# Convert this docx to PDF before submitting

**Should you use granger1 or granger2?**[**https://fxlin.github.io/p2-concurrency/#which-server**](https://fxlin.github.io/p2-concurrency/#which-server) **For p2, this is a soft suggestion, not a hard requirement.**

# Scalability

## 0. Reproduce the benchmarks

Repeat what has been described in the project description.

* Attach a scalability plot (ONLY the one showing all the program versions) you generated. (10)  
  To generate the plot, you can use any tool. There’s a boilerplate script (p2-concurrency/scripts/plot.py) that may help; but you are not required to use it.

A graph with lines and dots

Description automatically generated with medium confidence

* Compare your observation with the given results. What are the same? What are different? (10)
  + The observations are very similar to each other. The general shape of each line is very similar for both the generated graphs and the given results; that is, in both graphs, the -pm, -pml, and -pmla generally follow a positive linear slope and baseline and -p generally plateau. There are a couple of differences between both graphs. The -p program seems to drop less between the first two threadNum in the generated graph than it does in the given graph and there seems to have less noise in the generated graph. The -pm programs seem to have a higher throughput than the -pml in the generated graph than the given graph. The -pmla program seems to generally have a higher slope in the generated graph than the given graph.
* Explain your observation. (10)
  + The graphs are similar because the benefits of implementing the improvements is shared regardless of the settings or configuration of the machines. In other words, we will always see a general linear positive trend for -pmla unless there is a DRASTIC change in the configuration and settings of the machine when the given results were created and when the generated results were created. At the same the time, the difference between the two graphs are explained by the simple differences between the configuration and settings of the machine that generated the generated results versus the machine that generated the given results.

## 1. The unfinished scalability quest

How does the program scale to more than 8 cores?

* Attach a scalability plot (ONLY the one showing all the program versions) with core count = {1 2 4 6 8 10 12 16 20}. You may want to tweak run.sh and plot.py (10)
* Describe and explain your observation. (10)
* If there's any scalability bottleneck, profile the execution with VTune (e.g. consider trying VTune's "microarchitecture exploration"). Can you make the program scale better? If so, show your code and profiling results; if not, reason about possible bottlenecks. (5)

## 