Parallel Processing

Why?

- Compute stuff faster.

What?

- Do more than one thing at a time.

Computers are pretty fast

- This laptop has a 2 GHz processor.
- So it can do around a billion “operations” per second.

Times @ 1B ops/sec

O(?)	n = 10	100	1k	1M	1B
n	10 ns	100 ns	1 us	1 ms	1 s
n^2	100 ns	10 us	1 ms	15 min	30 yrs
n^3	1 us	1 ms	1 s	forever	forever
2^n	1 us	forever	forever	forever	forever

What does parallelism get us?

- Quad-core machine = x4 ops
- SIMD instructions = x4 ops
- A high end GPU = x100 ops
- A cluster of 100 machines = x100 ops

So… let’s use 100 machines, quad core processors, and SIMD. That’s x1600 ops.

Times @ 1600 * 1B = 1.6T ops/sec

O(?)	n = 10	100	1k	1M	1B
n	<1 ns	<1 ns	1 ns	1 us	1 ms
n^2	<1 ns	10 ns	1 us	1s	a week
n^3	1 ns	1 us	1 ms	forever	forever
2^n	1 ns	forever	forever	forever	forever

For a quadratic algorithm, we got:

- n = 1M, down from 15M to 1s
- n = 1B, down from 30 years to 1 week

In terms of O(?), we don’t win much.

- But, we get a decent improvement for “fast” algorithms.
- With 1000x parallelism, in the same time:
  - O(n): We can process 1000x larger n
  - O(n^2): We can process 30x larger n

Parallel Applications

- Graphics (e.g. GPU)
- HPC (science simulations)
- Data Warehousing
  - This is where we are.
  - Query only.
- “Normal Stuff”
  - Video game physics
  - Running “make” with the -j flag.

“Big Data”

Some situations produce very large data sets.

- Web Search
- CVS discount cards
- License Plate Cameras
- Facebook

How Big? What algorithms can we run?