Assumption: This lecture is based on PBSPro scheduler. While gnu-parallel is similar across different schedulers, the concept of job array might differ. For SLURM job array, instructors can modify the slide with information from https://slurm.schedmd.com/job_array.html

Overall concepts

Adding more computers to existing resources (perhaps only one computer at the beginning) to help solve a problem is called scaling. There are two typical issues in solving a problem:

- Existing resources are not enough to handle the current problem.
- Existing resources are just enough to handle the current problem, but the problem needs to be solved at larger scales.

Adding computers to handle the first issue is called strong scaling. Adding computers and increasing the size of the problem to address the second issue is called weak scaling.

Speedup

We expect (hope) the program to run faster when more computing resources are added. The rate of improvement in performance (or reduction in run time) is called speed up, and is calculated using the following equation:

$$S = \frac{t_1}{t_N}$$

With S represents the speedup, t_1 is the run time of the program with 1 processor, and t_N is the run time of the program with N processors.

Scaling Limitation

There is a limit to how much resources can be added to improve performance.
Strong or weak scalings are limited by proportion of serial (non-parallelizable) code/task within a program/workflow.

Strong scaling: Amdahl's Law
Let's call s the proportion of code that cannot be parallelized. Hence $1 - s$ represents the remainder proportion of code that can be parallelized. The speedup in strong scaling can then be calculated as:

$$S = \frac{1}{(s + \frac{1-s}{N})}$$

Weak scaling: Gustafson's Law

Since in weak scaling we scale up the problem as well as resources, we need to consider scaled speedup (SS). Let's call s the proportion of code that cannot be parallelized. Hence $1 - s$ represents the remainder proportion of code that can be parallelized. The scaled speedup in strong scaling can then be calculated as:

$$SS = s + (1 - s) \times N$$

Scaling in a cluster: job-array and gnu-parallel

This is a straightforward parallelization approach that focuses on running multiple instances of a program in parallel. This can be done via job-array (scheduler's features) or via support of an external library called gnu-parallel. No code modification is needed in this approach.

In job array, the scheduler (PBS) will submit a number of jobs from a single job submission template. An environmental variable (PBS_ARRAY_INDEX) is introduced and serves as a job identifier and also can be used in the job script to help managing/distributing work. The job submission's resource request is designed for a single job.

In gnu-parallel, a single job is submitted, and the gnu-parallel library will help managing/distributing work by utilizing all resource requests. In other words, resource request should be made in a scale-up manner.