Multi-core System Programming

Sulaiman.S.M.
smsulaiman@vit.ac.in

What is Parallel Computing?

- **Parallel computing:** using multiple processors in parallel to solve problems more quickly than with a single processor
- Examples of parallel machines:
  - A **cluster computer** that contains multiple PCs combined together with a high speed network
  - A **shared memory multiprocessor** (SMP*) by connecting multiple processors to a single memory system
  - A **Chip Multi-Processor** (CMP) contains multiple processors (called cores) on a single chip
- Concurrent execution comes from desire for performance
- * Technically, SMP stands for “Symmetric Multi-Processor”

Questions

- Why we need ever-increasing Performance?
- Why build parallel systems?
- Why we need to write parallel programs?

Why we need ever-increasing Performance?

Weather forecast

- time critical
- tomorrow's news today

Computer Simulation (many application)

- Games
- 3D rendering, Animations
- Image Processing

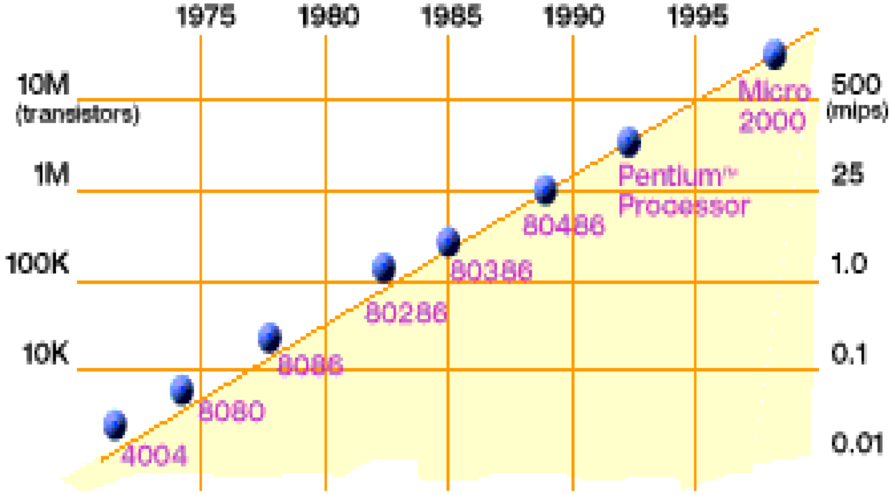
Data mining

- DNA analysis

- Researchers have been using parallel computing for decades:
  - Mostly used in computational science and engineering
  - Problems too large to solve on one computer; use 100s or 1000s
- There is a desperate need for parallel programmers
- Let’s see why…

Why we need to build parallel systems?

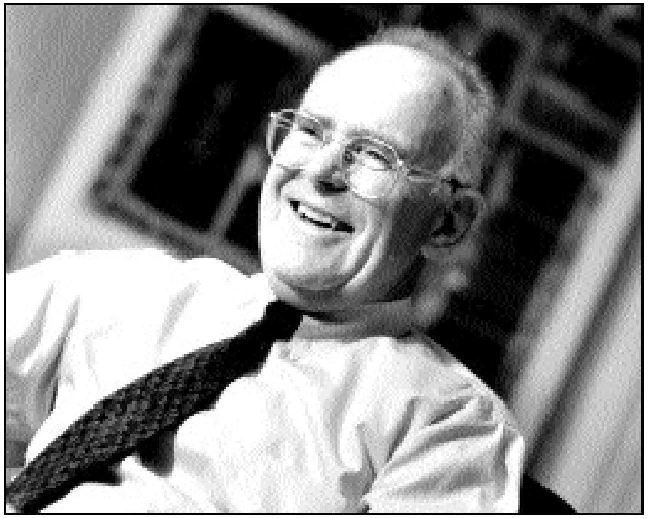
Technology Trends: Microprocessor Capacity



2X transistors/Chip Every 1.5 years

Called "[Moore's Law]"

Microprocessors have become smaller, denser, and more powerful.



Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

Slide source: Jack Dongarra

Microprocessor Transistors and Clock Rate

1986 -2002

- ✓Growth in transistors per chip and Increase in clock rate

Since 2002

- ✓Slow down to 20% per year

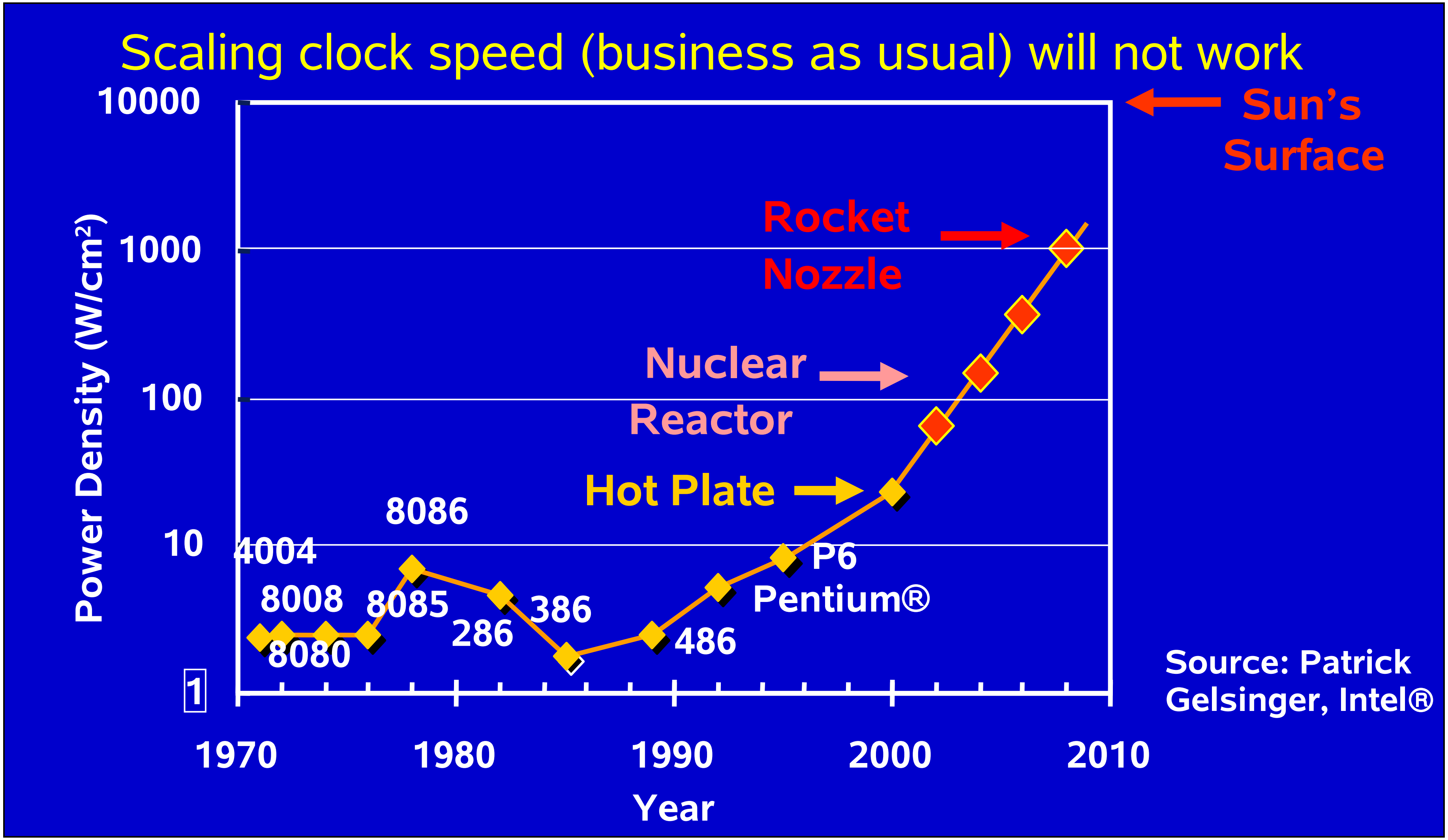
Difference:

- ✓With 50%/yr gain is about 60
- ✓With 20%/yr gain is about 6 only

Fundamentals:

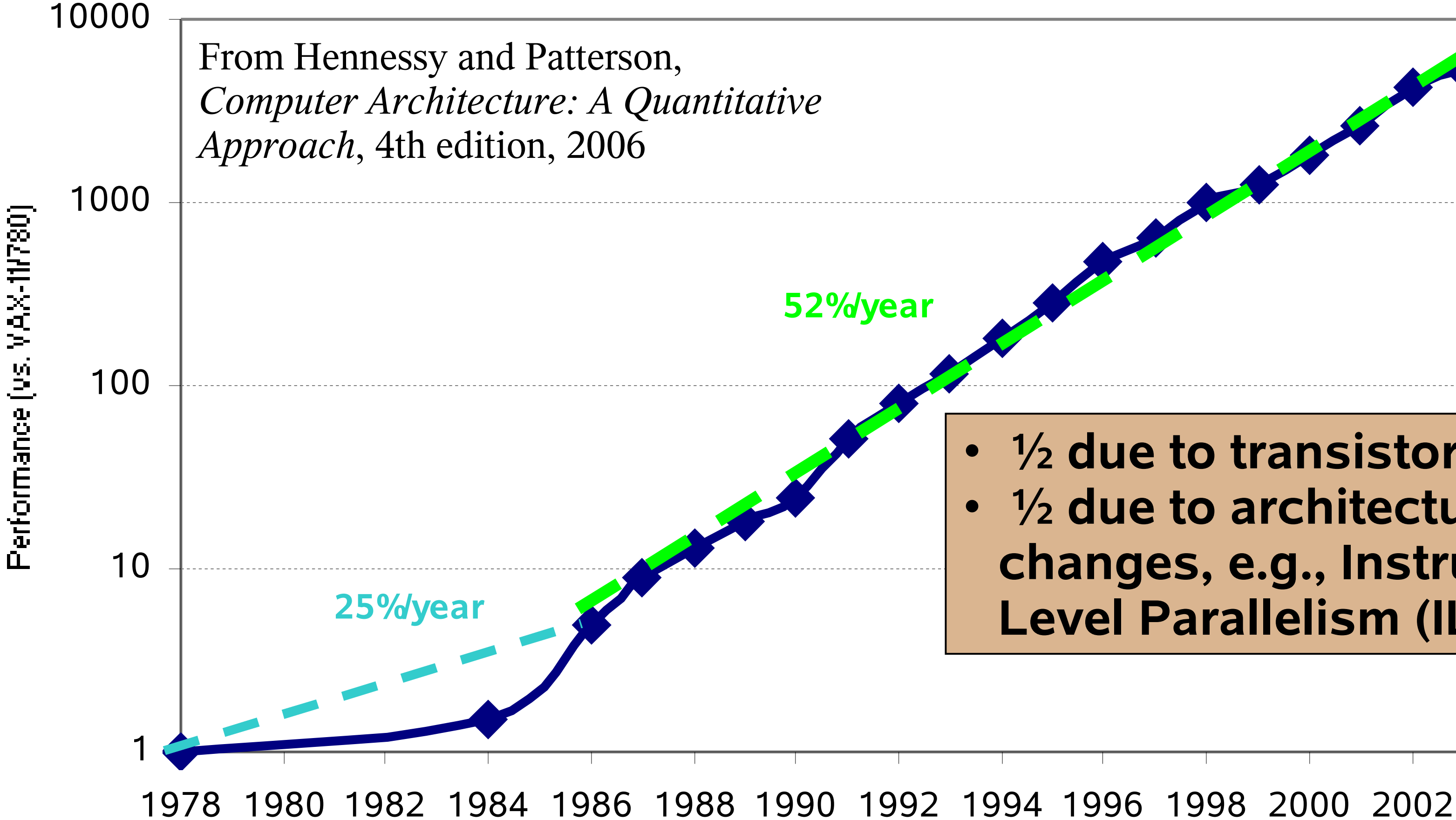
- ✓As we decrease the transistor size, its speed increases
- ✓As speed increases the power consumption increases
- ✓This causes high heat energy

Speed -> Power -> Heat



Hidden Parallelism

Application performance was increasing by 52% per year as measured by the SpecInt benchmarks here



- **½ due to transistor density**
- **½ due to architecture changes, e.g., Instruction Level Parallelism (ILP)**

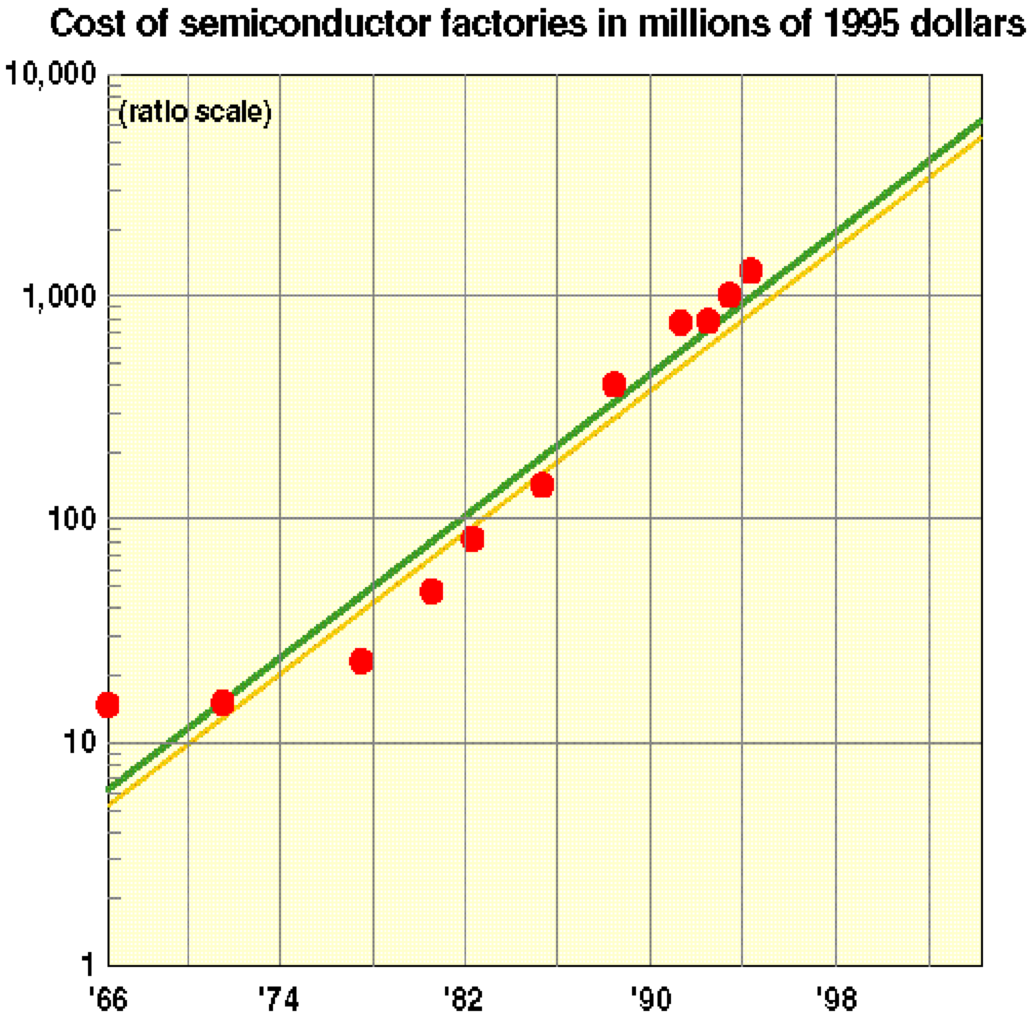
- **VAX : 25%/year 1978 to 1986**
- **RISC + x86: 52%/year 1986 to 2002**

Hidden Parallelism

- Pipelining
- Super-pipeline
- Superscalar (SS)
- VLIW
- Many forms of parallelism not visible to programmer
  - multiple instruction issue
  - dynamic scheduling: hardware discovers parallelism between instructions
  - speculative execution: look past predicted branches
  - non-blocking caches: multiple outstanding memory ops
- Unfortunately, these sources have been used up
- Improvement is limited
  - Waiting for memory (D and I-cache stalls)
  - Dependencies (pipeline stalls)

Chip Yield

Manufacturing costs and yield problems limit use of density

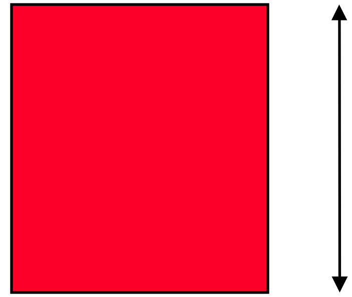


- **Moore's (Rock's) 2nd law: fabrication costs go up**
- **Yield (% usable chips) drops**

- **Parallelism can help**
  - **More smaller, simpler processors are easier to design and validate**
  - **Can use partially working chips:**
  - **E.g., Cell processor (PS3) is sold with 7 out of 8 “on” to improve yield**

Speed of Light (Fundamental)

1 Tflop/s, 1 Tbyte sequential machine

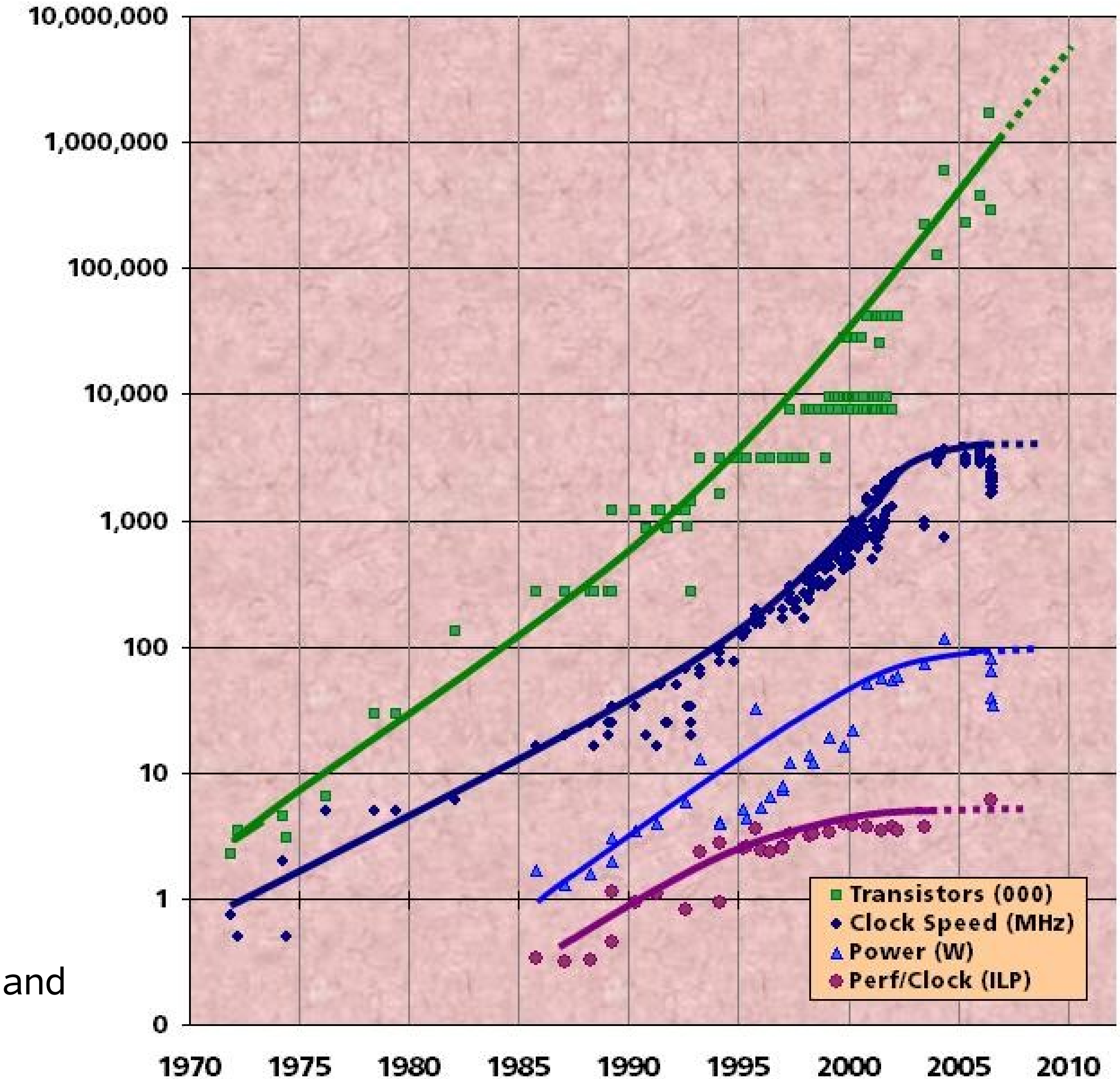


r = 0.3 mm

- Consider the 1 Tflop/s 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, $c = 3\text{x}10^8$ m/s. Thus $r < c/10^{12} = 0.3$ mm.
- Now put 1 Tbyte of storage in a 0.3 mm x 0.3 mm area:
  - Each bit occupies about 1 square Angstrom, or the size of a small atom.
- No choice but parallelism

Revolution is Happening Now

- Chip density is continuing increase ~2x every 2 years
  - Clock speed is not
  - Number of processor cores may double instead
- There is little or no hidden parallelism (ILP) to be found
- Parallelism must be exposed to and managed by software



Source: Intel, Microsoft (Sutter) and Stanford (Olukotun, Hammond)

Multicore in Products

- “We are dedicating all of our future product development to multicore designs. … This is a sea change in computing”

  Paul Otellini, President, Intel (2005)

- All microprocessor companies switch to MP (2X CPUs / 2 yrs)

Tunnel Vision by Experts

- "On several recent occasions, I have been asked whether parallel computing will soon be relegated to the trash heap reserved for promising technologies that never quite make it."
    - Ken Kennedy, CRPC Directory, 1994
- "640K [of memory] ought to be enough for anybody."
    - Bill Gates, chairman of Microsoft,1981.
- "There is no reason for any individual to have a computer in their home"
    - Ken Olson, president and founder of Digital Equipment Corporation, 1977.
- "I think there is a world market for maybe five computers."
    - Thomas Watson, chairman of IBM, 1943.

Why we need to write Parallel Programs?

- All major processor vendors are producing multicore chips
  - Every machine will soon be a parallel machine
  - All programmers will be parallel programmers???
- New software model
  - Want a new feature? Hide the “cost” by speeding up the code first
  - All programmers will be performance programmers???
- Some may eventually be hidden in libraries, compilers, and high level languages
  - But a lot of work is needed to get there
- Big open questions:
  - What will be the killer apps for multicore machines
  - How should the chips be designed, and how will they be programmed?

Why writing (fast) parallel programs is hard

Principles of Parallel Computing

- Finding enough parallelism (Amdahl's Law)
- Granularity
- Locality
- Load balance
- Coordination and synchronization
- Performance modeling

All of these things makes parallel programming even harder than sequential programming.

Finding Enough Parallelism

- Suppose only part of an application seems parallel
- Amdahl's law
  - let s be the fraction of work done sequentially, so (1-s) is fraction parallelizable
  - P = number of processors

$$\text{Speedup}(P) = \text{Time}(1)/\text{Time}(P)$$

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

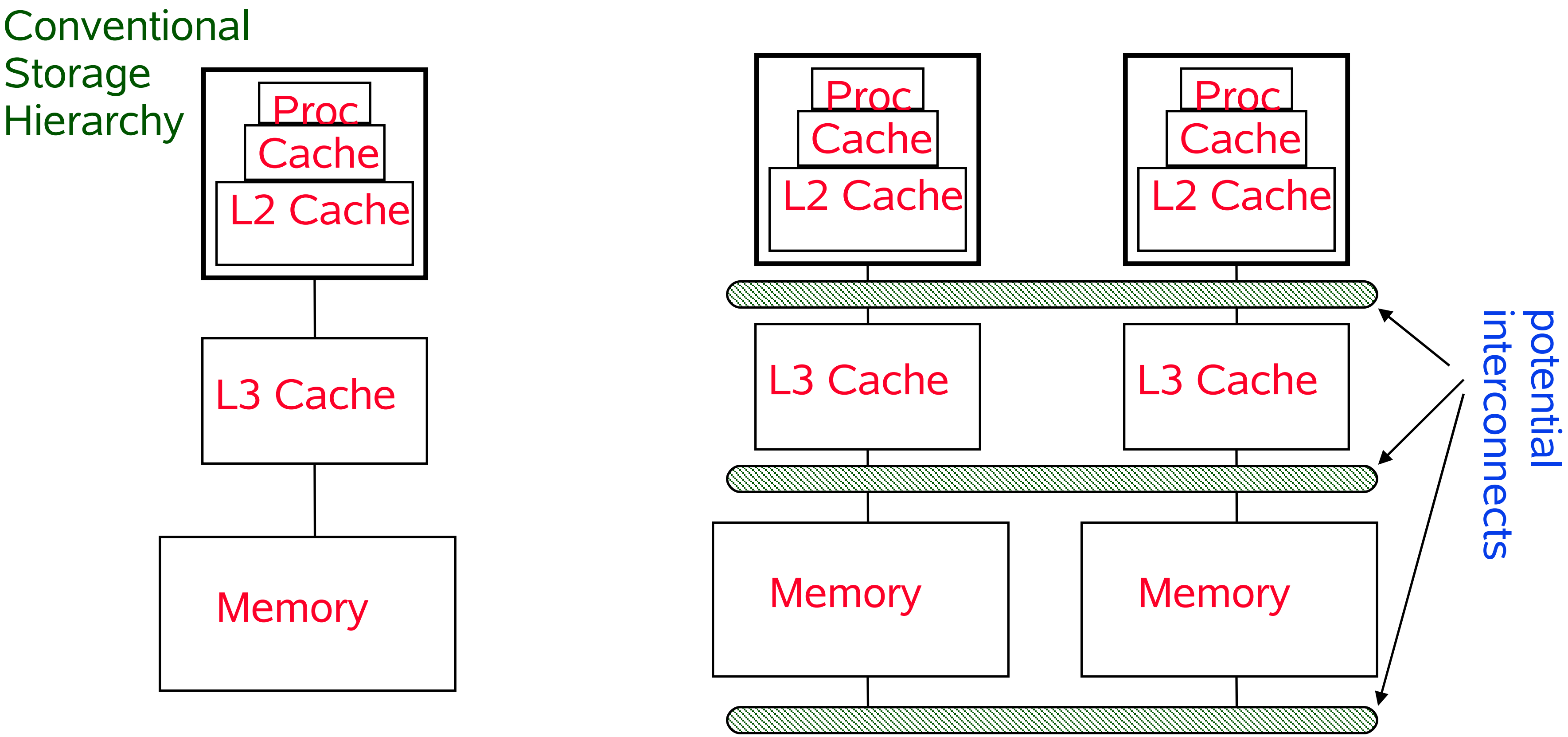
$$<= 1/s$$

- Even if the parallel part speeds up perfectly performance is limited by the sequential part

Overhead of Parallelism

- Given enough parallel work, this is the biggest barrier 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 (=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

Locality and Parallelism



- Large memories are slow, fast memories are small
- Storage hierarchies are large and fast <u>on average</u>
- Parallel processors, collectively, have large, fast cache
  - the slow accesses to "remote" data we call "communication"
- Algorithm should do most work on local data

Load Imbalance

- 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
- Algorithm needs to balance load

Course Organization

- Expected background
  - C Programming
  - Computer Architecture
  - Operating System concepts – Process and Threads
- Evaluation
  - CAT 1 & 2 (15 + 15)
  - Programming Assignments/Quizes (20)
- Caveat: This is the first offering of this course, so things will change dynamically