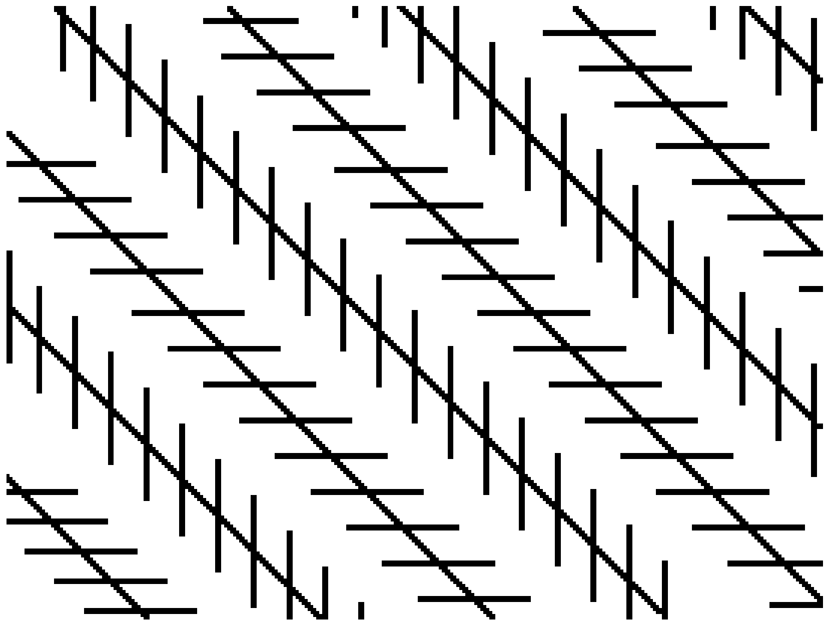
inst.eecs.berkeley.edu/~cs61c/su06

CS61C : Machine Structures

Lecture #28: Parallel Computing + Summary



2006-08-16

Andy Carle

What Programs Measure for Comparison?

- Ideally run typical programs with typical input before purchase, or before even build machine
  - Called a "workload"; For example:
  - Engineer uses compiler, spreadsheet
  - Author uses word processor, drawing program, compression software
- In some situations it's hard to do
  - Don't have access to machine to "benchmark" before purchase
  - Don't know workload in future

Example Standardized Benchmarks (1/2)

- Standard Performance Evaluation Corporation (SPEC) SPEC CPU2000
  - CINT2000 12 integer (gzip, gcc, crafty, perl, ...)
  - CFP2000 14 floating-point (swim, mesa, art, ...)
  - All relative to base machine Sun 300MHz 256Mb-RAM Ultra5_10, which gets score of 100
  - `www.spec.org/osg/cpu2000/`
  - They measure
    - System speed (SPECint2000)
    - System throughput (SPECint_rate2000)

Example Standardized Benchmarks (2/2)

- SPEC
  - Benchmarks distributed in source code
  - Big Company representatives select workload
    - Sun, HP, IBM, etc.
  - Compiler, machine designers target benchmarks, so try to change every 3 years

Example PC Workload Benchmark

- PCs: Ziff-Davis Benchmark Suite
  - "Business Winstone is a system-level, application-based benchmark that measures a PC's overall performance when running today's top-selling Windows-based 32-bit applications… it doesn't mimic what these packages do; it runs real applications through a series of scripted activities and uses the time a PC takes to complete those activities to produce its performance scores.
  - Also tests for CDs, Content-creation, Audio, 3D graphics, battery life

`http://www.etestinglabs.com/benchmarks/`

Performance Evaluation

- Good products created when have:
  - Good benchmarks
  - Good ways to summarize performance
- Given sales is a function of performance relative to competition, should invest in improving product as reported by performance summary?
- If benchmarks/summary inadequate, then choose between improving product for real programs vs. improving product to get more sales; Sales almost always wins!

Performance Evaluation: The Demo

If we're talking about performance, let's discuss the ways shady salespeople have fooled consumers (so that you don't get taken!)

5. Never let the user touch it
4. Only run the demo through a script
3. Run it on a stock machine in which "no expense was spared"
2. Preprocess all available data
1. Play a movie

OUR NEW CHIP IS SLOWER THAN OUR COMPETITION'S PRODUCTS.
WE'LL CLAIM WE'RE THE FASTEST. IF ANY-ONE DOES BENCHMARK TESTS, WE'LL SAY THEY USED OLD DRIVERS.
WHENEVER I TALK TO YOU, I FEEL LIKE I SHOULD BE WEARING A WIRE.
SINCE WHEN IS MARKETING A CRIME?
www.dilbert.com scottadams@aol.com
© 2004 Scott Adams, Inc./Dist. by UFS, Inc.

Performance Summary

- Benchmarks
  - Attempt to predict performance
  - Updated every few years
  - Measure everything from simulation of desktop graphics programs to battery life
- Megahertz Myth
  - MHz ≠ performance, it's just one factor

Scientific Computing

- Traditional Science
  1) Produce theories and designs on "paper"
  2) Perform experiments or build systems
  - Has become difficult, expensive, slow, and dangerous for fields on the leading edge
- Computational Science
  - Use ultra-high performance computers to simulate the system we're interested in
- Acknowledgement
  - Many of the concepts and some of the content of this lecture were drawn from Prof. Jim Demmel's CS 267 lecture slides which can be found at http://www.cs.berkeley.edu/~demmel/cs267_Spr05/

Example Applications

- Science
  - Global climate modeling
  - Biology: genomics; protein folding; drug design
  - Astrophysical modeling
  - Computational Chemistry
  - Computational Material Sciences and Nanosciences
- Engineering
  - Semiconductor design
  - Earthquake and structural modeling
  - Computation fluid dynamics (airplane design)
  - Combustion (engine design)
  - Crash simulation
- Business
  - Financial and economic modeling
  - Transaction processing, web services and search engines
- Defense
  - Nuclear weapons -- test by simulations
  - Cryptography

Performance Requirements

- Terminology
  - Flop – Floating point operation
  - Flops/second – standard metric for expressing the computing power of a system
- Global Climate Modeling
  - Divide the world into a grid (e.g. 10 km spacing)
  - Solve fluid dynamics equations to determine what the air has done at that point every minute
    - Requires about 100 Flops per grid point per minute
  - This is an extremely simplified view of how the atmosphere works, to be maximally effective you need to simulate many additional systems on a much finer grid

Performance Requirements (2)

- Computational Requirements
  - To keep up with real time (i.e. simulate one minute per wall clock minute): 8 Gflops/sec
  - Weather Prediction (7 days in 24 hours): 56 Gflops/sec
  - Climate Prediction (50 years in 30 days): 4.8 Tflops/sec
  - Climate Prediction Experimentation (50 years in 12 hours): 288 Tflops/sec
- Perspective
  - Pentium 4 1.4GHz, 1GB RAM, 4x100MHz FSB
    - ~320 Mflops/sec, effective
    - Climate Prediction would take ~1233 years

Reference:http://www.tc.cornell.edu/~lifka/Papers/SC2001.pdf

What Can We Do?

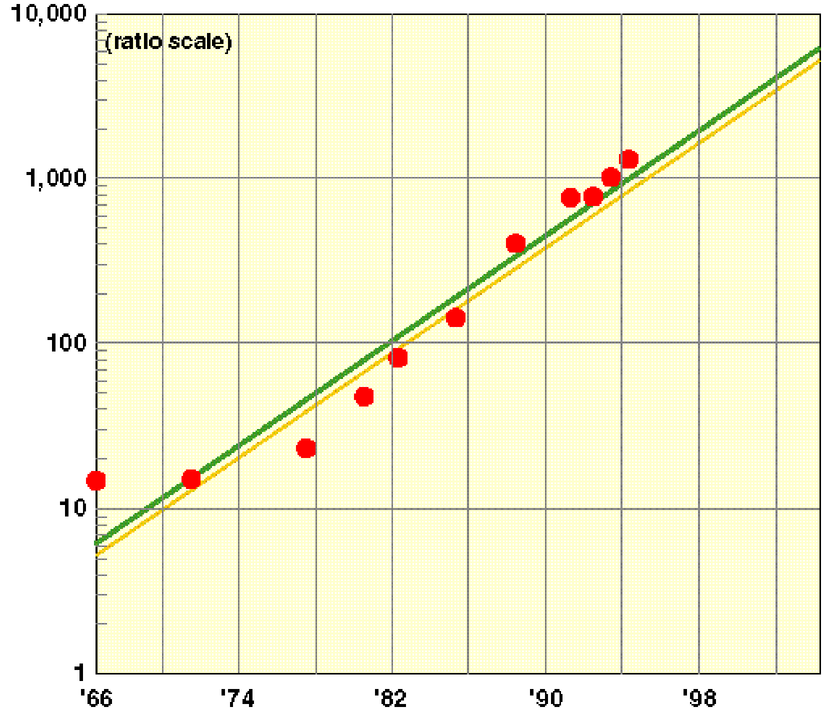
- Wait
  - Moore’s law tells us things are getting better; why not stall for the moment?
- Parallel Computing!

Prohibitive Costs

° **Rock's Law**

- The cost of building a semiconductor chip fabrication plant that is capable of producing chips in line with Moore's law doubles every four years

Cost of semiconductor factories in millions of 1995 dollars



Source: Forbes Magazine

How fast can a serial computer be?

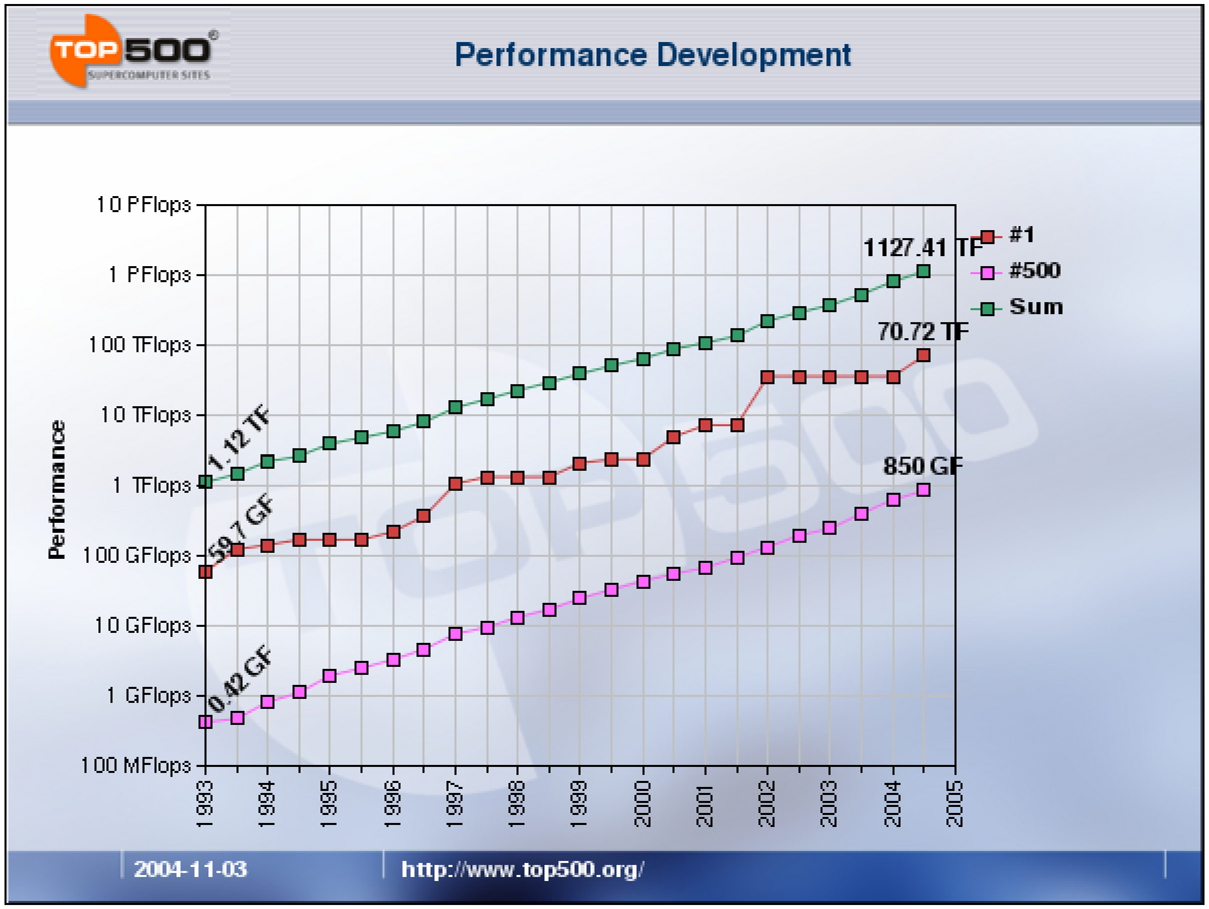
- Consider a 1 Tflop/sec sequential machine:
  - Data must travel some distance, r, to get from memory to CPU
  - To get 1 data element per cycle, this means 10^{12} times per second at the speed of light, $c = 3x10^8$ m/s. Thus $r < c/10^{12} = 0.3$ mm
    - So all of the data we want to process must be stored within 0.3 mm of the CPU
- Now put 1 Tbyte of storage in a 0.3 mm x 0.3 mm area:
  - Each word occupies about 3 square Angstroms, the size of a very small atom
  - Maybe someday, but it most certainly isn’t going to involve transistors as we know them

What is Parallel Computing?

- Dividing a task among multiple processors to arrive at a unified (meaningful) solution
  - For today, we will focus on systems with many processors executing identical code
- How is this different from Multiprogramming (which we've touched on some in this course)?
- How is this different from Distributed Computing?

Recent History

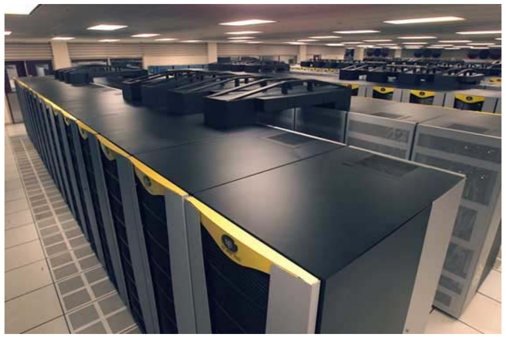
- Parallel Computing as a field exploded in popularity in the mid-1990s
- This resulted in an “arms race” between universities, research labs, and governments to have the fastest supercomputer in the world



Source: top500.org

Current Champions

BlueGene/L – IBM/DOE
Rochester, United States
32768 Processors, 70.72 Tflops/sec
0.7 GHz PowerPC 440



Columbia – NASA/Ames
Mountain View, United States
10160 Processors, 51.87 Tflops/sec
1.5 GHz SGI Altix

Earth Simulator – Earth Simulator Ctr.
Yokohama, Japan
5120 Processors, 35.86 Tflops/sec
SX6 Vector

Data Source: top500.org

Administrivia

- Proj 4 Due Friday
- HW8 (Optional) Due Friday
- Final Exam on Friday
    - Yeah, sure, you can have 3 one-sided cheat sheets
        - But I really don’t think they’ll help you all that much
- Course Survey in lab today

Parallel Programming

- Processes and Synchronization
- Processor Layout
- Other Challenges
  - Locality
  - Finding parallelism
  - Parallel Overhead
  - Load Balance

Processes

- We need a mechanism to intelligently split the execution of a program
- Fork:

```
int main(…){
  int pid = fork();
  if (pid == 0) printf(“I am the child.”);
  if (pid != 0) printf(“I am the parent.”);
  return 0;
}
```

- What will this print?

Processes (2)

- We don't know! Two potential orderings:
  - I am the child.I am the parent.
  - I am the parent.I am the child.
  - This situation is a simple <u>race condition.</u> This type of problem can get far more complicated…
- Modern parallel compilers and runtime environments hide the details of actually calling fork() and moving the processes to individual processors, but the complexity of <u>synchronization</u> remains

Synchronization

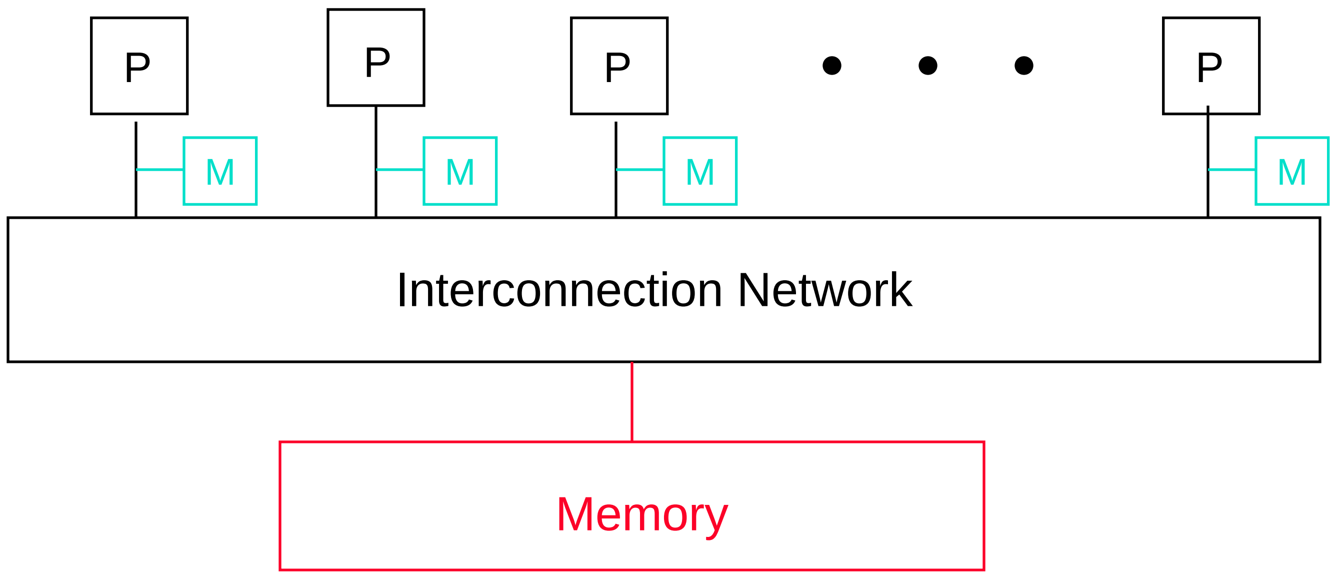
- How do processors communicate with each other?
- How do processors know when to communicate with each other?
- How do processors know which other processor has the information they need?
- When you are done computing, which processor, or processors, have the answer?

Synchronization (2)

- Some of the logistical complexity of these operations is reduced by standard communication frameworks
  - Message Passing Interface (MPI)
- Sorting out the issue of who holds what data can be made easier with the use of explicitly parallel languages
  - Unified Parallel C (UPC)
  - Titanium (Parallel Java Variant)
- Even with these tools, much of the skill and challenge of parallel programming is in resolving these problems

Processor Layout

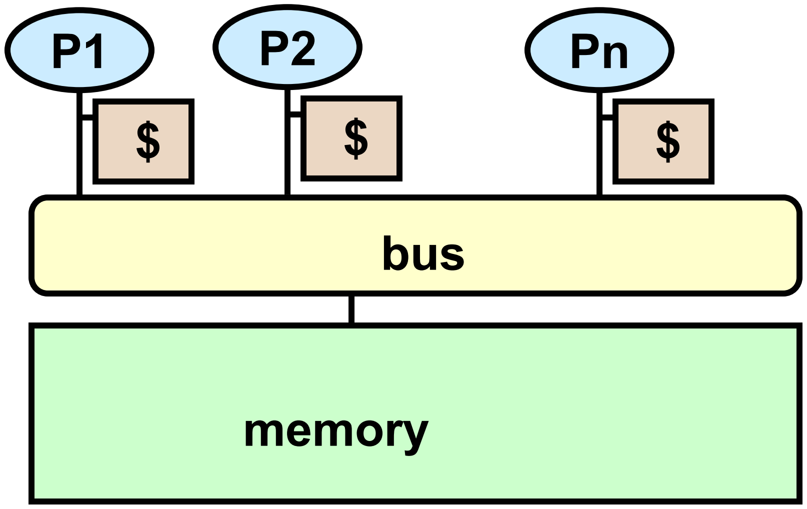
Generalized View



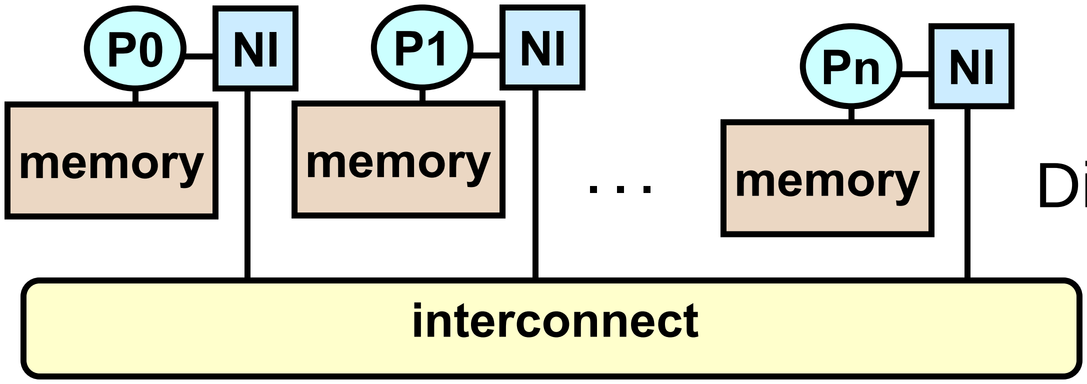
M = Memory local to one processor

Memory = Memory local to all *other* processors

Processor Layout (2)



Shared Memory



Distributed Memory

Processor Layout (3)

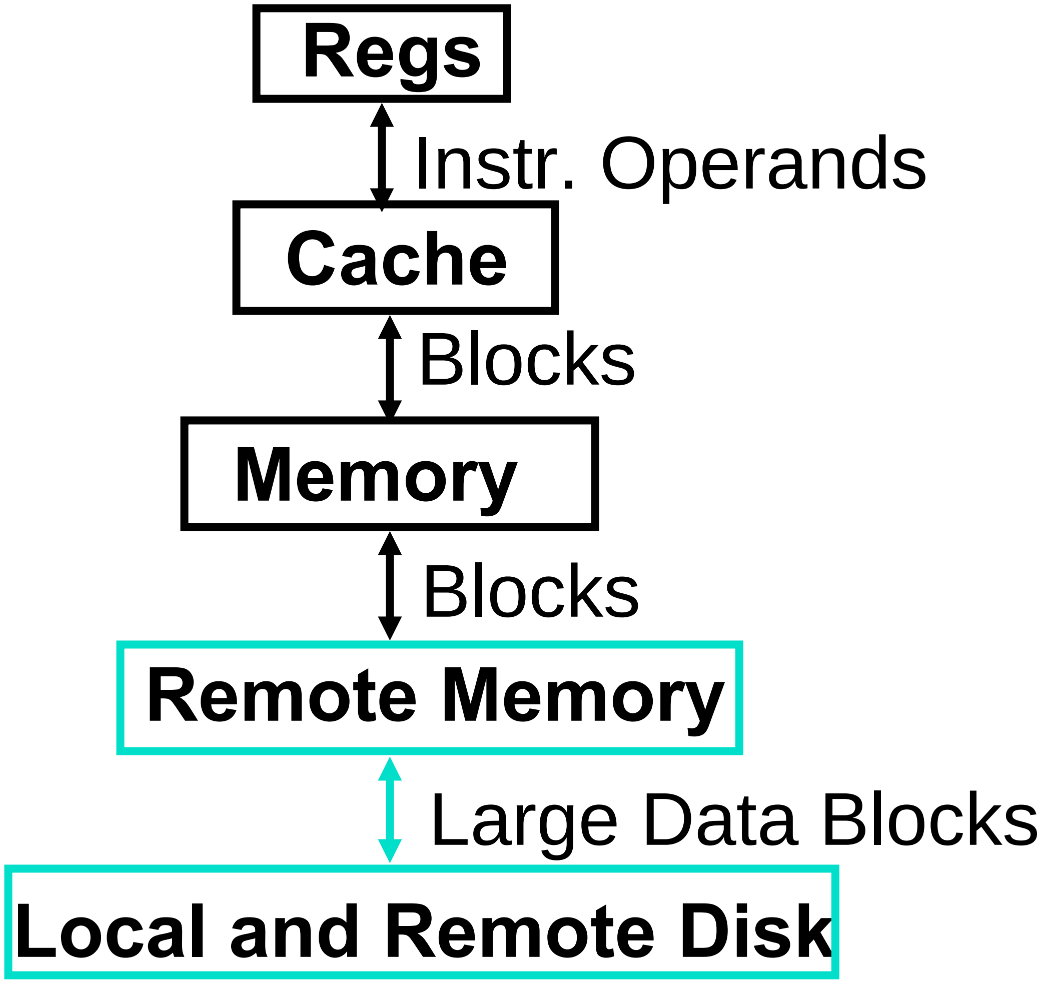
- Clusters of SMPs
  - n of the N total processors share one memory
  - Simple shared memory communication within one cluster of n processors
  - Explicit network-type calls to communicate from one group of n to another
- Understanding the processor layout that your application will be running on is crucial!

Administrivia

- There IS discussion today
  - Lab tomorrow will be an open office hour to review for the final
  - Review session tomorrow instead of lecture
  - Make sure to talk to your TAs and get your labs taken care of.
- If you did well in CS3 or 61{A,B,C} (A- or above) and want to be on staff?
  - Usual path: Lab assistant ⇒ Reader ⇒ TA
  - Fill in form outside 367 Soda before first week of semester…
  - We strongly encourage anyone who gets an A- or above in the class to follow this path…

Parallel Locality

- We now have to expand our view of the memory hierarchy to include remote machines
- Remote memory behaves like a very fast network
  - Bandwidth vs. Latency becomes important



Amdahl's Law

- Applications can almost never be completely parallelized
- Let s be the fraction of work done sequentially, so (1-s) is fraction parallelizable, and P = number of processors

Speedup(P) = Time(1)/Time(P)

<= 1/(s + (1-s)/P)

<= 1/s

- Even if the parallel portion of your application speeds up perfectly, your performance may be limited by the sequential portion

Parallel Overhead

- Given enough parallel work, these are the biggest barriers to getting desired speedup
- Parallelism overheads include:
  - cost of starting a thread or process
  - cost of communicating shared data
  - cost of synchronizing
  - extra (redundant) computation
- Each of these can be in the range of milliseconds (many millions of flops) on some systems
- Tradeoff: Algorithm needs sufficiently large units of work to run fast in parallel (I.e. large granularity), but not so large that there is not enough parallel work

Load Balance

- Load imbalance is the time that some processors in the system are idle due to
  - insufficient parallelism (during that phase)
  - unequal size tasks
- Examples of the latter
  - adapting to "interesting parts of a domain"
  - tree-structured computations
  - fundamentally unstructured problems
- Algorithms need to carefully balance load

Summary

- Parallel Computing is a multi-billion dollar industry driven by interesting and useful scientific computing applications
- It is extremely unlikely that sequential computing will ever again catch up with the processing power of parallel systems
- Programming parallel systems can be extremely challenging, but is built upon many of the concepts you've learned this semester in 61c

CS61C: So what's in it for me? (1st lecture)

Learn some of the big ideas in CS & engineering:

- 5 Classic components of a Computer
- Principle of abstraction, systems built as layers
- Data can be anything (integers, floating point, characters): a program determines what it is
- Stored program concept: instructions just data
- Compilation v. interpretation thru system layers
- Principle of Locality, exploited via a memory hierarchy (cache)
- Greater performance by exploiting parallelism (pipelining)
- Principles/Pitfalls of Performance Measurement

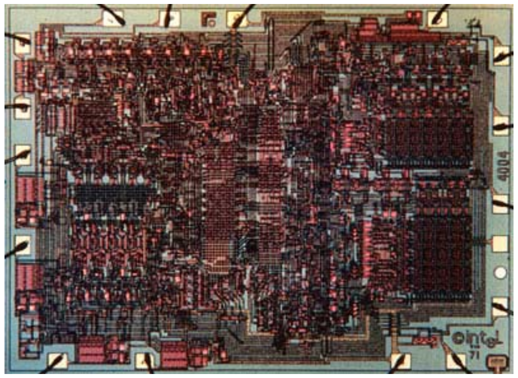
Thanks to Dave Patterson for these

Conventional Wisdom (CW) in Comp Arch

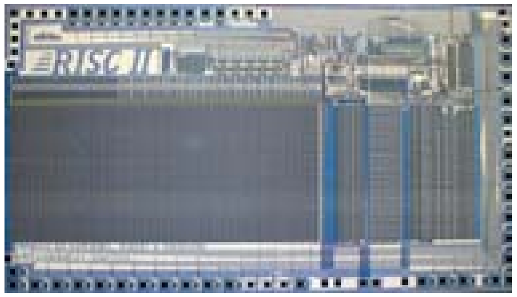
- Old CW: Power free, Transistors expensive
- New CW: Power expensive, Transistors free
  - Can put more on chip than can afford to turn on
- Old CW: Chips reliable internally, errors at pins
- New CW: ≤ 65 nm $\Rightarrow$ high error rates
- Old CW: CPU manufacturers minds closed
- New CW: Power wall + Memory gap = Brick wall
  - New idea receptive environment
- Old CW: Uniprocessor performance 2X / 1.5 yrs
- New CW: 2X CPUs per socket / ~ 2 to 3 years
  - Additional simple processors, more power efficient

Massively Parallel Socket

- Processor = new transistor?
  - Does it only help power/cost/performance?
- Intel 4004 (1971): 4-bit processor, 2312 transistors, 0.4 MHz, 10 µm PMOS, 11 mm^2 chip



- RISC II (1983): 32-bit, 5 stage pipeline, 40,760 transistors, 3 MHz, 3 µm NMOS, 60 mm^2 chip
  - 4004 shrinks to ~ 1 mm^2 at 3 micron



- 125 mm^2 chip, 65 nm CMOS = 2312 RISC IIs + Icache + Dcache
  - RISC II shrinks to ~ 0.02 mm^2 at 65 nm
  - Caches via DRAM or 1 transistor SRAM (www.t-ram.com)?
  - Proximity Communication at > 1 TB/s ?
  - Ivan Sutherland @ Sun spending time in Berkeley!

20th vs. 21st Century IT Targets

- 20th Century Measure of Success
  - Performance (peak vs. delivered)
  - Cost (purchase cost vs. ownership cost, power)
- 21st Century Measure of Success? "SPUR"
  - Security
  - Privacy
  - Usability
  - Reliability
- Massive parallelism greater chance (this time) if
  - Measure of success is SPUR vs. only cost-perf
  - Uniprocessor performance improvement decelerates

Other Implications

- Need to revisit chronic unsolved problem
  - Parallel programming!!
- Implications for applications:
  - Computing power >>> CDC6600, Cray XMP (choose your favorite) on an economical die inside your watch, cell phone or PDA
    - On your body health monitoring
    - Google + library of congress on your PDA
- As devices continue to shrink…
  - The need for great HCI critical as ever!

Taking advantage of Cal Opportunities

Why are we a top university?

- Research, reseach, research!
- Whether you want to go to grad school or industry, you need someone to vouch for you! (as is the case with the Mob)

° Techniques

- Find out what you like, do lots of web research (read published papers), hit OH of Prof, show enthusiasm & initiative (and get to know grad students!)

° `http://research.berkeley.edu/`

UC-WISE and Curriculum Development

- Actively seeking undergrad volunteers to work on the new UCWISE interface
  - CS 199 Credit
- Developing UC-WISE based version of CS61c
  - Those of you who just took the course and are interested in curriculum design are the perfect people to help
- Andy needs undergrads!
  - Work on interesting user interface design issues related to curriculum development
- Contact Andy if you are interested in any of these opportunities!

Penultimate slide: Thanks to the staff!

- TAs
  - Chris
  - Michael
- Reader
  - Albert

Thanks to Dave Patterson, John Wawrzynek, Dan Garcia, Mike Clancy, Kurt Meinz, and everyone else that has worked on these lecture notes over the years.

The Future for Future Cal Alumni

- What's The Future?
- New Millennium
  - Internet, Wireless, Nanotechnology, ...
  - Rapid Changes in Technology
  - World's Best Education
  - Work Hard!

"The best way to predict the future is to invent it" – Alan Kay

The Future is up to you!