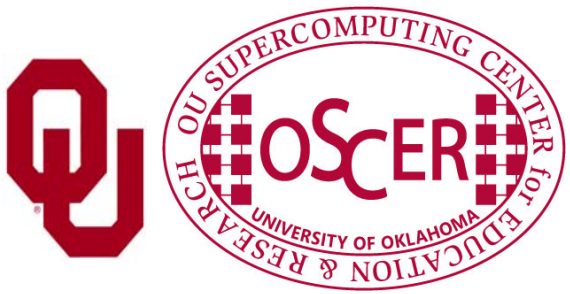
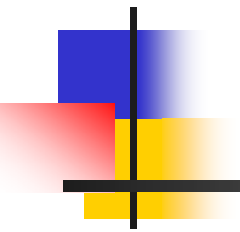
Parallel Programming & Cluster Computing

An Overview of High Performance Computing

Henry Neeman, University of Oklahoma
Paul Gray, University of Northern Iowa

SC08 Education Program's Workshop on Parallel & Cluster computing
Monday October 6 2008





What is Supercomputing?

What is Supercomputing?

Supercomputing is the **biggest, fastest computing right this minute**.

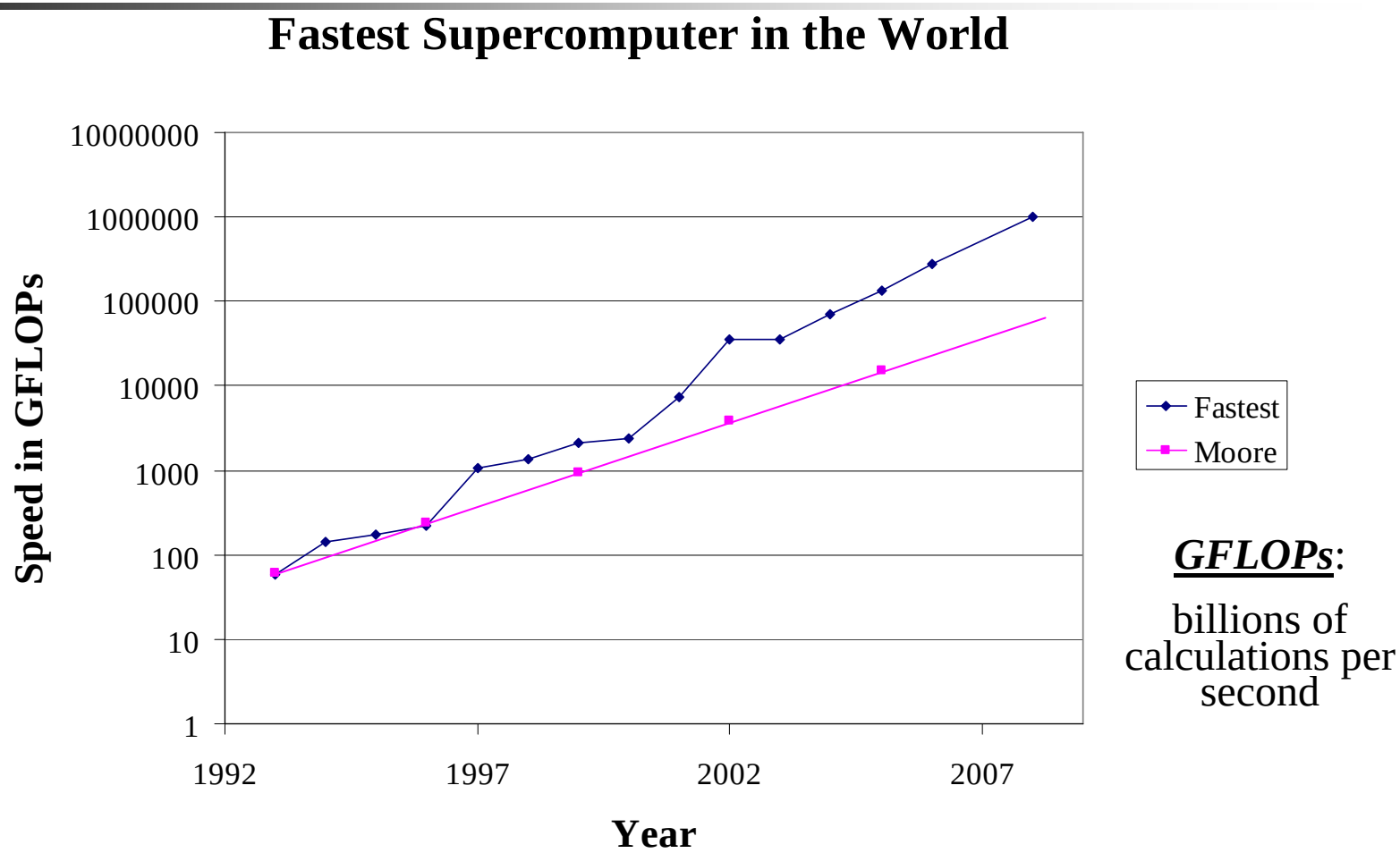
Likewise, a ***supercomputer*** is one of the biggest, fastest computers right this minute.

So, the definition of supercomputing is **constantly changing**.

Rule of Thumb: A supercomputer is typically at least 100 times as powerful as a PC.

Jargon: Supercomputing is also known as ***High Performance Computing*** **(HPC)** or ***High End Computing*** **(HEC)** or ***Cyberinfrastructure*** **(CI).**

Fastest Supercomputer vs. Moore



GFLOPs:

billions of calculations per second

What is Supercomputing About?

Size

Speed

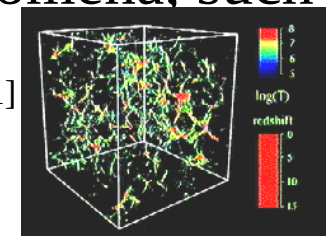
What is Supercomputing About?

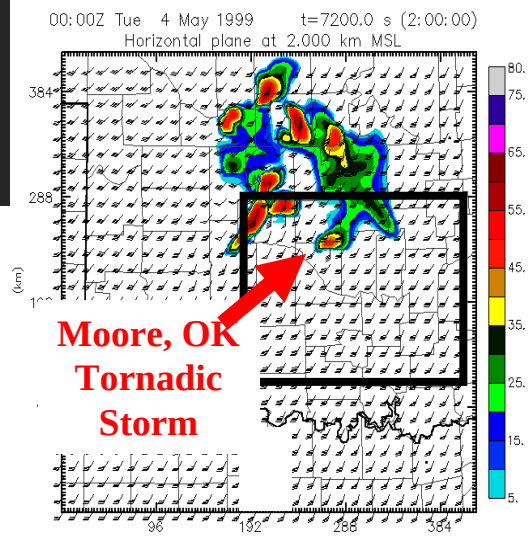
- **Size**: Many problems that are interesting to scientists and engineers **can't fit on a PC** – usually because they need more than a few GB of RAM, or more than a few 100 GB of disk.
- **Speed**: Many problems that are interesting to scientists and engineers would take a very very long time to run on a PC: months or even years. But a problem that would take **a month on a PC** might take only **a few hours on a supercomputer**.

What Is HPC Used For?

- ***Simulation*** of physical phenomena, such as
  - Weather forecasting
  - Galaxy formation
  - Oil reservoir management

[1]

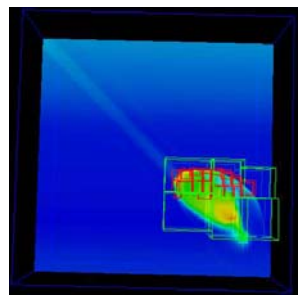




May 3 1999[2]

- ***Data mining***: finding **needles** of information in a **haystack** of data, such as
  - Gene sequencing
  - Signal processing
  - Detecting storms that might produce tornados
- ***Visualization***: turning a vast sea of **data** into **pictures** that a scientist can understand

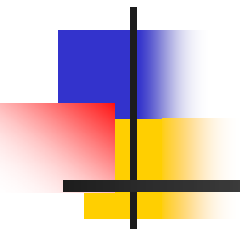
[3]



Supercomputing Issues

- The tyranny of the ***storage hierarchy***
- ***Parallelism***: doing many things at the same time
  - ***Instruction-level parallelism***: doing multiple operations at the same time within a single processor (e.g., add, multiply, load and store simultaneously)
  - ***Multicomputing***: multiple CPUs working on different parts of a problem at the same time
    - ***Shared Memory*** Multithreading
    - ***Distributed*** Multiprocessing
    - ***Hybrid*** Multithreading/Multiprocessing

A Quick Primer on Hardware



Henry's Laptop

Dell Latitude D620[4]

- Pentium 4 Core Duo T2400
  1.83 GHz w/2 MB L2 Cache
  ("Yonah")
- 2 GB (2048 MB)
  667 MHz DDR2 SDRAM
- 100 GB 7200 RPM SATA Hard Drive
- DVD±RW/CD-RW Drive (8x)
- 1 Gbps Ethernet Adapter
- 56 Kbps Phone Modem

Typical Computer Hardware

- Central Processing Unit
- Primary storage
- Secondary storage
- Input devices
- Output devices

Central Processing Unit

Also called ***CPU*** or ***processor***: the "brain"

Parts:

- ***Control Unit***: figures out what to do next -- e.g., whether to load data from memory, or to add two values together, or to store data into memory, or to decide which of two possible actions to perform (***branching***)
- ***Arithmetic/Logic Unit***: performs calculations – e.g., adding, multiplying, checking whether two values are equal
- ***Registers***: where data reside that are **being used right now**

Primary Storage

- ***Main Memory***
  - Also called ***RAM*** ("Random Access Memory")
  - Where data reside when they're **being used by a program that's currently running**
- ***Cache***
  - Small area of much faster memory
  - Where data reside when they're **about to be used** and/or **have been used recently**
- Primary storage is ***volatile***: values in primary storage disappear when the power is turned off.

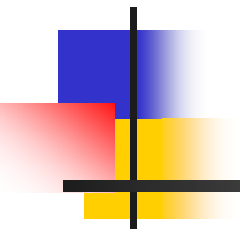
Secondary Storage

- Where data and programs reside that are going to be used **in the future**
- Secondary storage is *non-volatile*: values **don't** disappear when power is turned off.
- Examples: hard disk, CD, DVD, magnetic tape, Zip, Jaz
- Many are ***portable***: can pop out the CD/DVD/tape/Zip/floppy and take it with you

Input/Output

- Input devices – e.g., keyboard, mouse, touchpad, joystick, scanner
- Output devices – e.g., monitor, printer, speakers

The Tyranny of the Storage Hierarchy



The Storage Hierarchy

[5]

Fast, expensive, few

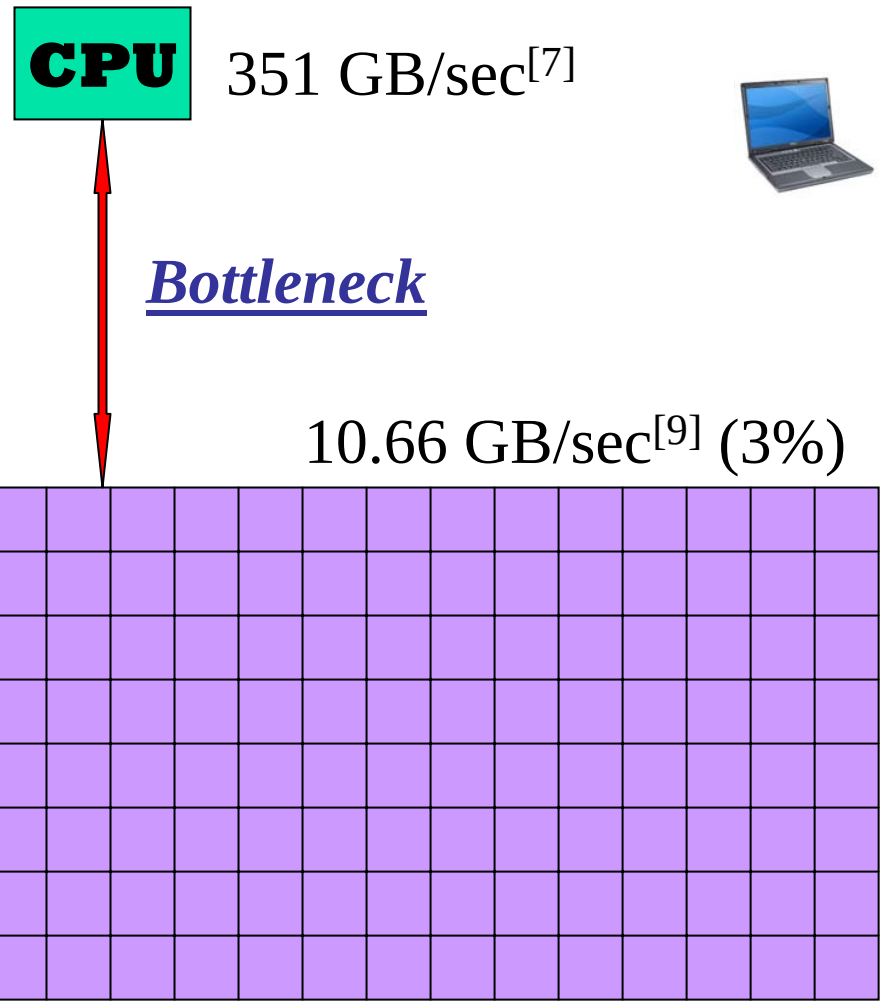
- Registers
- Cache memory
- Main memory (RAM)
- Hard disk
- Removable media (e.g., DVD)
- Internet

Slow, cheap, a lot

[6]

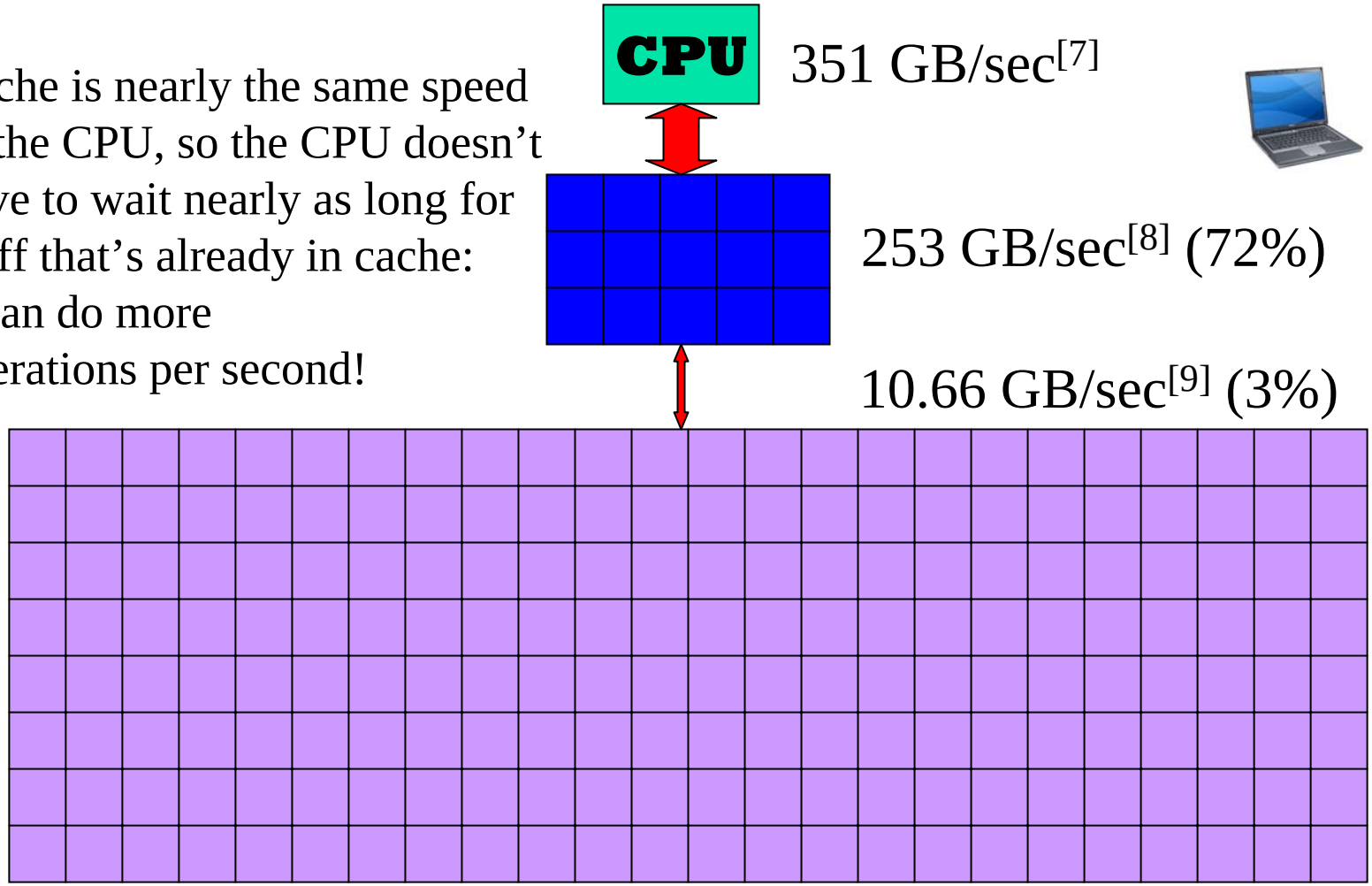
RAM is Slow

The speed of data transfer between Main Memory and the CPU is much slower than the speed of calculating, so the CPU spends most of its time waiting for data to come in or go out.



Why Have Cache?

Cache is nearly the same speed as the CPU, so the CPU doesn't have to wait nearly as long for stuff that's already in cache: it can do more operations per second!



Henry's Laptop, Again

Dell Latitude D620[4]

- Pentium 4 Core Duo T2400
  1.83 GHz w/2 MB L2 Cache
  ("Yonah")
- 2 GB (2048 MB)
  667 MHz DDR2 SDRAM
- 100 GB 7200 RPM SATA Hard Drive
- DVD±RW/CD-RW Drive (8x)
- 1 Gbps Ethernet Adapter
- 56 Kbps Phone Modem

Storage Speed, Size, Cost

Henry's Laptop	Registers (Pentium 4 Core Duo 1.83 GHz)	Cache Memory (L2)	Main Memory (667 MHz DDR2 SDRAM)	Hard Drive (SATA 7200 RPM)	Ethernet (1000 Mbps)	DVD±RW (8x)	Phone Modem (56 Kbps)
Speed (MB/sec) [peak]	359,792[7] (14,640 MFLOP/s*)	259,072 [8]	10,928 [9]	100 [10]	125	10.8 [11]	0.007
Size (MB)	304 bytes** [12]	2	2048	100,000	unlimited	unlimited	unlimited
Cost ($/MB)	–	$46 [13]	$0.14 [13]	$0.0001 [13]	charged per month (typically)	$0.00004 [13]	charged per month (typically)

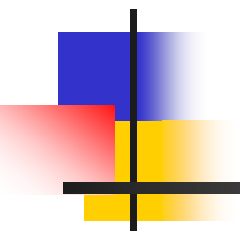
* MFLOP/s: millions of floating point operations per second

** 8 32-bit integer registers, 8 80-bit floating point registers, 8 64-bit MMX integer registers, 8 128-bit floating point XMM registers

Storage Use Strategies

- ***Register reuse***: do a lot of work on the same data before working on new data.
- ***Cache reuse***: the program is much more efficient if all of the data and instructions fit in cache; if not, try to use what's in cache a lot before using anything that isn't in cache.
- ***Data locality***: try to access data that are near each other in memory before data that are far.
- **I/O efficiency**: do a bunch of I/O all at once rather than a little bit at a time; don't mix calculations and I/O.

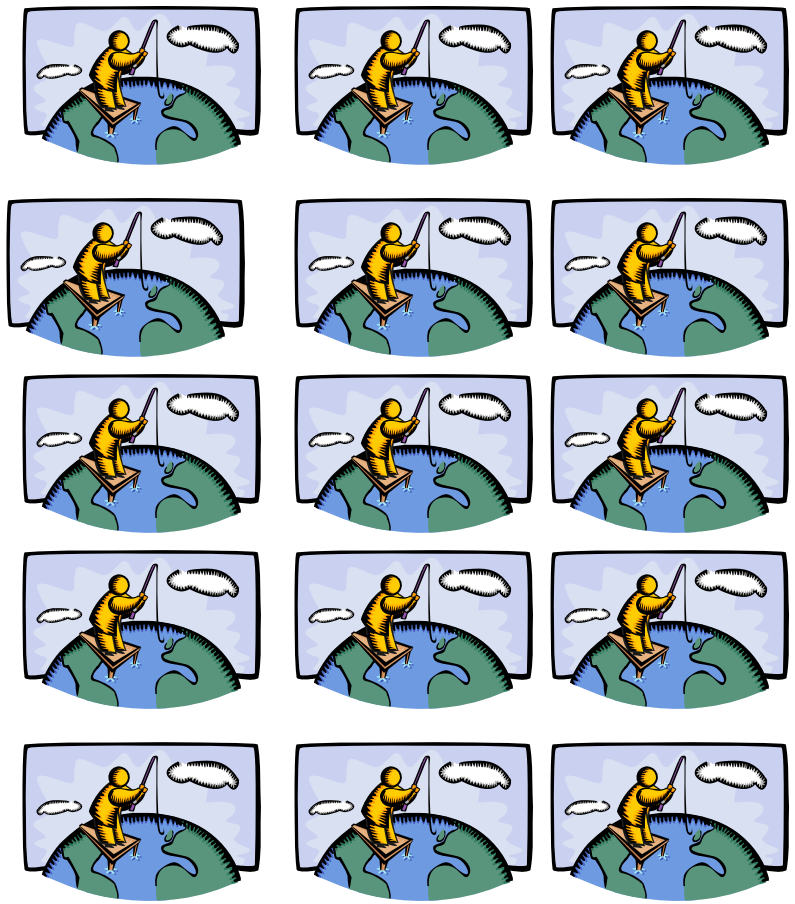
Parallelism



Parallelism

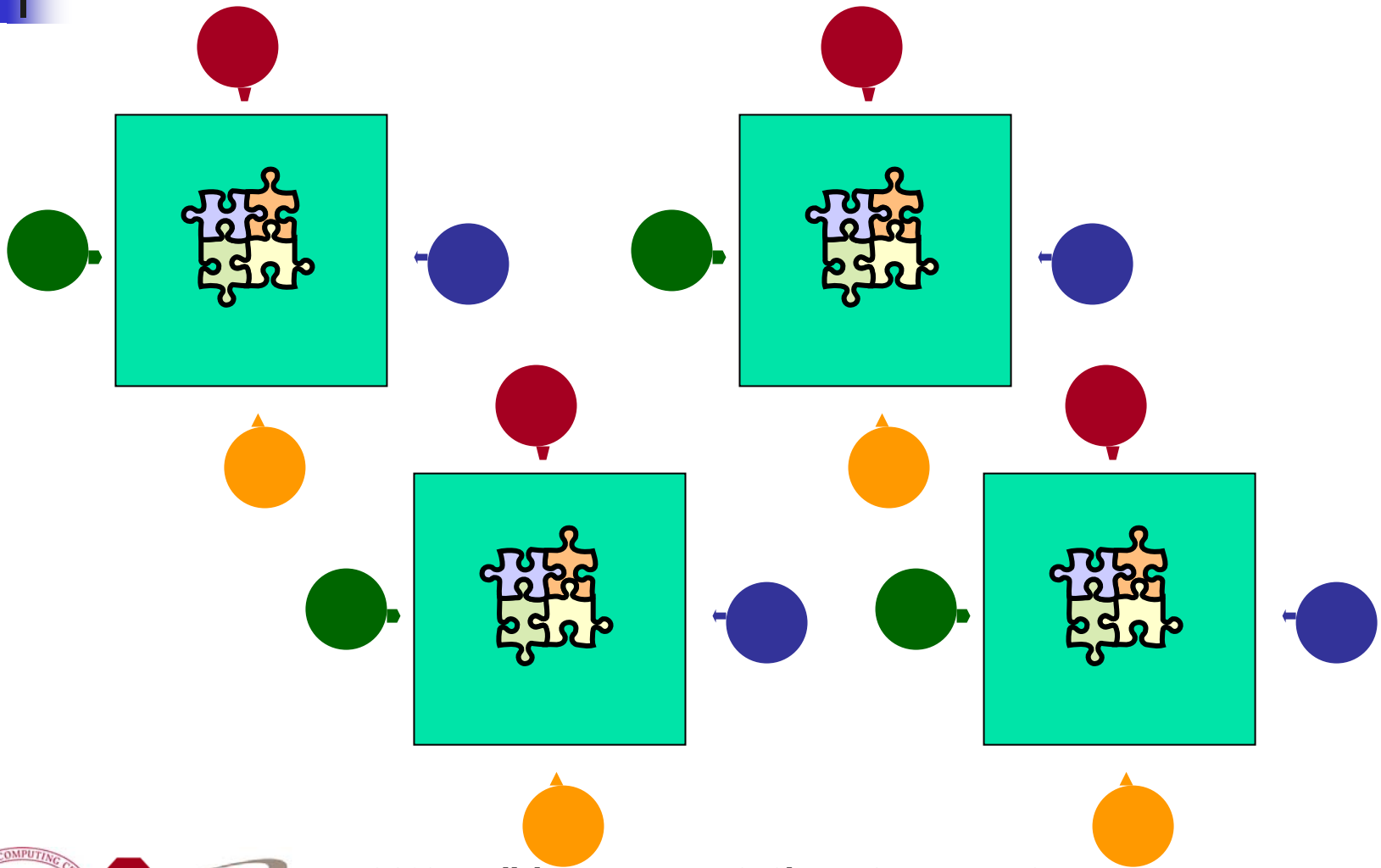
Parallelism means doing multiple things at the same time: you can get more work done in the same time.

Less fish …

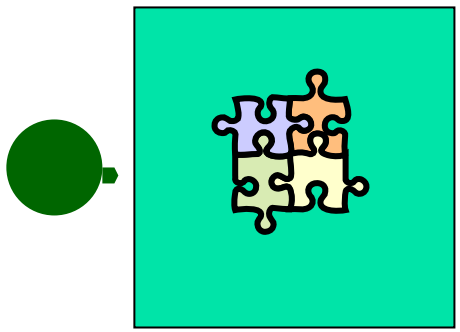


More fish!

The Jigsaw Puzzle Analogy



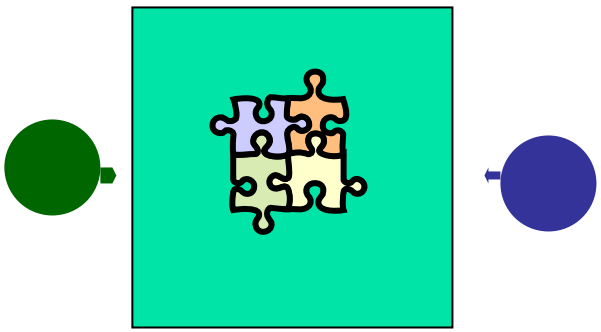
Serial Computing



Suppose you want to do a jigsaw puzzle that has, say, a thousand pieces.

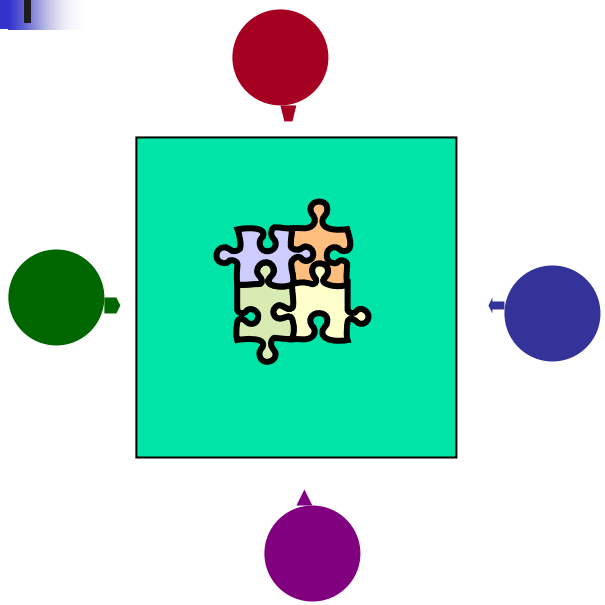
We can imagine that it'll take you a certain amount of time. Let's say that you can put the puzzle together in an hour.

Shared Memory Parallelism



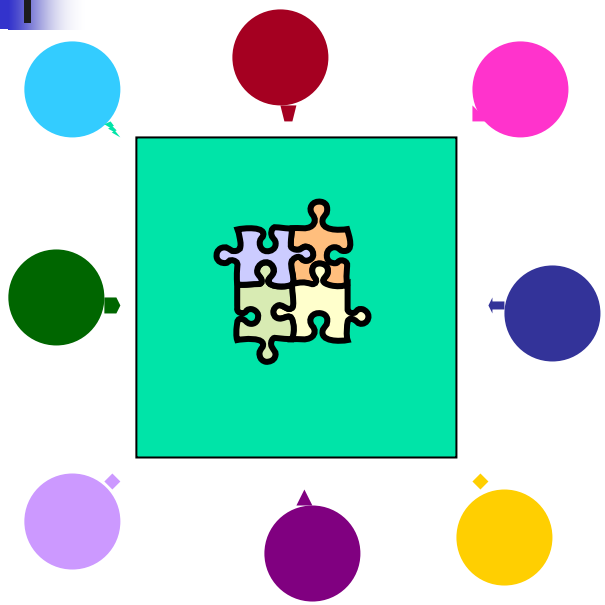
If Scott sits across the table from you, then he can work on his half of the puzzle and you can work on yours. Once in a while, you'll both reach into the pile of pieces at the same time (you'll ***contend*** for the same resource), which will cause a little bit of slowdown. And from time to time you'll have to work together (***communicate***) at the interface between his half and yours. The speedup will be nearly 2-to-1: y'all might take 35 minutes instead of 30.

The More the Merrier?



Now let's put Paul and Charlie on the other two sides of the table. Each of you can work on a part of the puzzle, but there'll be a lot more contention for the shared resource (the pile of puzzle pieces) and a lot more communication at the interfaces. So y'all will get noticeably less than a 4-to-1 speedup, but you'll still have an improvement, maybe something like 3-to-1: the four of you can get it done in 20 minutes instead of an hour.

Diminishing Returns



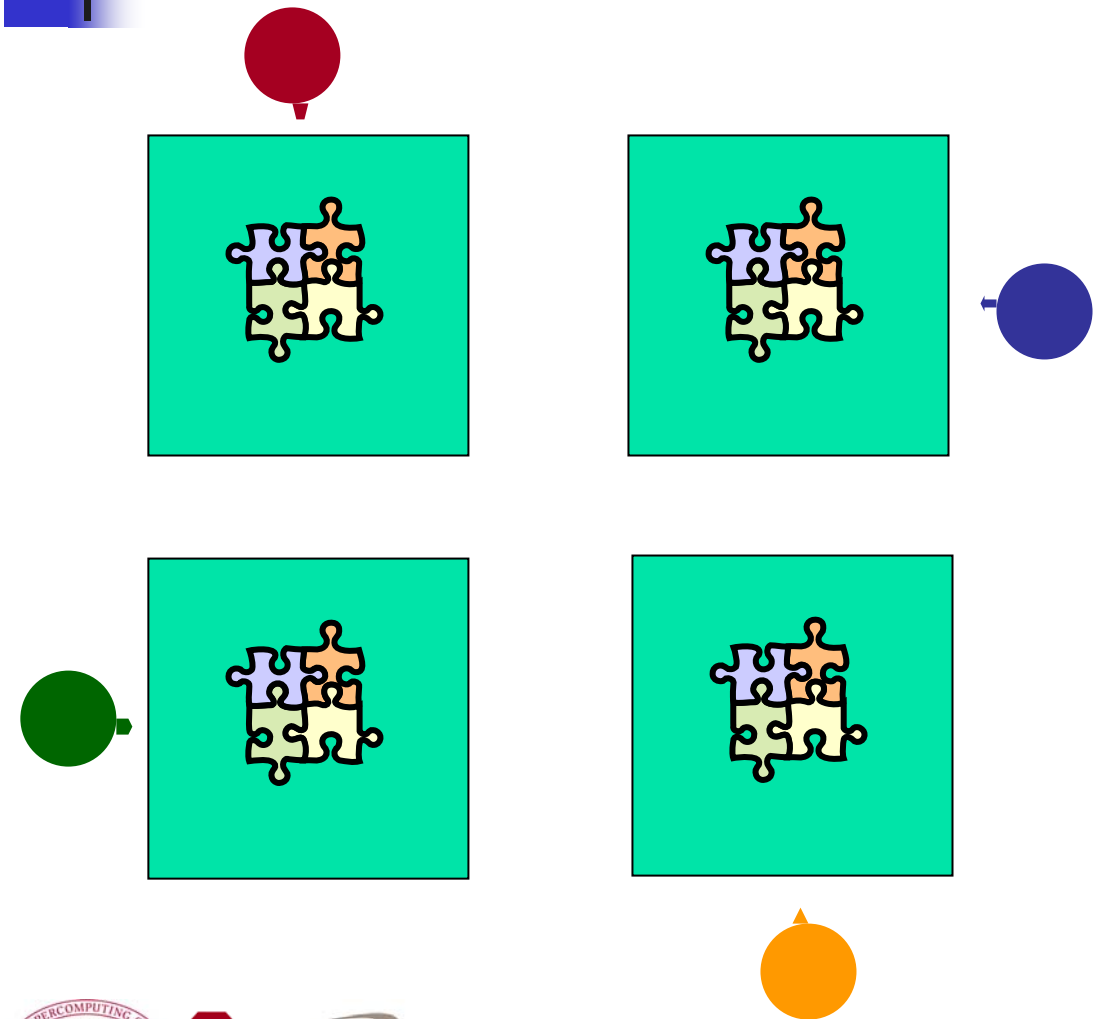
If we now put Dave and Tom and Horst and Brandon on the corners of the table, there's going to be a whole lot of contention for the shared resource, and a lot of communication at the many interfaces. So the speedup y'all get will be much less than we'd like; you'll be lucky to get 5-to-1.

So we can see that adding more and more workers onto a shared resource is eventually going to have a diminishing return.

Distributed Parallelism

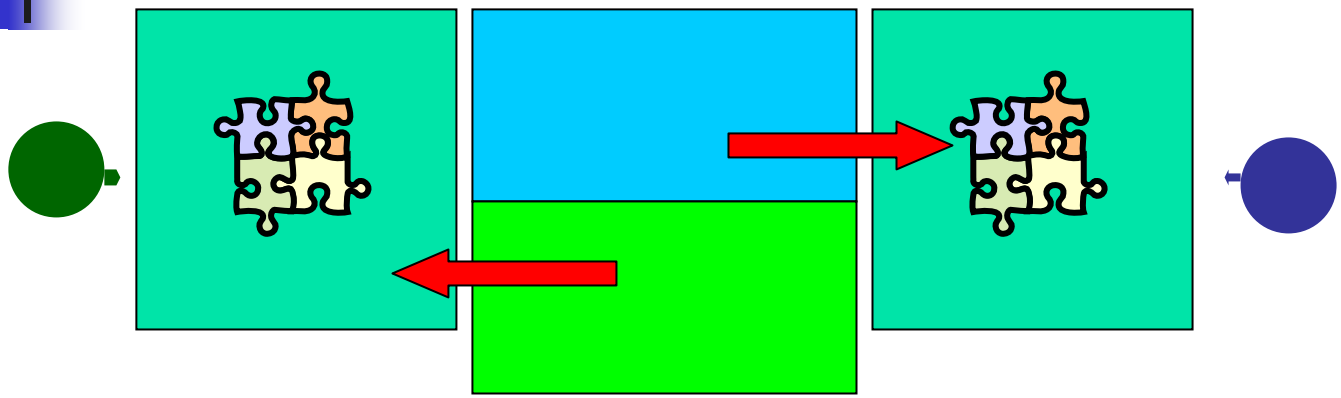
Now let's try something a little different. Let's set up two tables, and let's put you at one of them and Scott at the other. Let's put half of the puzzle pieces on your table and the other half of the pieces on Scott's. Now y'all can work completely independently, without any contention for a shared resource. **BUT**, the cost of communicating is **MUCH** higher (you have to scootch your tables together), and you need the ability to split up (***decompose***) the puzzle pieces reasonably evenly, which may be tricky to do for some puzzles.

More Distributed Processors



It's a lot easier to add more processors in distributed parallelism. But, you always have to be aware of the need to decompose the problem and to communicate between the processors. Also, as you add more processors, it may be harder to ***load balance*** the amount of work that each processor gets.

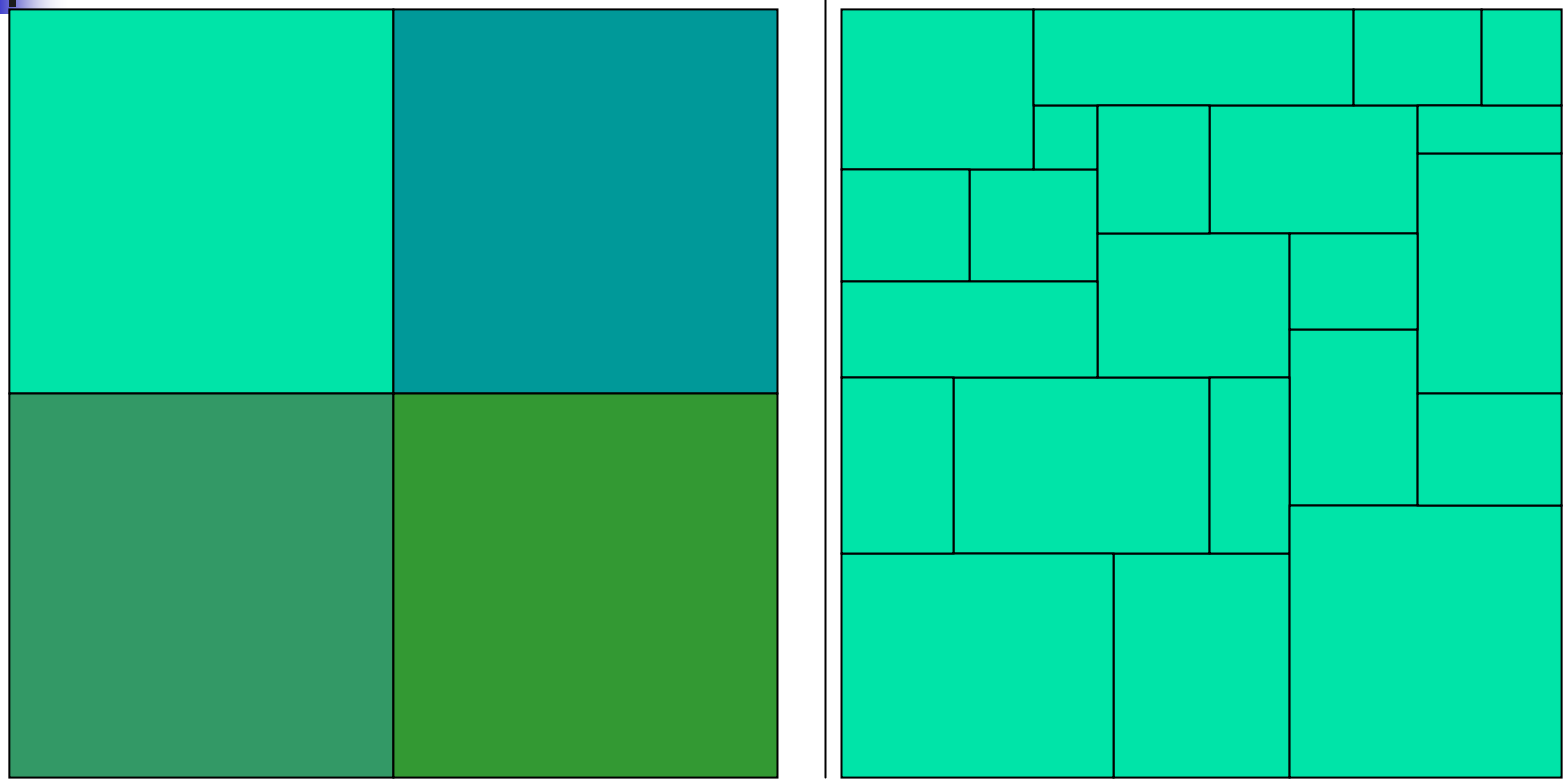
Load Balancing



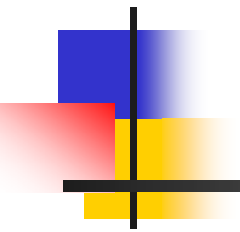
Load balancing means giving everyone roughly the same amount of work to do.

For example, if the jigsaw puzzle is half grass and half sky, then you can do the grass and Julie can do the sky, and then y'all only have to communicate at the horizon – and the amount of work that each of you does on your own is roughly equal. So you'll get pretty good speedup.

Load Balancing



Load balancing can be easy, if the problem splits up into chunks of roughly equal size, with one chunk per processor. Or load balancing can be very hard.



Moore’s Law

Moore's Law

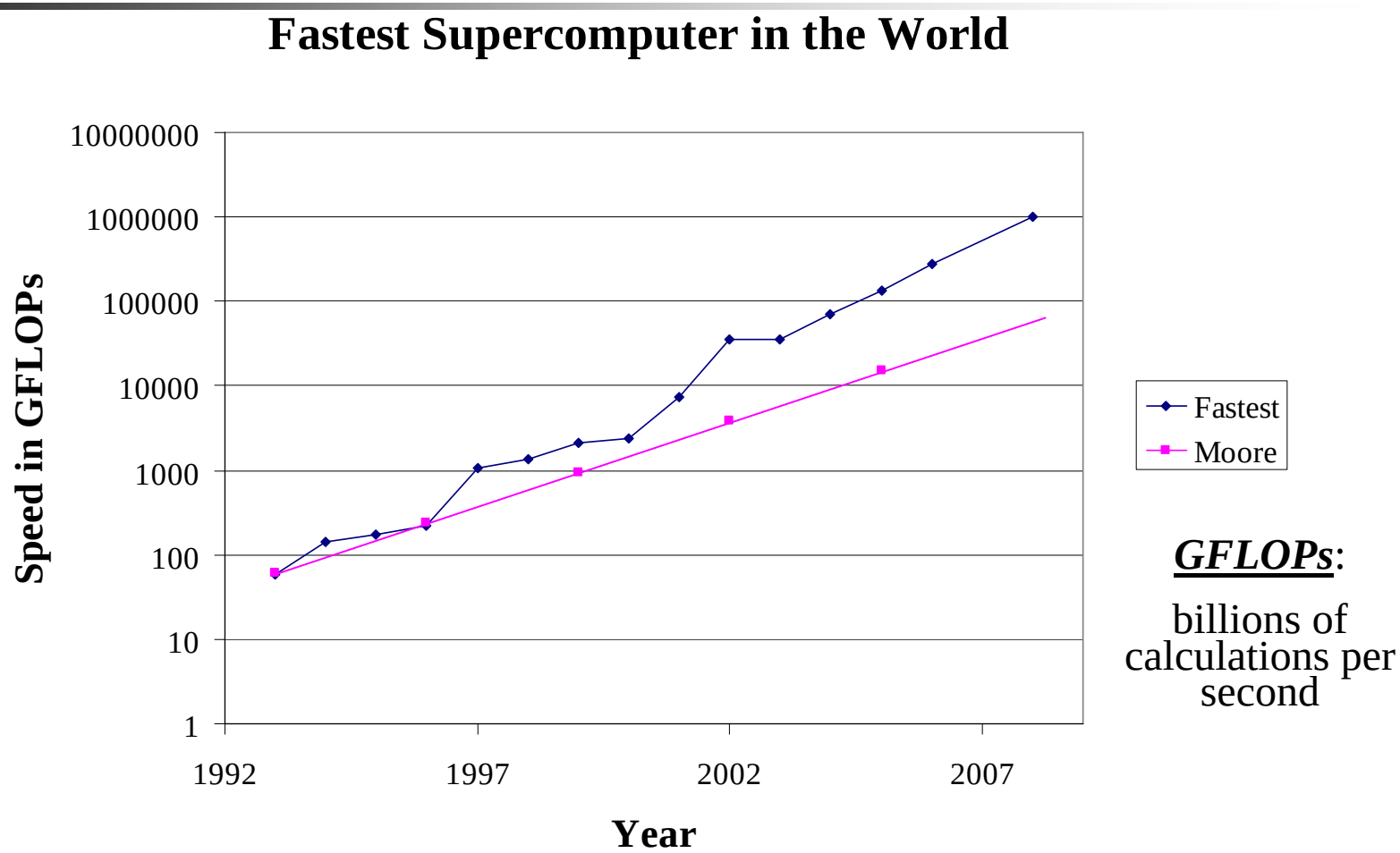
In 1965, Gordon Moore was an engineer at Fairchild Semiconductor.

He noticed that the number of transistors that could be squeezed onto a chip was doubling about every 18 months.

It turns out that computer speed is roughly proportional to the number of transistors per unit area.

Moore wrote a paper about this concept, which became known as ***"Moore's Law."***

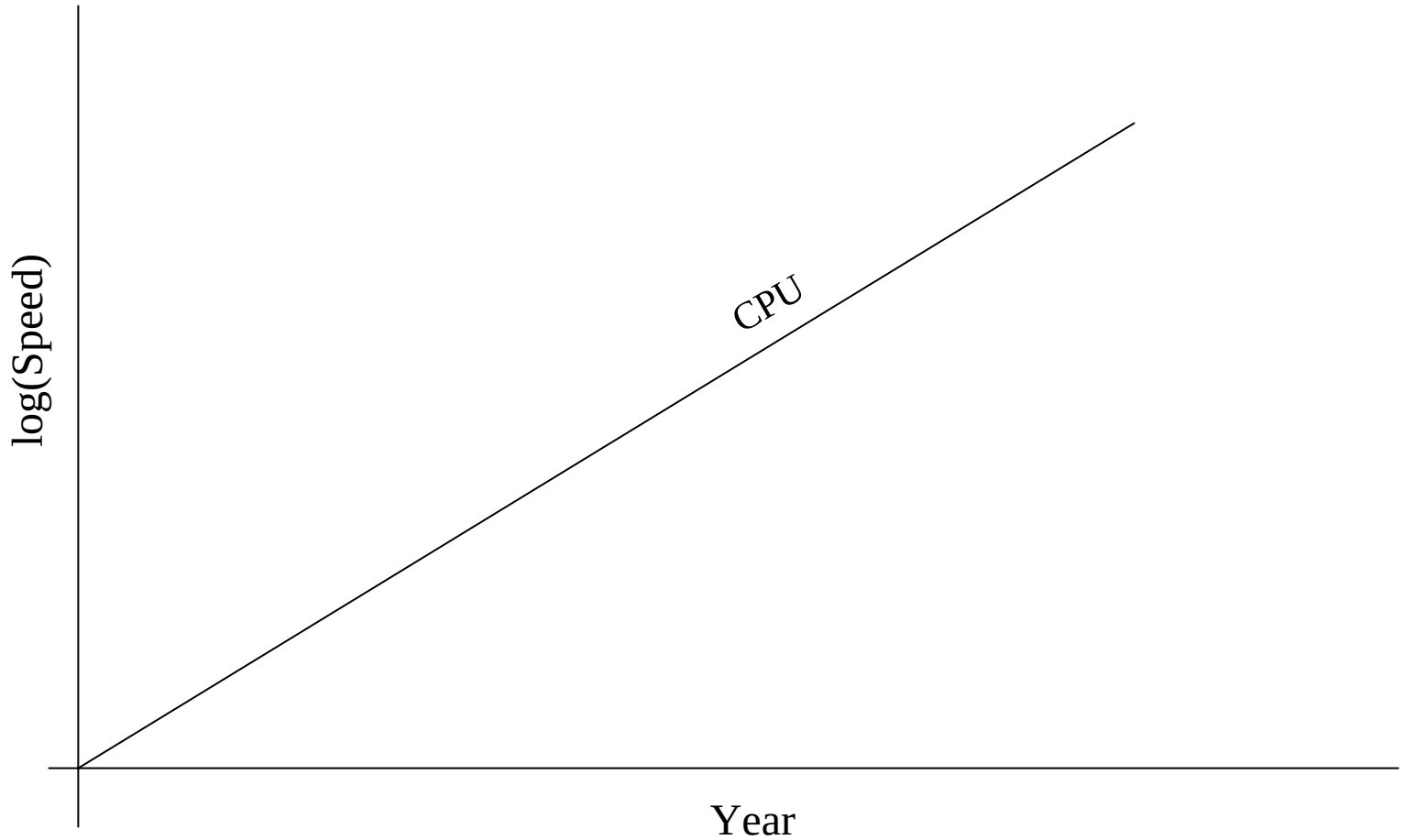
Fastest Supercomputer vs. Moore



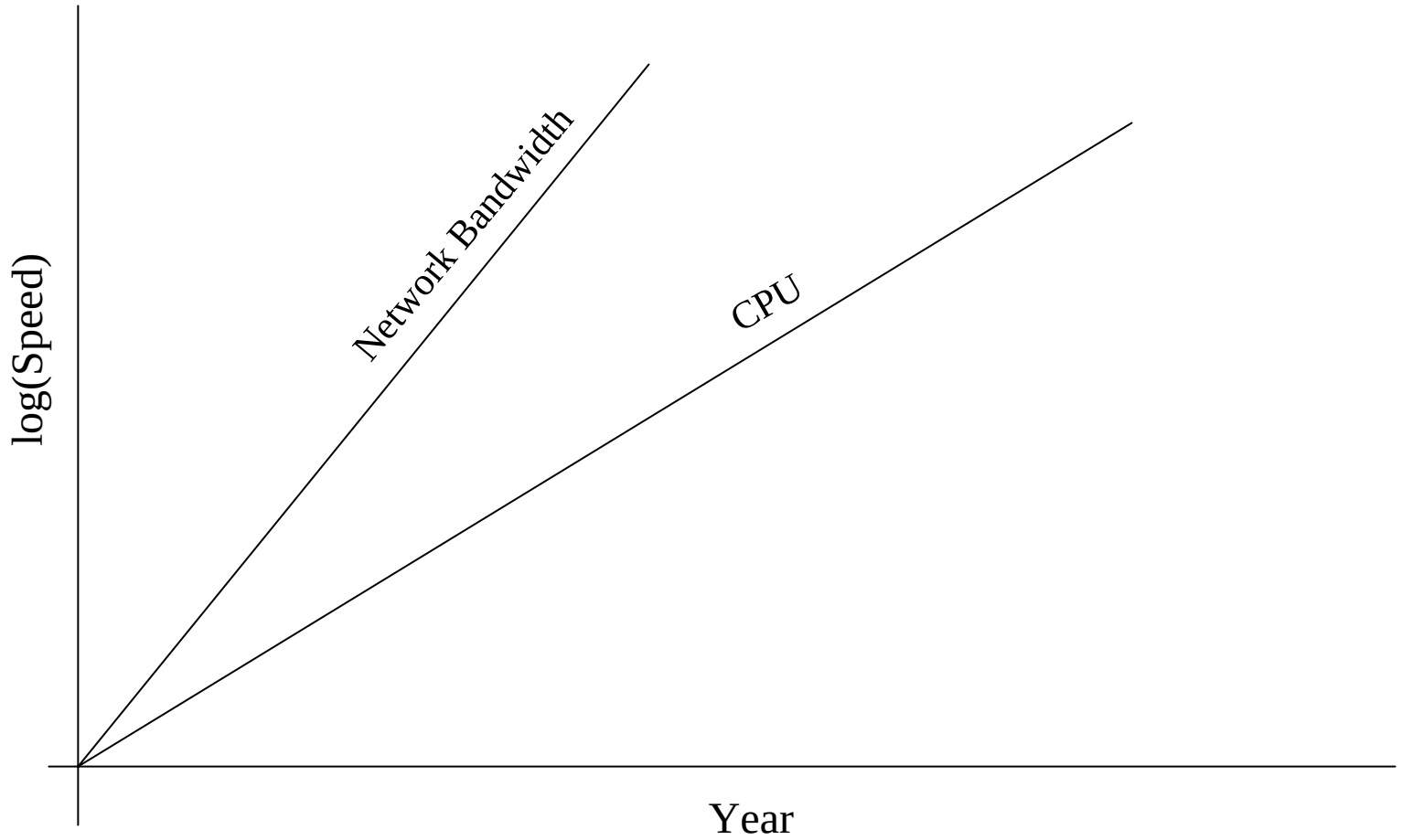
GFLOPs:

billions of calculations per second

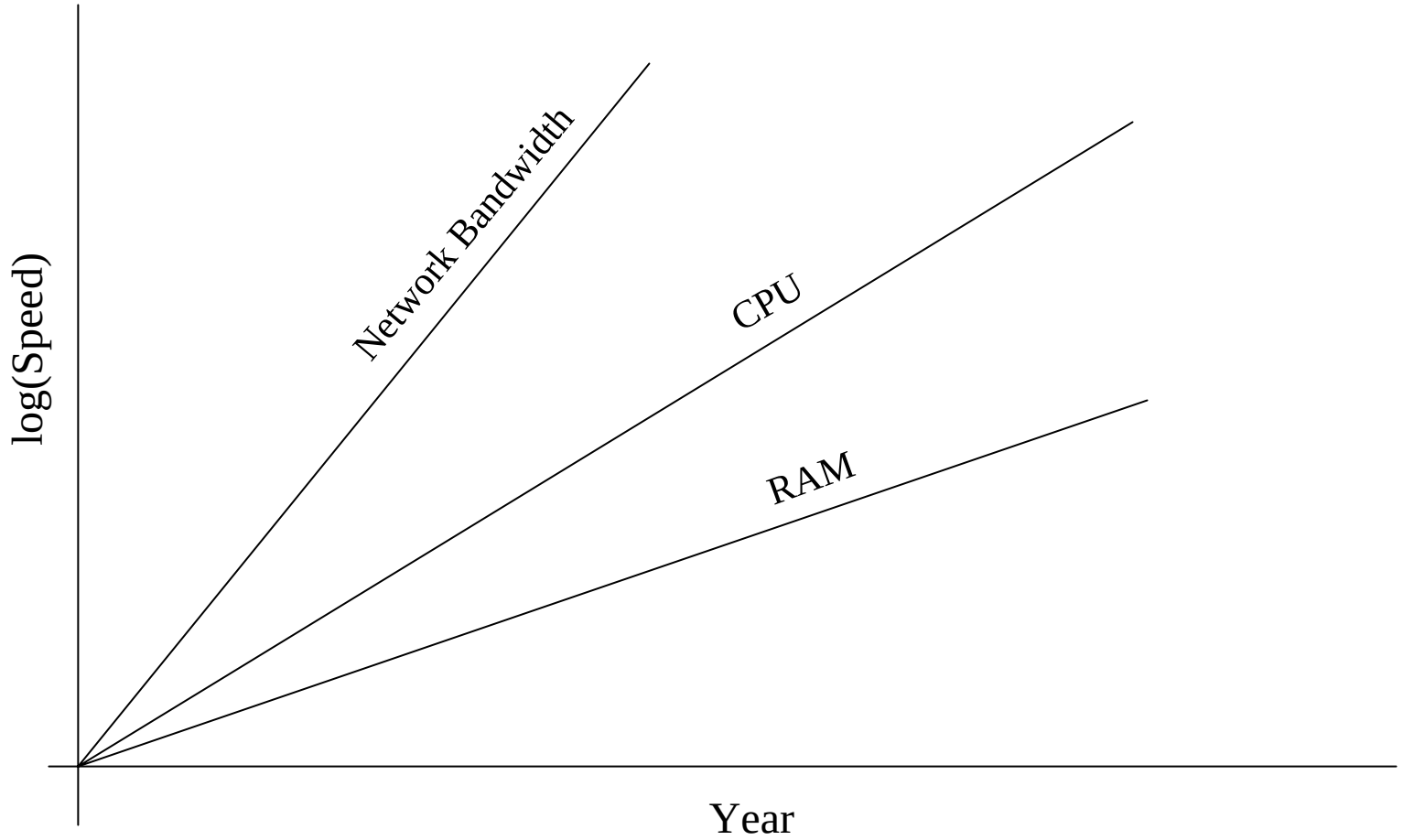
Moore's Law in Practice



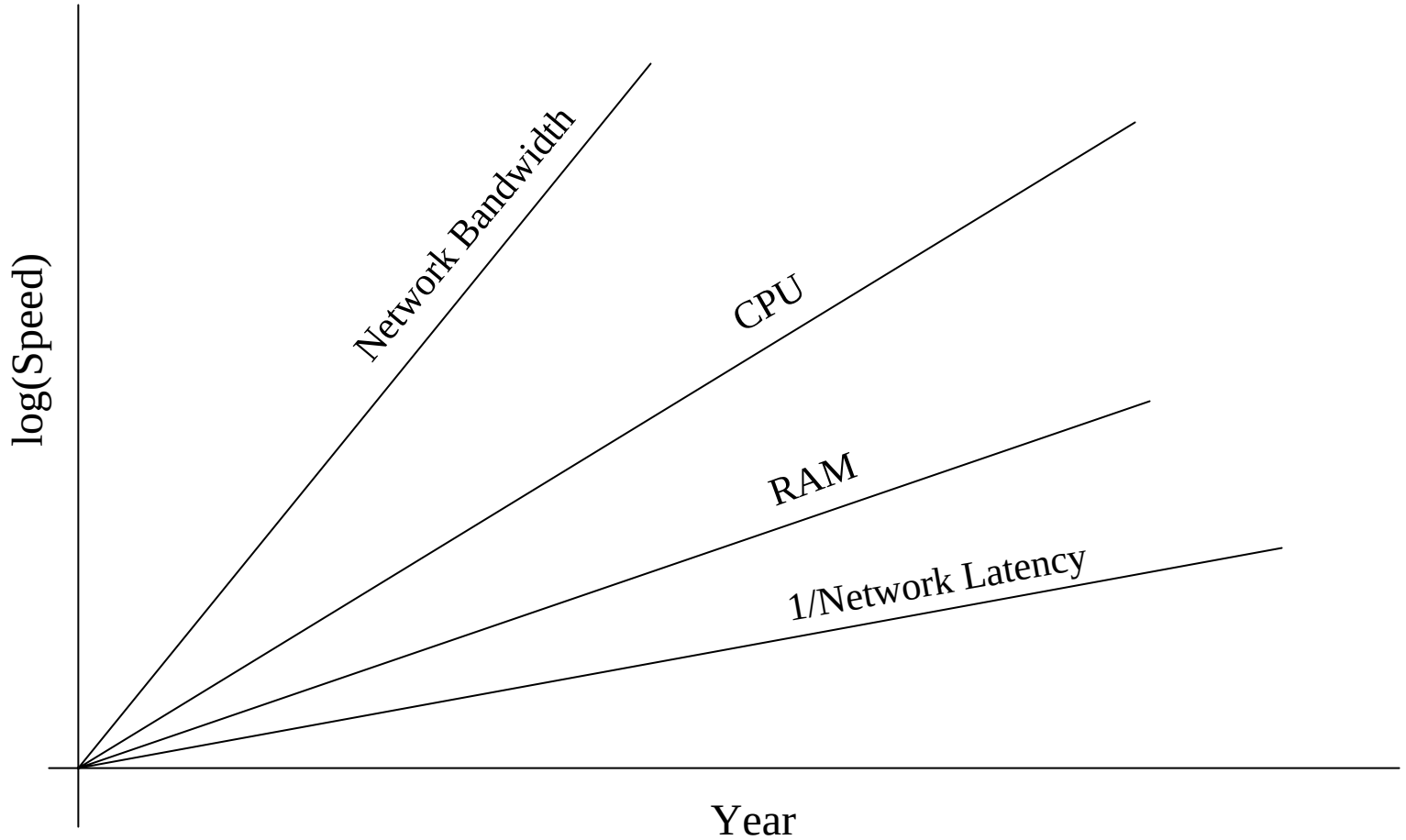
Moore's Law in Practice



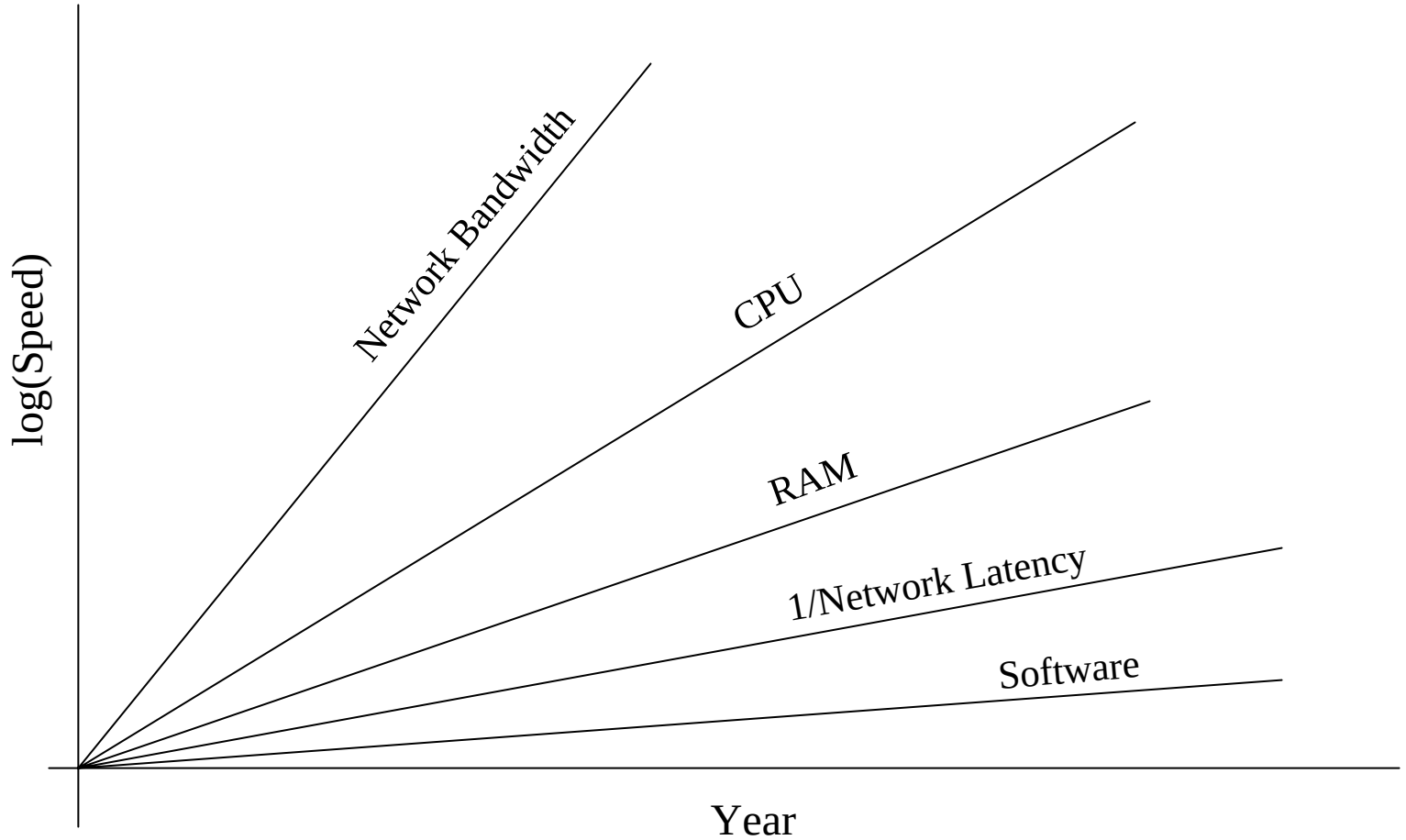
Moore's Law in Practice



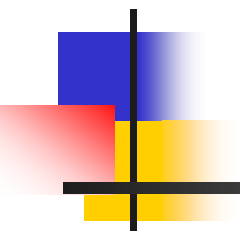
Moore's Law in Practice



Moore's Law in Practice



Why Bother?



Why Bother with HPC at All?

It's clear that making effective use of HPC takes quite a bit of effort, both learning how and developing software.

That seems like a lot of trouble to go to just to get your code to run faster.

It's nice to have a code that used to take a day run in an hour. But if you can afford to wait a day, what's the point of HPC?

Why go to all that trouble just to get your code to run faster?

Why HPC is Worth the Bother

- What HPC gives you that you won't get elsewhere is the ability to do **bigger, better, more exciting science**. If your code can run faster, that means that you can tackle much bigger problems in the same amount of time that you used to need for smaller problems.
- HPC is important not only for its own sake, but also because what happens in HPC today will be on your desktop in about 15 years: it puts you **ahead of the curve**.

The Future is Now

Historically, this has always been true:

Whatever happens in supercomputing today will be on your desktop in 10 – 15 years.

So, if you have experience with supercomputing, you'll be ahead of the curve when things get to the desktop.

OK Cyberinfrastructure Initiative

- Oklahoma is an EPSCoR state.
- Oklahoma recently submitted an NSF EPSCoR Research Infrastructure Proposal (up to $15M).
- This year, for the first time, all NSF EPSCoR RII proposals MUST include a statewide Cyberinfrastructure plan.
- Oklahoma's plan – the Oklahoma Cyberinfrastructure Initiative (OCII) – involves:
  - all academic institutions in the state are eligible to sign up for free use of OU's and OSU's centrally-owned CI resources;
  - other kinds of institutions (government, NGO, commercial) are eligible to use, though not necessarily for free.
- To join: see Henry after this talk.

To Learn More Supercomputing

http://www.oscer.ou.edu/education.php

Thanks for your attention!

Questions?

References

[1] Image by Greg Bryan, MIT: http://zeus.ncsa.uiuc.edu:8080/chdm_script.html

[2] "Update on the Collaborative Radar Acquisition Field Test (CRAFT): Planning for the Next Steps." Presented to NWS Headquarters August 30 2001.

[3] See http://scarecrow.caps.ou.edu/~hneeman/hamr.html for details.

[4] http://www.dell.com/

[5] http://www.f1photo.com/

[6] http://www.vw.com/newbeetle/

[7] Richard Gerber, The Software Optimization Cookbook: High-performance Recipes for the Intel Architecture. Intel Press, 2002, pp. 161-168.

[8] http://www.anandtech.com/showdoc.html?i=1460&p=2

[9] ftp://download.intel.com/design/Pentium4/papers/24943801.pdf

[10] http://www.seagate.com/cda/products/discsales/personal/family/0,1085,621,00.html

[11] http://www.samsung.com/Products/OpticalDiscDrive/SlimDrive/OpticalDiscDrive_SlimDrive_SN_S082D.asp?page=Specifications

[12] ftp://download.intel.com/design/Pentium4/manuals/24896606.pdf

[13] http://www.pricewatch.com/

[14] Steve Behling et al, *The POWER4 Processor Introduction and Tuning Guide*, IBM, 2001, p. 8.

[15] Kevin Dowd and Charles Severance, *High Performance Computing*, 2nd ed. O'Reilly, 1998, p. 16.

[16] http://emeagwali.biz/photos/stock/supercomputer/black-shirt/