Introduction to Parallel Computing

Jinzhen Wang
jwang96@charlotte.edu

Department of Computer Science
UNC Charlotte

Learning Outcome

At this end of this lecture, you will be able to

- Measure the performance of parallelism.
- Articulate why not all problems will benefit from parallelism.
- Articulate the difference between weak and strong scaling.
- Articulate the difference between horizontal and vertical scaling.

Why not just measure time?

The end goal of parallelism is to decrease the time of some calculation.
One can measure time and report time improvements.

- Often raw time is hard to make sense of.
- Not all applications use time as the metrics.
  - Video games: FPS
- Makes it hard to get insight across different machines.

Measure of Success: speedup

Definition

$S(N) = \frac{T_1}{T_N}$ where T_1 is sequential time and T_N is time on N processors.
You should use as T_1 the best sequential time you can get.

Example

A sequential application taking 100 minutes, being executed in parallel on 4 processors to take 50 minutes has a speedup on 4 processors:

$$S(4) = \frac{100}{50} = 2$$

Speed is the key to measure how well your parallel computing works.

What speedup is good?

Linear speedup

Ideally, you would want to achieve linear speedup: $S(N) = N$.
If you achieve linear speedup (or close), the code is said to scale well.
It is ideal, you almost never get $S(N) = N$

Super linear speedup

You can sometimes get $S(N) > N$.
It is usually an anomaly that comes from the fact that the parallel algorithm does less computation than the sequential one in that particular case.

- if you double the population of a city, you increase the productivity of the city by more than double (roughly 115%).
- if a problem can be divided into pieces that fit entirely into a CPU's cache, a significant speedup can be observed.

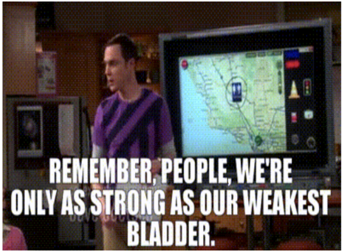
Why wouldn't all computation scale?

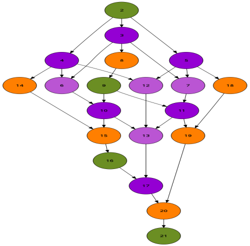
Inherently sequential computation

- Two ovens won't cook a cake faster.
- Combining two half-cups of water at 50°C will not result in one cup of water at 100°C.

Why wouldn't all computation scale?

Communications/Synchronization





How much can a computation scale ?

Strong Scaling

In a strong scaling experiment, you pick one particular test case that you want to solve. And you see how fast that one can be solved.

+ The best kind of experiment
- Scaling depends on problem size
- How to pick a representative problem?

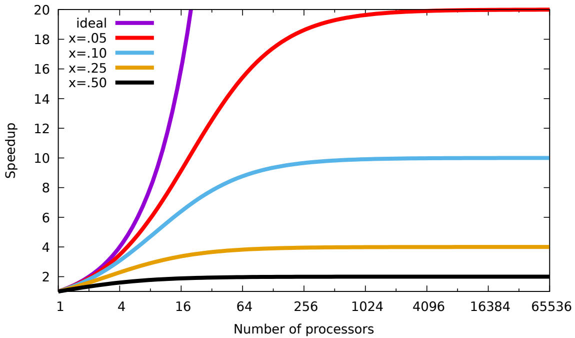
How much can a computation scale ?

Amdahl's Law

- A fraction x of the computation that is serial.
- The rest $(1-x)$ is completely parallel

Time on N computational units
$T(N) \geq T(1)(x + \frac{1-x}{N})$

Speedup on N computational units
$S(N) \leq \frac{1}{x}$



How much can a computation scale?

Weak Scaling

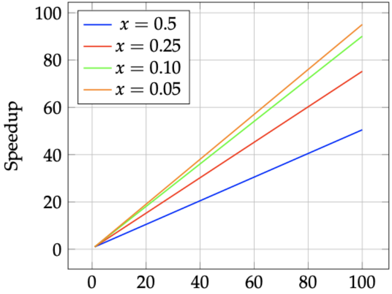
In a weak scaling experiment, you increase the volume of computation (in term of complexity) with the amount of resource. (Usually report time.)

+ Often easier to achieve scaling
- Not all problems have increasing size

How much can a computation scale?

Gustafson's Law

- The sequential part x is a one time fixed overhead **independent** of the amount of work
- The rest of the computation scales linearly
- Then you can process a dataset of size N with N processors in the same amount of time 1 processor would a dataset of size 1.
- $S(N) = x + N(1 - x)$. with x the fraction the overhead represents on 1 computational unit.



Examples of Strong/Weak Scaling?

- Strong Scaling
    - Decode a passcode
    - Image processing
- Weak Scaling
    - Weather forecast
    - Protein folding

Scaling systems

Metric: Latency

Time it takes to perform one computation.

Metric: Bandwidth

Rate at which one can perform computation

Scaling systems

Metric: Latency

Time it takes to perform one computation.

Vertical Scaling

Vertically scaling adds ressources into a single node to increase its computational speed.
The goal is often to improve the latency of the system

Metric: Bandwidth

Rate at which one can perform computation

Horizontal Scaling

Horizontal scaling adds ressources as additional nodes to increase the computational speed.
The goal is often to improve the bandwidth of the system

Scaling systems

Metric: Latency

Time it takes to perform one computation.

Metric: Bandwidth

Rate at which one can perform computation

Vertical Scaling

Vertically scaling adds ressources into a single node to increase its computational speed.
The goal is often to improve the latency of the system

- Increasing speed limits will decrease commute time.
- Using better servers to decrease query time.

Horizontal Scaling

Horizontal scaling adds ressources as additional nodes to increase the computational speed.
The goal is often to improve the bandwidth of the system

- Adding a lane to the highway will increase flow.
- Adding servers to process more clients.

External

On parallel computing:

- Tim Mattson on why parallel computing: `https://www.youtube.com/watch?v=cMWGeJyrc9w`
- Tim Mattson on concurrency and parallelism: `https://www.youtube.com/watch?v=6jFkNjhJ-Z4`
- Wikipedia on parallel computing `https://en.wikipedia.org/wiki/Parallel_computing`
- Amdahl's Law `https://en.wikipedia.org/wiki/Amdahl's_law`
- "Parallel Programming in MPI and OpenMP" by Victor Eijkhout `https://bitbucket.org/VictorEijkhout/parallel-computing-book/raw/94518afb68da8f5af7104c0ce2343a8dc04f3a16/EijkhoutParComp.pdf`

Books that could be useful throughout the semester (most e-copy available through the library):

- Sushil K. Prasad, AnshulGupta, Arnold Rosenberg, Alan Sussman, and Charles Weems. Topics in Parallel and Distributed Computing. Enhancing the Undergraduate Curriculum: Performance, Concurrency, and Programming on Modern Platforms. Springer 2018. `https://link.springer.com/book/10.1007/978-3-319-93109-8`
- Barbara Chapman, Gabriele Jost, and Ruud van der Pas. Using OpenMP. Portable Shared Memory Parallel Programming. MIT Press. 2007. Available through the library at `https://librarylink.uncc.edu/login?url=http://ieeexplore.ieee.org/servlet/opac?bknumber=6267237`
- Using MPI, 3rd edition. William Gropp, Ewing Lusk and Anthony Skjellum. MIT Press. Available through the library at `https://librarylink.uncc.edu/login?url=http://ieeexplore.ieee.org/xpl/bkabstractplus.jsp?bkn=6981847`