Introduction to Parallel Computing

Jinzhen Wang
jwang96@charlotte.edu

Department of Computer Science
UNC Charlotte

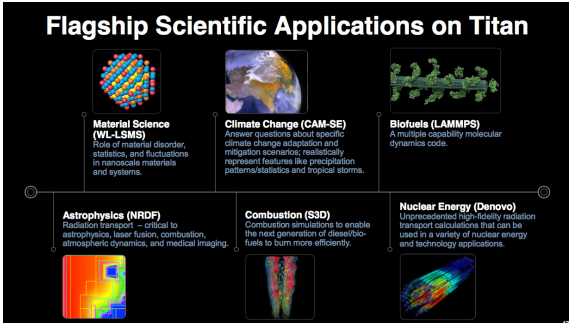
Learning Outcome

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

- Articulate the difference between concurrency and parallelism.
- Explain why parallelism has become critical.
- Name different types of parallelism.

Some Background

- "Supercomputers" for scientific computing



Some Background

- Vendors in supercomputers

Some Background

- Vendors in supercomputers

Wanna work for any of those?

The end of Dennard scaling (Moore's Law)

A single CPU can only get so powerful, until ...

Dennard 1974

"As transistors get smaller their power density stays constant."
When transistors shrink, they use less power.
So frequency can be increased.

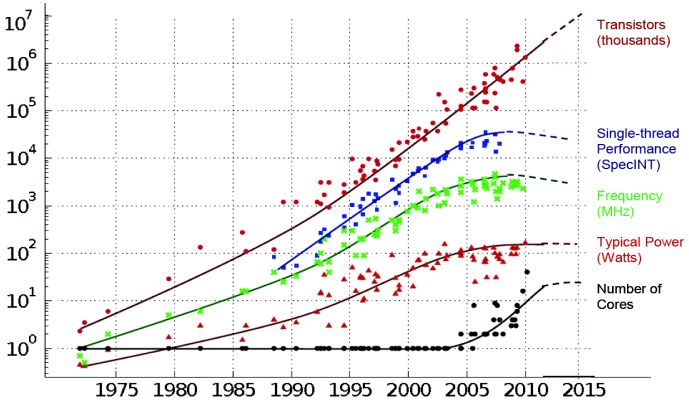
Computer Scientist in the 80's and 90's

"Why care about code efficiency? Chips will be twice faster in 18 months."
Emphasis was on programmer productivity, rather than performance.

Dennard Scaling no longer really works because of Physics reasons (mostly power leakage).

The end of Dennard scaling (Moore's Law)

A single CPU can only get so powerful, until ...



Why Parallel Computing

- These big machines look awesome, But, Why Should I Learn Parallel Computing?
- At beginning, it is because single CPUs are not powerful enough
  - So, more CPUs ...
- But later
  - GPU ...
  - more GPUs ...
  - Single machine is not enough ...
  - Single data center is not enough ...
- Everyone needs parallel computing now!

Why Parallelism?

Main memory limits

- ∼8GB on a laptop.
- ∼64GB on a desktop.
- up to 4TB on a server.

So if your problem needs more than that, there are about 4 options:

- Don't compute it
- Shrink memory usage somehow
- Swap using disk
- Use multiple computers

Why Parallelism?

Time limits

There is only so much one core can compute. You might not want to wait 3 months for matching two databases!
Some tasks have a target computation time.

- Weather forecast
- Google search
- Video game

Types of computational parallelism

Circuit level parallelism

Bits of an instructions are decoded in parallel. With SIMD, integers can be added simultaneously.

Types of computational parallelism

Instruction level parallelism

Modern processors can execute multiple instructions per cycle.

Types of computational parallelism

Shared memory parallelism

Different cores all access the same memory space.
Here often the problem is to make sure that simultaneous execution make sense.

Types of computational parallelism

Distributed memory parallelism

With multiple nodes, each has its own memory space.
The problem is often to reduce the amount of communication that the nodes must exchange.

Types of computational parallelism

Accelerators

Many systems have devices one can communicate with that provide additional processing power. Often they are parallel themselves.
Modern accelerators: GPU, Xeon Phi, FPGA.

Concurrency Vs Parallelism

Concurrency

Processes are concurrent when they can happen in different order **relatively to one another**.
It often needs to **concurrency control** to make sure that the execution make sense.
The problem with concurrency is to obtain a **correct** execution of the application.

Concurrency Vs Parallelism

Concurrency

Processes are concurrent when they can happen in different order **relatively to one another**.
It often needs to **concurrency control** to make sure that the execution make sense.
The problem with concurrency is to obtain a **correct** execution of the application.

Example

- The US postal system
- Laundry in a household

Concurrency Vs Parallelism

Parallelism

Processes are parallel when they run at the same time on different execution units.
It uses techniques that **expose** computation that can be executed simultaneously and **organize** execution units in doing them as fast as possible.
The purpose is to **extract more performance** out of the execution.

Concurrency Vs Parallelism

Parallelism

Processes are parallel when they run at the same time on different execution units.
It uses techniques that **expose** computation that can be executed simultaneously and **organize** execution units in doing them as fast as possible.
The purpose is to **extract more performance** out of the execution.

Example

- Team building a house
- Cashiers at a grocery store

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`

Books that could be useful:

- 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.
- Barbara Chapman, Gabriele Jost, and Ruud van der Pas. Using OpenMP. Portable Shared Memory Parallel Programming. MIT Press. 2007.
- Using MPI, 3rd edition. William Gropp, Ewing Lusk and Anthony Skjellum. MIT Press.