) = S / N = 1.28899568558 / 2 = **0.64449784279**

Number of Threads (N) = **4**

Serial Runtime (Ts) = **0.201965s**

Parallel Runtime (Tp) = **0.157535s**

Speed up (S) = Ts / Tp = 0.201965 / 0.157535 = **1.28203256419**

Efficiency (E) = S / N = 1.28203256419 / 4 = **0.32050814104**

Number of Threads (N) = **8**

Serial Runtime (Ts) = **0.201965s**

Parallel Runtime (Tp) = **0.166883s**

Speed up (S) = Ts / Tp = 0.201965 / 0.166883 = **1.21021913556**

Efficiency (E) = S / N = 1.21021913556 / 8 = **0.15127739194**

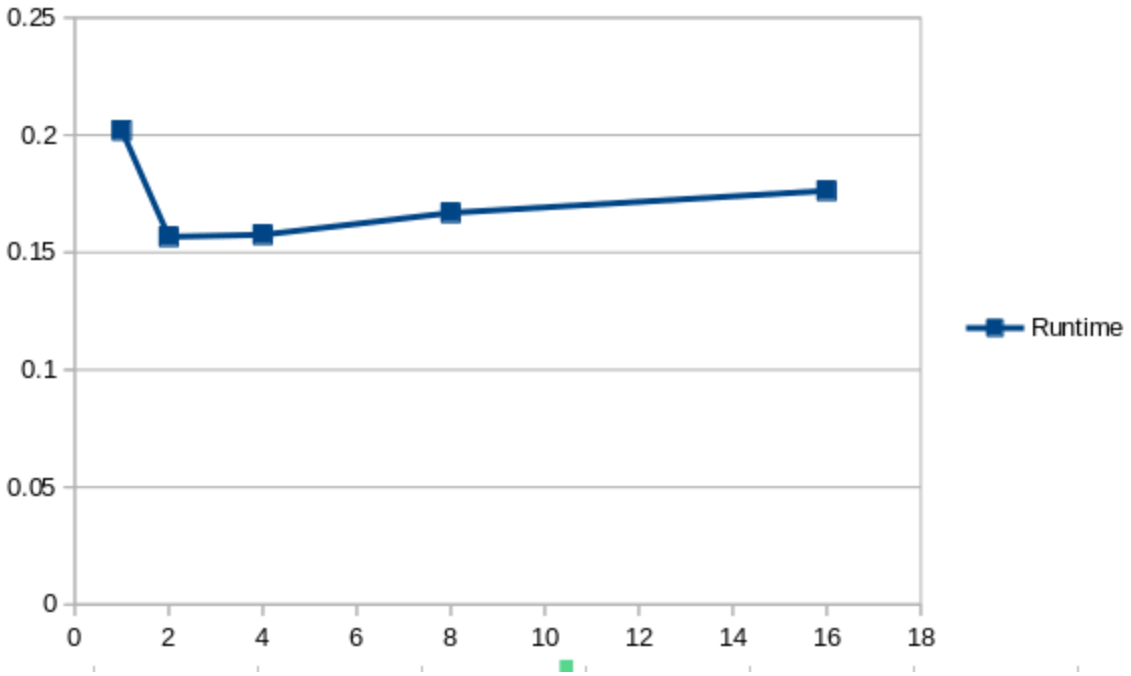
Number of Threads (N) = **16**

Serial Runtime (Ts) = **0.201965s**

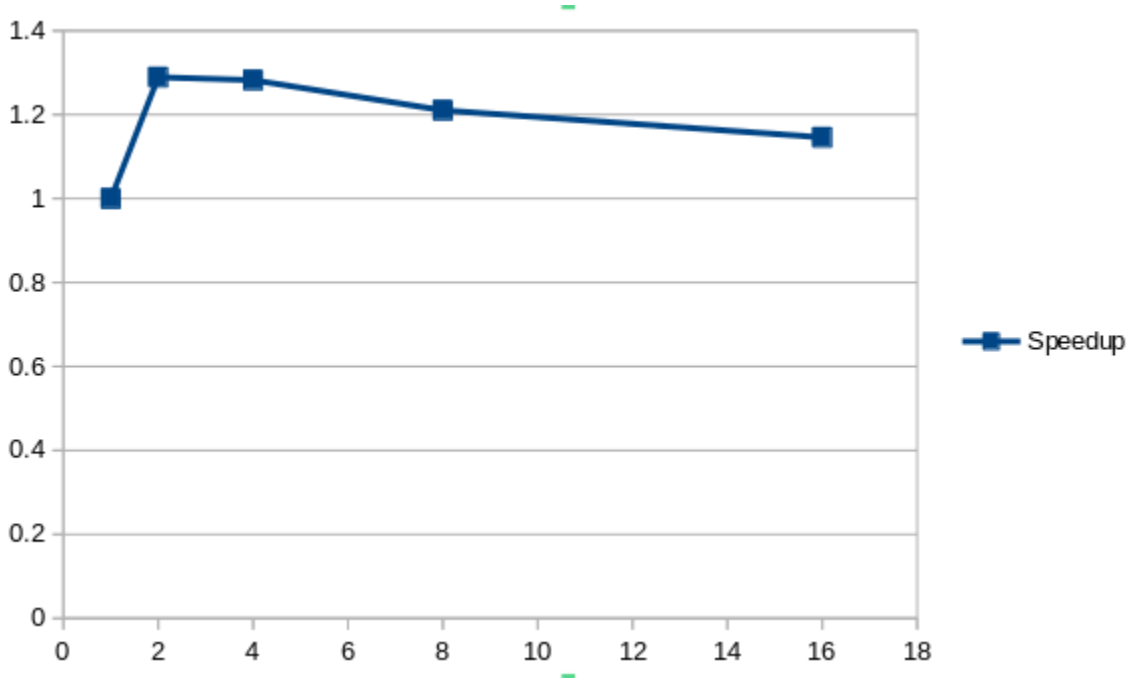
Parallel Runtime (Tp) = **0.176268s**

Speed up (S) = Ts / Tp = 0.201965 / 0.176268 = **1.14578369301**

Efficiency (E) = S / N = 1.14578369301 / 16 = **0.07161148081**



This graph highlights the runtime using an increasing number of threads. From this we see that the program runs faster using threads, but quickly runs into overhead and slows down.



This graph highlights how the Speedup initially has a sharp raise when using threads, but quickly flattens out.

b)

Optimal Thread Count: 2

Based on the speed up and run times found using threads, we found that using 2 threads provided the most optimal results. This is likely due to a large amount of overhead.

c)

Program scaling proportionally:

Number of Threads (N) = **1**

Program Size = **100,000**

Runtime = **0.201965s**

Number of Threads (N) = **2**

Program Size = **200,000**

Runtime = **0.509656s**

Number of Threads (N) = **4**

Program Size = **400,000**

Runtime = **1.629000s**

Number of Threads (N) = **8**

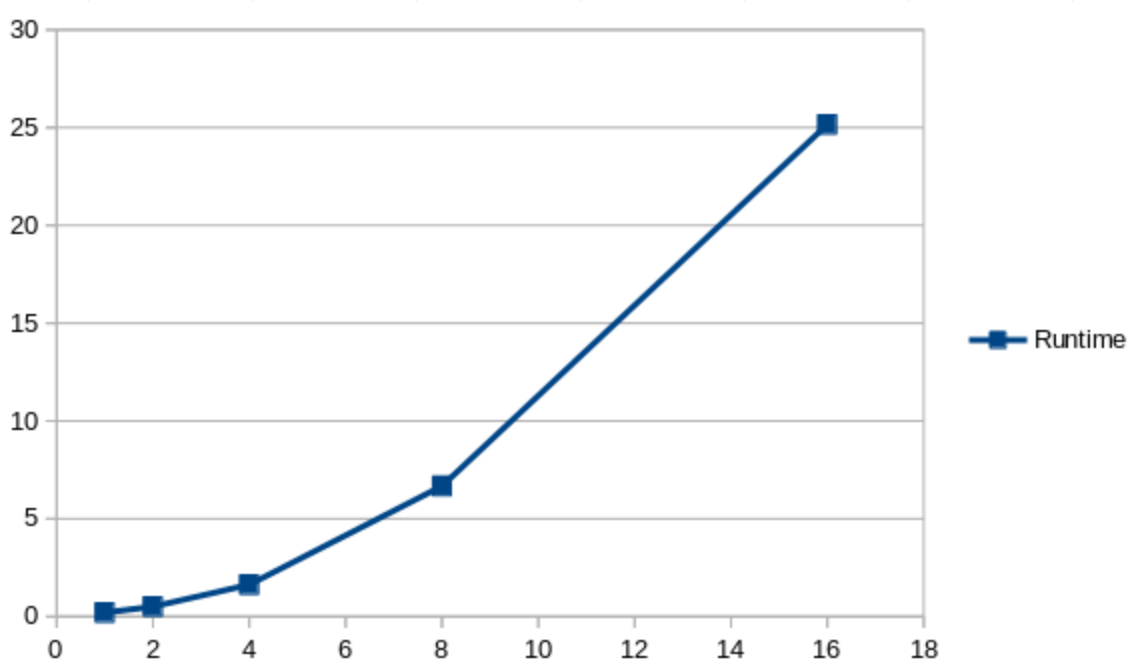
Program Size = **800,000**

Runtime = **6.672177s**

Number of Threads (N) = **16**

Program Size = **1,600,000**

Runtime = **25.159900s**



This graph highlights how when scaling the size of the process with the number of threads, there is higher runtime due to more overhead and a larger process size.

Analysis of MPI

a)

Number of Processors (N) = **2**

Serial Runtime (Ts) = **0.201965s**

Parallel Runtime (Tp) = **0.200378s**

Speed up (S) = Ts / Tp = 0.201965 / 0.200378 = **1.007920031**

Efficiency (E) = S / N = 1.007920031 / 2 = **0.5039600155**

Number of Processors (N) = **4**

Serial Runtime (Ts) = **0.201965s**

Parallel Runtime (Tp) = **0.148118s**

Speed up (S) = Ts / Tp = 0.201965 / 0.148118 = **1.363541231**

Efficiency (E) = S / N = 1.363541231/ 4 = **0.3408853077**

Number of Processors (N) = **8**

Serial Runtime (Ts) = **0.201965s**

Parallel Runtime (Tp) = **0.156685s**

Speed up (S) = Ts / Tp = 0.201965 / 0.156685 = **1.288987459**

Efficiency (E) = S / N = 1.288987459 / 8 = **0.1611234324**

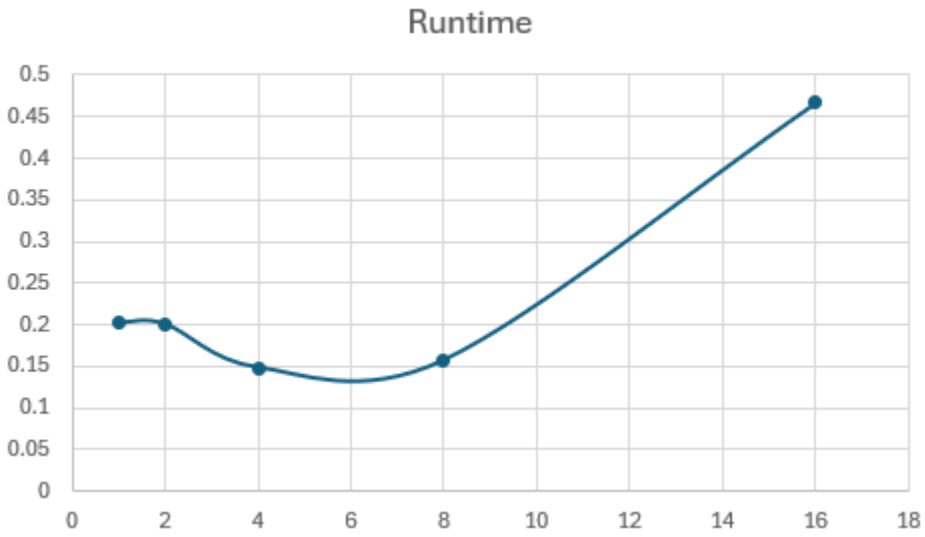
Number of Processors (N) = **16**

Serial Runtime (Ts) = **0.201965s**

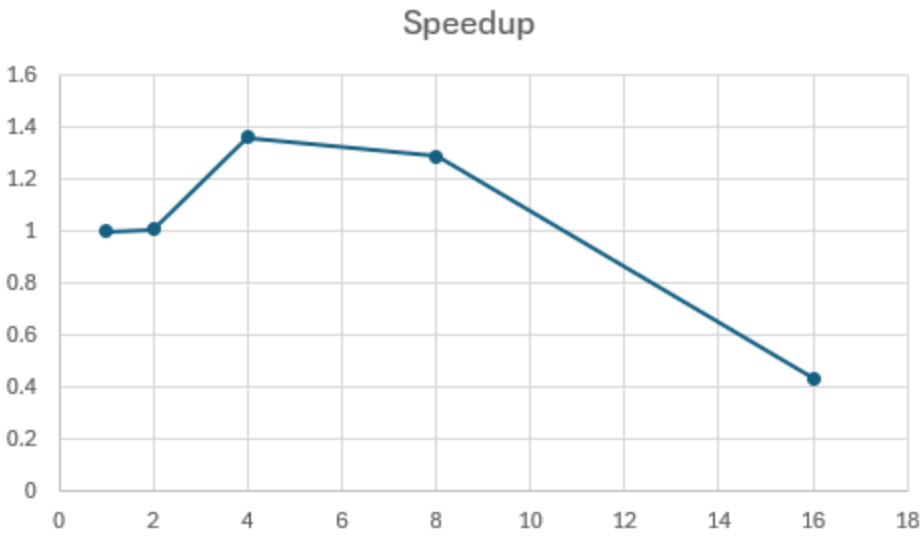
Parallel Runtime (Tp) = **0.466245s**

Speed up (S) = Ts / Tp = 0.201965 / 0.466245 = **0.4331735461**

Efficiency (E) = S / N = 0.4331735461 / 16 = **0.02707334663**



This graph highlights the runtime using an increasing number of processors. From this we see that the program runs faster using more processors, but begins to veer off as more overhead is introduced.



This graph highlights how the Speedup initially has a sharp raise when using more processors, but eventually takes a deep dive as overhead starts being a bigger problem.

b)

Optimal Processor Count: 4

Based on the speed up and run times found using MPI, we found that using 4 processors provided the most optimal results. This is likely due to a large amount of overhead present using too many processors.

c)

Program scaling proportionally:

Number of Processors (N) = **1**

Program Size = **100,000**

Runtime = **0.201965s**

Number of Processors (N) = **2**

Program Size = **200,000**

Runtime = **0.399808s**

Number of Processors (N) = **4**

Program Size = **400,000**

Runtime = **0.722416s**

Number of Processors (N) = **8**

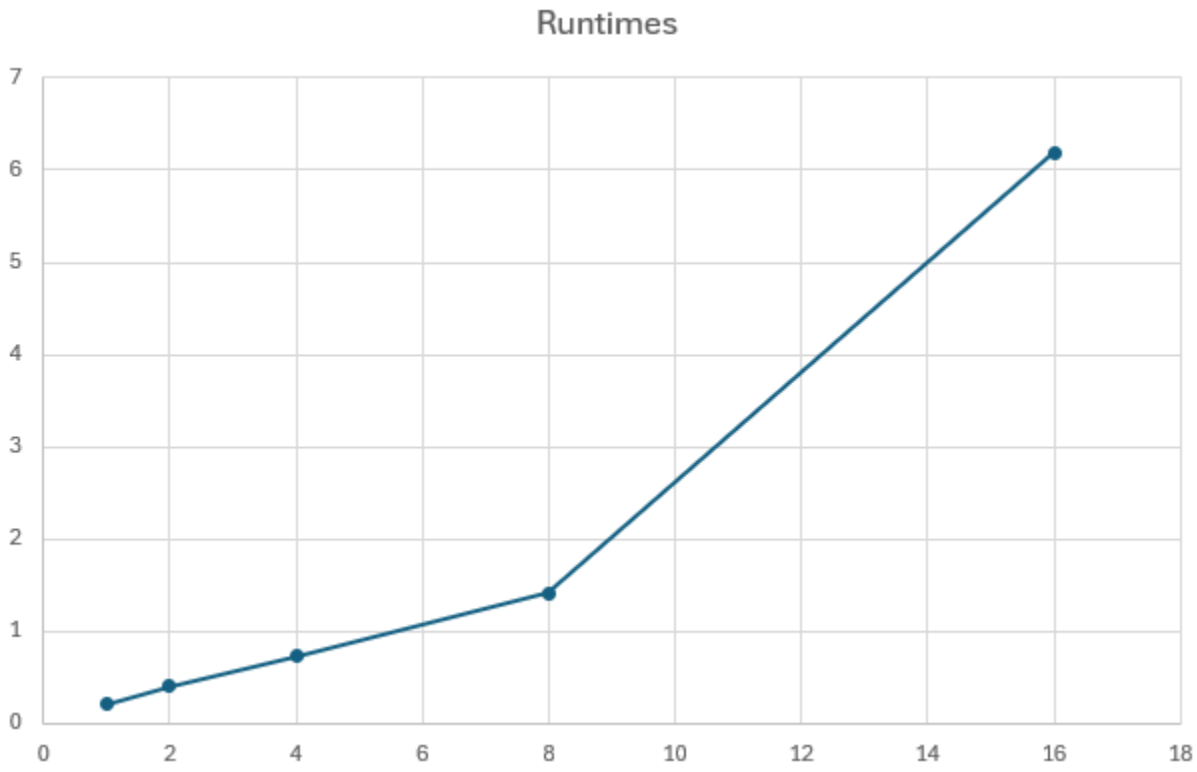
Program Size = **800,000**

Runtime = **1.410944s**

Number of Processors (N) = **16**

Program Size = **1,600,000**

Runtime = **6.192180s**



This graph highlights how when scaling the size of the process with the number of processors, there is higher runtime due to more overhead and a larger process size.