LAU
الجامعة اللبنانية الأميركية
Lebanese American University

CSC 447: Parallel Programming for Multi-Core and Cluster Systems

Introduction and Administrivia

Haidar M. Harmanani

Spring 2019

Course Introduction

- Lectures
  - TTh, 11:00-12:15 from January 15, 2019 until April 24, 2019
  - Prerequisites
    - Know how to program
    - Data Structures
    - Computer Architecture would be helpful but not required.

Grading and Class Policies

- Grading
  - Midterm (1-2): 30%
  - Final: 40%
  - Labs [6-8]: 20%
  - Two milestones projects (OpenMP and CUDA): 10%
- Exams Details
  - Exams are closed book, closed notes
- All assignments must be your own original work.
  - Cheating/copying/partnering will not be tolerated

Teaching Methodology

- We will use multiple choice questions to initiate discussions and reinforce learning
- We will also use a a flipped-classroom methodology for one part of the course

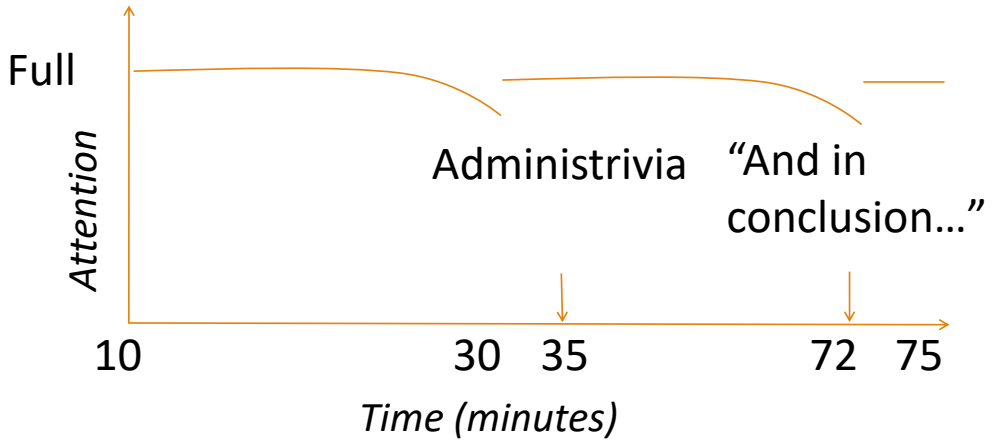
Lab Assignments

- All assignments and handouts will be communicated via `piazza`
  - Make sure you enable your account
- Use `piazza` for questions and inquiries
  - No questions will be answered via email
- All assignments must be submitted via `github`
  - `git` is a distributed version control system
  - `git` or its variations have become a universal standard for developing and sharing code
  - Make sure you get a private repo
    - Apply for a free account: https://education.github.com/discount_requests/new

Policy on Assignments and Independent Work

- With the exception of laboratories and assignments (projects and HW) that explicitly permit you to work in groups, all homework and projects are to be YOUR work and your work ALONE.
- It is NOT acceptable to copy solutions from other students.
- It is NOT acceptable to copy (or start your) solutions from the Web.
- PARTNER TEAMS MAY NOT WORK WITH OTHER PARTNER TEAMS
- You are encouraged to help teach other to debug. Beyond that, we don't want you sharing approaches or ideas or code or whiteboarding with other students, since sometimes the point of the assignment is the "algorithm" and if you share that, they won't learn what we want them to learn). We expect that what you hand in is yours.
- It is NOT acceptable to leave your code anywhere where an unscrupulous student could find and steal it (e.g., public GITHUBs, walking away while leaving yourself logged on, leaving printouts lying around,etc)
- The first offense is a zero on the assignment and an F in the course the second time
- Both Giver and Receiver are equally culpable and suffer equal penalties

Architecture of a typical Lecture



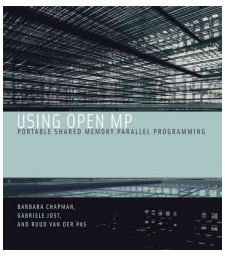
Contact Information

- Haidar M. Harmanani
  - Office: Block A, 810
  - Hours: TTh 9:00-10:30 or by appointment.
  - Email: haidar@lau.edu.lb

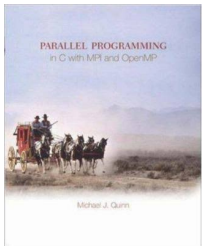
Topics

- Introduction to parallel programming
- Parallel programming performance
- Programming using pThreads, OpenMP, and GPUs
- Introduction to Neural Networks and Deep Learning using cuDNN and Python

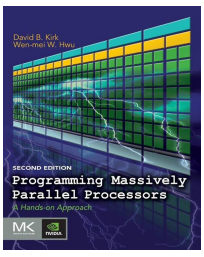
Course Materials



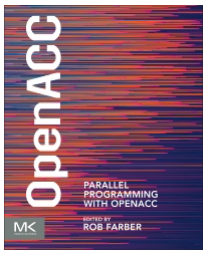
Optional



Optional



Required



Optional

Computers/Programming

- You will be using:
  - The lab or your own machines for MPI and OpenMP
  - The lab or your own machines for Pthreads and OpenMP
  - The Computer Science Lab for CUDA and GPU Programming, or your own machine if you have a CUDA-capable GPU
  - The *Cloud* for GPU Tutorial Labs
    - Access will be provided in due time
    - There will be a 5% bonus for completing at least four labs

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

Administrative Questions?

Definitions

- Parallel computing
  - Using parallel computer to solve single problems faster
- Parallel computer
  - A computer with multiple processors that supports parallel programming
- Parallel programming
  - Programming in a language that supports concurrency explicitly

Parallel Processing – What is it?

- A parallel computer is a computer system that uses multiple processing elements simultaneously in a cooperative manner to solve a computational problem
- Parallel processing includes techniques and technologies that make it possible to compute in parallel
  - Hardware, networks, operating systems, parallel libraries, languages, compilers, algorithms, tools, …
- Parallel computing is an evolution of serial computing
  - Parallelism is natural
  - Computing problems differ in level / type of parallelism
- Parallelism is all about performance! Really?

Concurrency

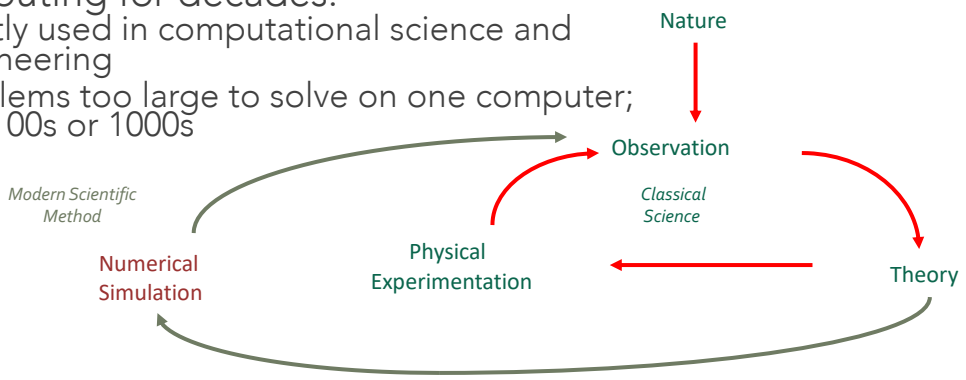
- Consider multiple tasks to be executed in a computer
- Tasks are concurrent with respect to each if
  - They can execute at the same time (concurrent execution)
  - Implies that there are no dependencies between the tasks
- Dependencies
  - If a task requires results produced by other tasks in order to execute correctly, the task's execution is *dependent*
  - If two tasks are dependent, they are not concurrent
  - Some form of synchronization must be used to enforce (satisfy) dependencies
- Concurrency is fundamental to computer science
  - Operating systems, databases, networking, …

Concurrency and Parallelism

- Concurrent is not the same as parallel! Why?
- Parallel execution
  - Concurrent tasks *actually* execute at the same time
  - Multiple (processing) resources have to be available
- Parallelism = concurrency + "parallel" hardware
  - Both are required
  - Find concurrent execution opportunities
  - Develop application to execute in parallel
  - Run application on parallel hardware
- Is a parallel application a concurrent application?
- Is a parallel application run with one processor parallel? Why or why not?

Why Parallel Computing Now?

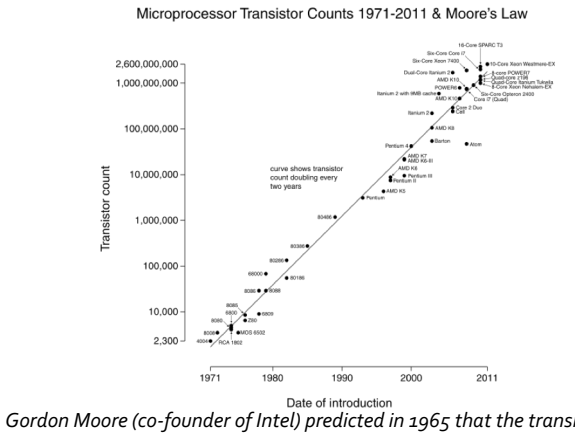
- Researchers have been using parallel computing for decades:
  - Mostly used in computational science and engineering
  - Problems too large to solve on one computer; use 100s or 1000s

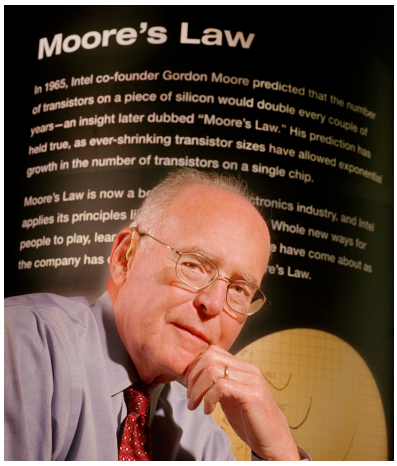


Why Parallel Computing Now?

- Many companies in the 80s/90s "bet" on parallel computing and failed
  - Computers got faster too quickly for there to be a large market
- Why an undergraduate course on parallel programming?
  - Because the entire computing industry has bet on parallelism
  - There is a desperate need for parallel programmers
- There are 3 ways to improve performance:
  - Work Harder
  - Work Smarter
  - Get Help
- Computer Analogy
  - Using faster hardware
  - Optimized algorithms and techniques used to solve computational tasks
  - Multiple computers to solve a particular task

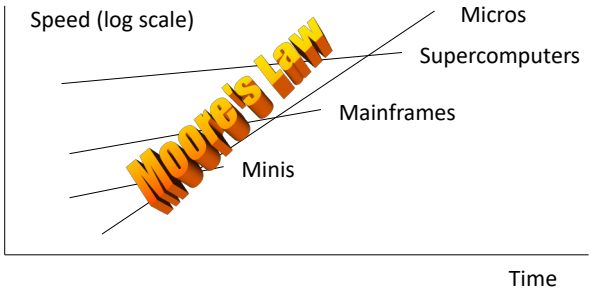
Technology Trends: Microprocessor Capacity

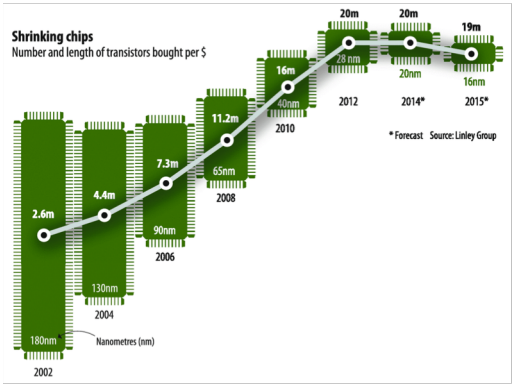




Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

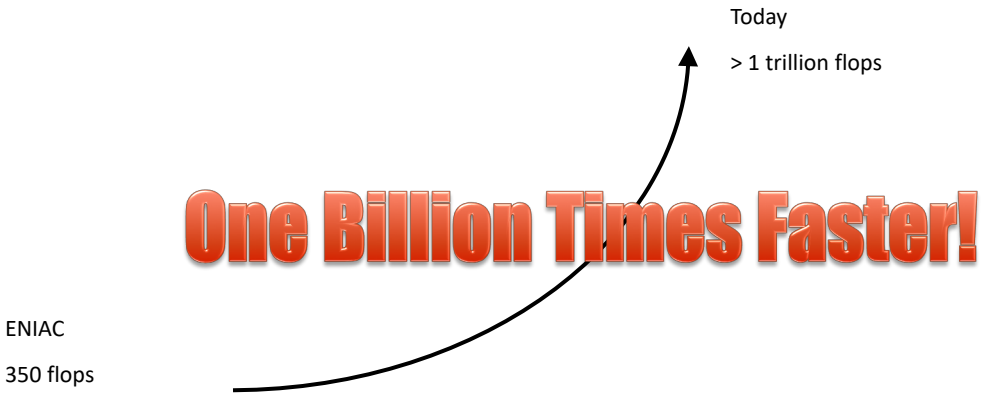
Technology Trends: Microprocessor Capacity





Microprocessors have become smaller, denser, and more powerful.

50 Years of Speed Increases



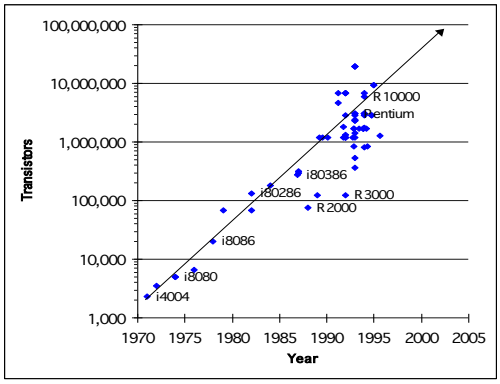
CPUs 1 Million Times Faster

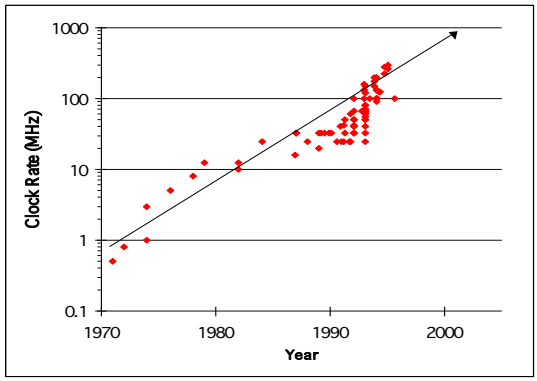
- Faster clock speeds
- Greater system concurrency
  - Multiple functional units
  - Concurrent instruction execution
  - Speculative instruction execution

Systems 1 Billion Times Faster

- Processors are 1 million times faster
- Combine thousands of processors
- Parallel computer
  - Multiple processors
  - Supports parallel programming
- Parallel computing = Using a parallel computer to execute a program faster

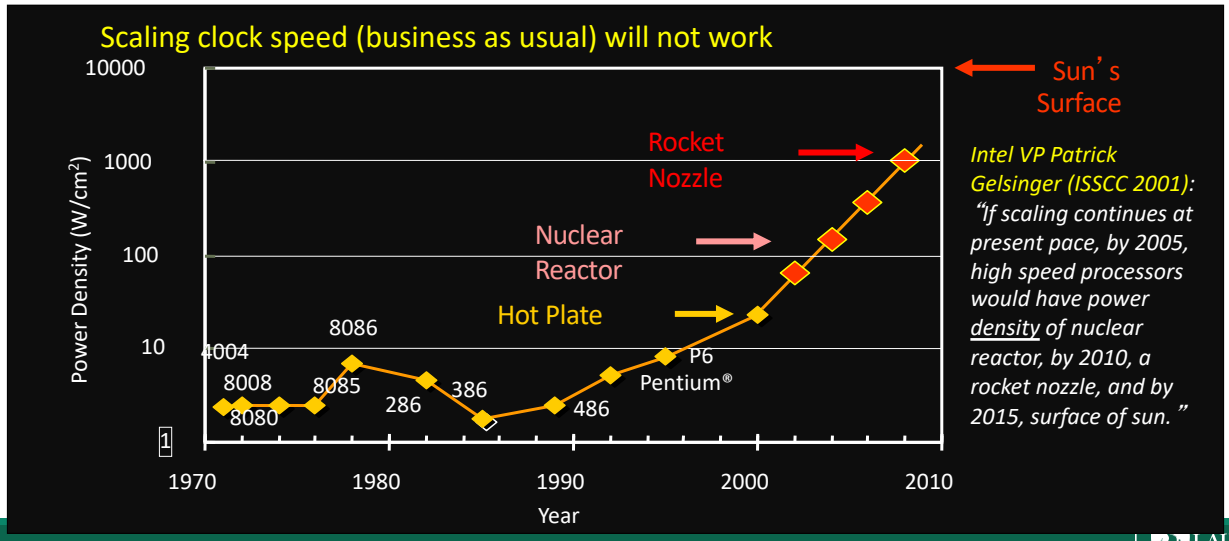
Microprocessor Transistors and Clock Rate





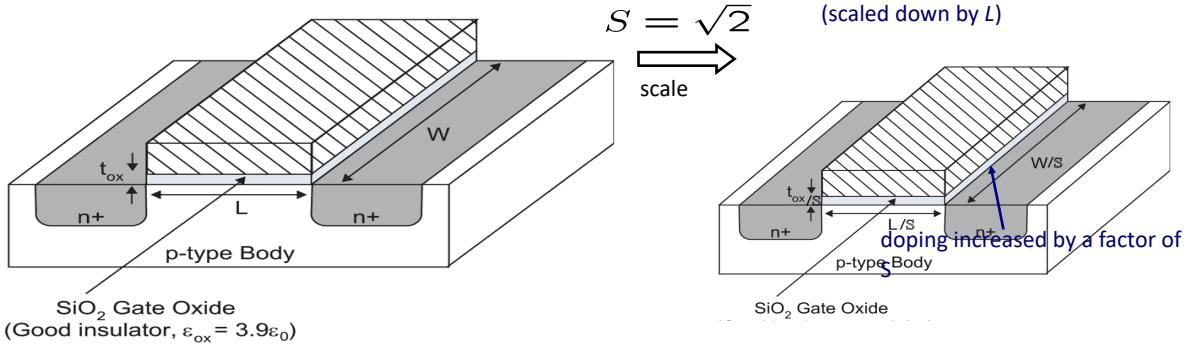
Why bother with parallel programming? Just wait a year or two…

Limit #1: Power density



Intel VP Patrick Gelsinger (ISSCC 2001): "If scaling continues at present pace, by 2005, high speed processors would have power density of nuclear reactor, by 2010, a rocket nozzle, and by 2015, surface of sun."

Device scaling



Increasing the channel doping density decreases the depletion width
⇒ improves isolation between source and drain during OFF status
⇒ permits distance between the source and drain regions to be scaled

(very idealistic NMOS transistor)

Parallelism Saves Power

- Exploit explicit parallelism for reducing power

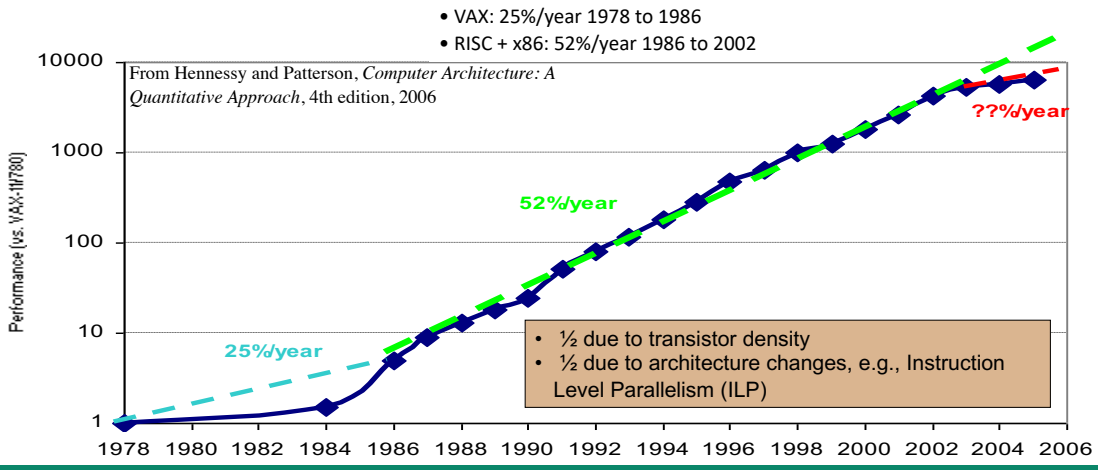
  Power = (C * V^2 * F)/4 Performance = (Cores * F)*1

  Capacitance Voltage Frequency

  - Using additional cores
    - Increase density (= more transistors = more capacitance)
    - Can increase cores (2x) and performance (2x)
    - Or increase cores (2x), but decrease frequency (1/2): same performance at ¼ the power
  - Additional benefits
    - Small/simple cores → more predictable performance

Limit #2: Hidden Parallelism Tapped Out

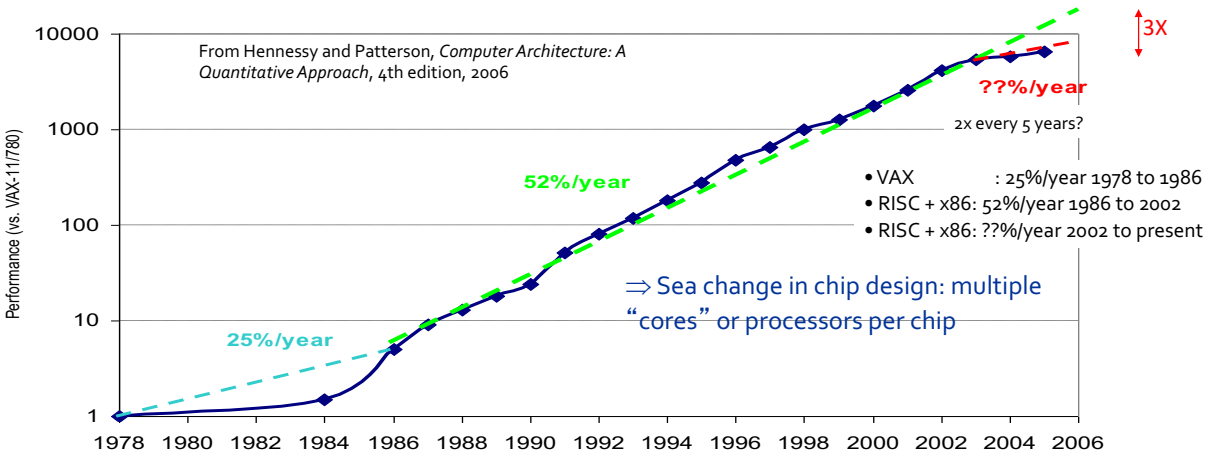
- Application performance was increasing by 52% per year as measured by the SpecInt benchmarks here



Limit #2: Hidden Parallelism Tapped Out

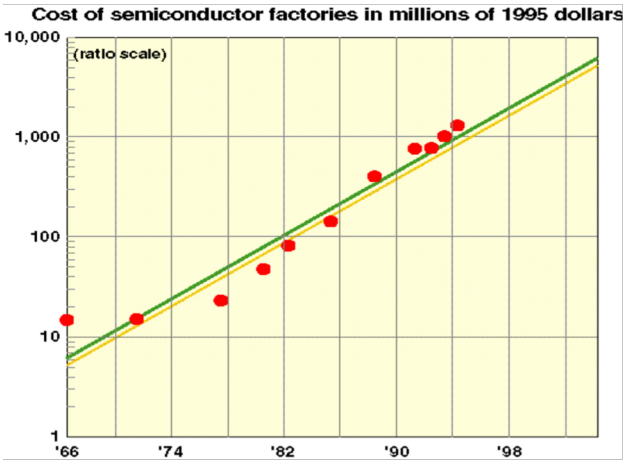
- Superscalar (SS) designs were the state of the art; many forms of parallelism not visible to programmer
  - multiple instruction issue
  - dynamic scheduling: hardware discovers parallelism between instructions
  - speculative execution: look past predicted branches
  - non-blocking caches: multiple outstanding memory ops
- You may have heard of these in CSC320, but you haven't needed to know about them to write software
- Unfortunately, these sources have been used up

Uniprocessor Performance (SPECint) Today



Limit #3: Chip Yield

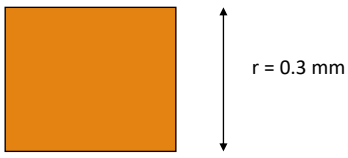
Manufacturing costs and yield problems limit use of density



- 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

Limit #4: Speed of Light (Fundamental)

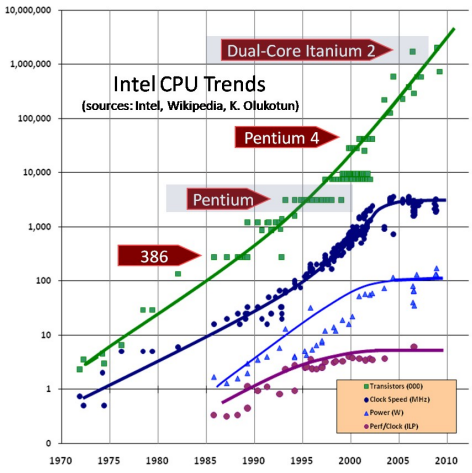
1 Tflop/s, 1 Tbyte sequential machine



- 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 1012 times per second at the speed of light, c = 3x108 m/s.
    - Thus r < c/1012 = 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



Tunnel Vision by Experts

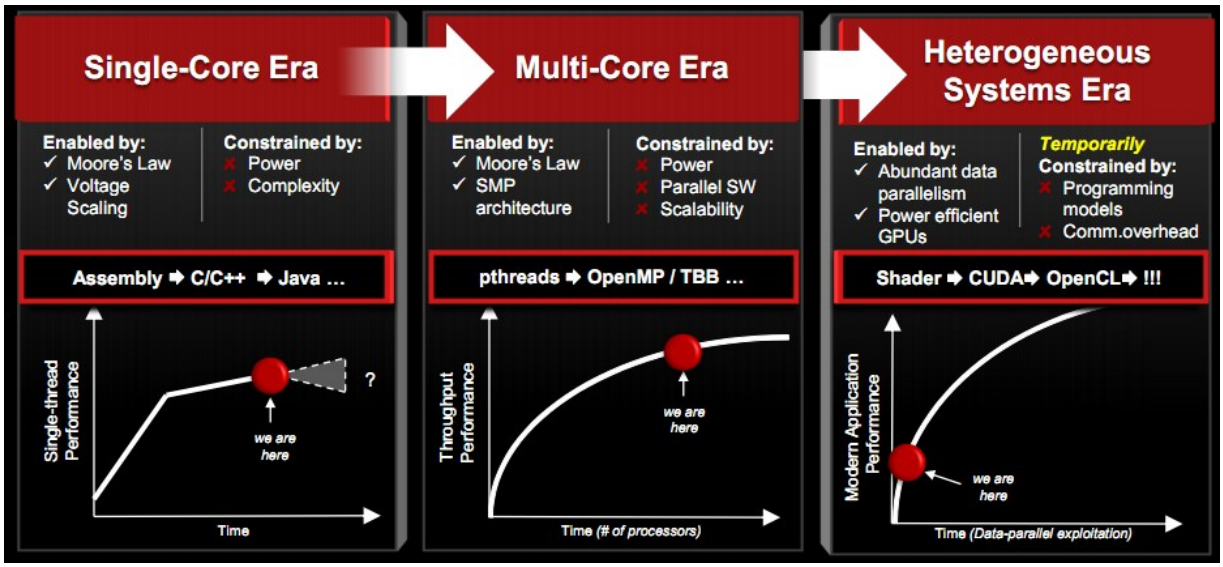
- "On several recent occasions, I have been asked whether parallel computing will soon be relegated to the trash heap reserved for promising technologies that never quite make it."
  - Ken Kennedy, CRPC Directory, 1994
- "640K [of memory] ought to be enough for anybody."
  - Bill Gates, chairman of Microsoft,1981.
- "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 Parallelism (2018)?

- 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?

Phases of Supercomputing (Parallel) Architecture

- Phase 1 (1950s): sequential instruction execution
- Phase 2 (1960s): sequential instruction issue
  - Pipeline execution, reservations stations
  - Instruction Level Parallelism (ILP)
- Phase 3 (1970s): vector processors
  - Pipelined arithmetic units
  - Registers, multi-bank (parallel) memory systems
- Phase 4 (1980s): SIMD and SMPs
- Phase 5 (1990s): MPPs and clusters
  - Communicating sequential processors
- Phase 6 (>2000): many cores, accelerators, scale, …


Single-Core Era
Enabled by:
✓ Moore's Law
✓ Voltage Scaling
Constrained by:
✗ Power
✗ Complexity
Assembly ➡ C/C++ ➡ Java …
Single-thread Performance
we are here
?
Time
Multi-Core Era
Enabled by:
✓ Moore's Law
✓ SMP architecture
Constrained by:
✗ Power
✗ Parallel SW
✗ Scalability
pthreads ➡ OpenMP / TBB …
Throughput Performance
we are here
Time (# of processors)
Heterogeneous Systems Era
Enabled by:
✓ Abundant data parallelism
✓ Power efficient GPUs
Temporarily
Constrained by:
✗ Programming models
✗ Comm.overhead
Shader ➡ CUDA➡ OpenCL➡ !!!
Modern Application Performance
we are here
Time (Data-parallel exploitation)