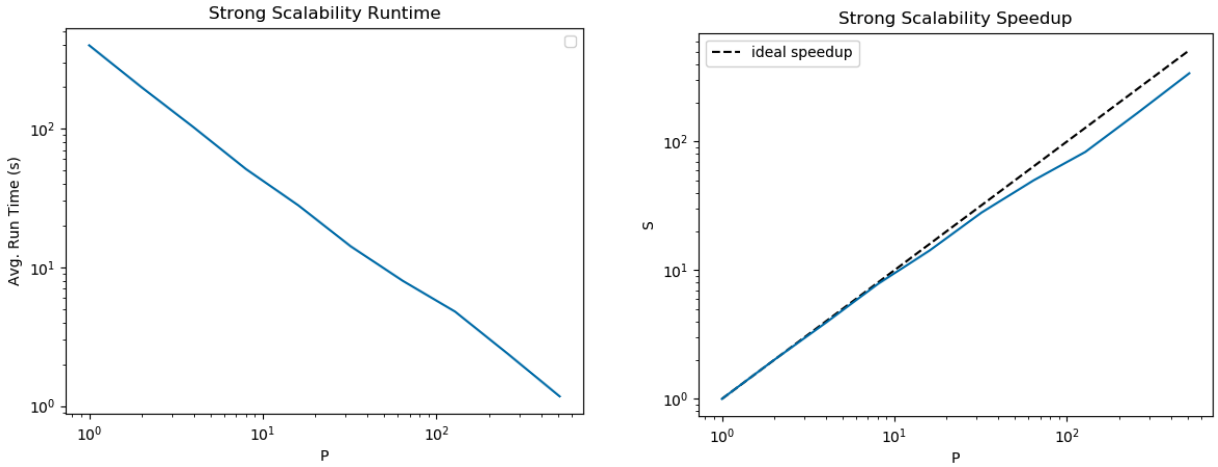
Project 4 Analysis

Brian Lee and Cara Dunnavant

9 December, 2021

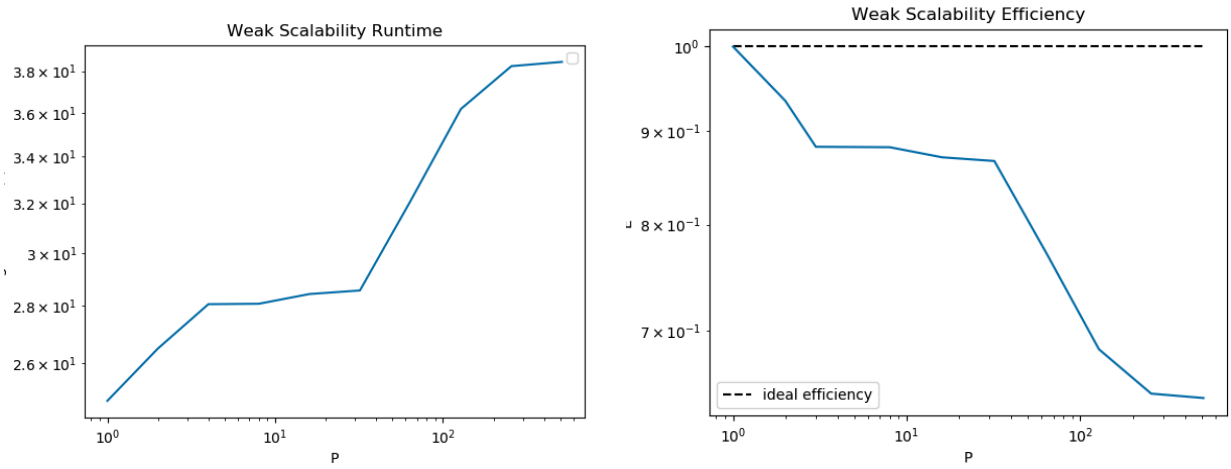
1 Strong Scalability Study



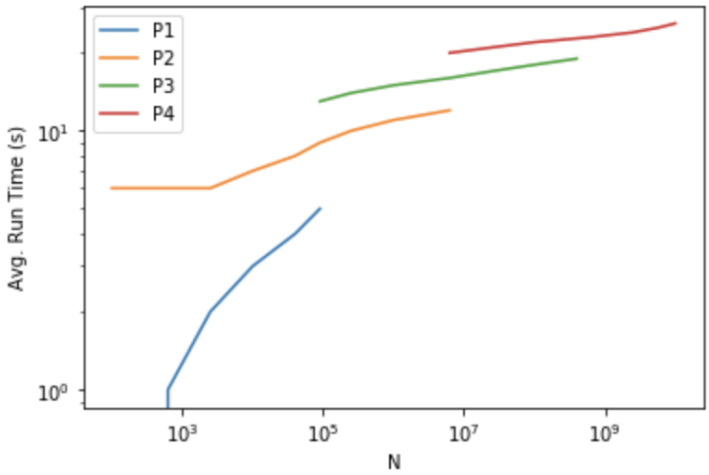
We ran 3 trials per experiment. A strong scaling study holds the total work constant as the number of processors grows. In our opinion, this problem exhibits very good strong scalability because the average runtime about cuts in half as you double the number of processors. The speedup of the study is very close to the ideal speedup as the number of processors increases. This behavior is observed because each experiment we double the number of processors on the same problem size.

2 Weak Scalability Study

We ran 3 trials per experiment. A weak scaling study keeps the work-per-processor constant as the number of processors grows. Ideally, we would see the runtime stay relatively constant as the problem size and number of processors are scaled linearly. As you can see, from $P = 4$ through $P = 32$, the runtime stays almost exactly constant. From $P = 64$ to $P = 256$, the runtime increases by a few seconds each time. However, the runtimes $P = 256$ and $P = 512$ are also almost exactly the same. Overall, the highest increase in runtime from one processor to the next is only a few seconds, so this project exhibits good weak scalability.



3 Comparing Project Runtimes



We estimated how many CPU hours it would take to run 100 simulation iterations with $n = 100,000$ for our previous projects. We used linear regression to approximate a runtime and then converted that value into CPU hours.

- Project 1: -0.08456 + 0.00238 (100,000) = 237.9154s
  - 237.9154s = 0.06608761h x 1024 CPUs = 67.67371 CPU hours
- Project 2: -0.0179865 + 0.0001011 (100,000) = 10.09201s
  - 10.09201s = 0.002803336h x 1024 CPUs = 2.870616 CPU hours
- Project 3 ($P = 64$): -2.595e-02 + 1.668e-05 (100,000) = 1.64205s
  - 1.64205s = 0.000456125 x 1024 CPUs = 0.467072 CPU hours

For Project 4, 65.166822s = 0.01810189h x 1024 CPUs = 18.53634 CPU hours.

Over the course of the semester, we learned how to optimize efficiency. By running our serial Python project and comparing it to our MPI C project, we realized that by parallelizing our code and dividing up work between processors you can solve larger and larger problems more quickly.

When working on future projects in school and in our careers, we can definitely apply topics we learned in this class. We could use Cython, MPI for Python, MPI or OpenMP, etc. to solve large problems in data analytics.

4 Additional Analysis Tasks

Not applicable since group of 2.

5 Project Working Style Evaluation

Done individually on Canvas.