CSE 613: Parallel Programming

Department of Computer Science
SUNY Stony Brook
Spring 2017

"We used to joke that
"parallel computing is the future, and always will be,"
but the pessimists have been proven wrong."

— *Tony Hey*

Course Information

– **Lecture Time:** MoFr 1:00 pm - 2:20 pm
– **Location:** CS 2114, West Campus

– **Instructor:** Rezaul A. Chowdhury
– **Office Hours:** MoFr 3:00 pm - 4:30 pm, 239 New CS Building
– **Email:** rezaul@cs.stonybrook.edu

– **TA:** Unlikely

– **Class Webpage:**
http://www3.cs.stonybrook.edu/~rezaul/CSE613-S17.html

Prerequisites

- **Required:** Background in algorithms analysis
  (e.g., CSE 373 or CSE 548)
- **Required:** Background in programming languages (C / C++)
- **Helpful but Not Required:** Background in computer architecture

- **Please Note:** This is not a course on
  - Programming languages
  - Computer architecture
- **Main Emphasis:** Parallel algorithms

Topics to be Covered

The following topics will be covered

- Analytical modeling of parallel programs
- Scheduling
- Programming using the message-passing paradigm and for shared address-space platforms
- Parallel algorithms for dense matrix operations, sorting, searching, graphs, computational geometry, and dynamic programming
- Concurrent data structures
- Transactional memory, etc.

Grading Policy

- Homeworks (three: lowest score 8%, highest score 20%, and the remaining one 12%): 40%
- Group project (one): 45%
  - Proposal: Feb 17
  - Progress report: Mar 31
  - Final demo / report: May 1-5
- Scribe note (one lecture): 10%
- Class participation & attendance: 5%

Programming Environment

This course is supported by an educational grant from

- Extreme Science and Engineering Discovery Environment (XSEDE): https://www.xsede.org

We have access to the following supercomputing resources

- **Stampede (** Texas Advanced Comp. Center **):** 6,400 nodes; 16 cores (2 Intel Sandy Bridge) and 1/2 Intel Xeon Phi coprocessor(s) per node; 128 nodes are augmented with one NVIDIA Kepler 2 GPU each.
- **Comet (** San Diego Supercomputer Center **):** 1,984 nodes; 24 cores (2 Intel Haswell) per node.
- **SuperMIC (** Louisiana State University **):** 360 nodes; 20 cores (2 Intel Ivy Bridge) and 2 Intel Xeon Phi coprocessor(s) per node; 20 nodes have 1 Intel Xeon Phi and 1 NVIDIA Tesla K20X.

Programming Environment

World's Most Powerful Supercomputers in November, 2016

(www.top500.org)

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939
6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Fujitsu	556,104	13,554.6	24,913.5	2,719
7	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	206,720	9,779.0	15,988.0	1,312
9	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
10	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	4,233

Programming Environment

World's Most Powerful Supercomputers in November, 2016

(www.top500.org)

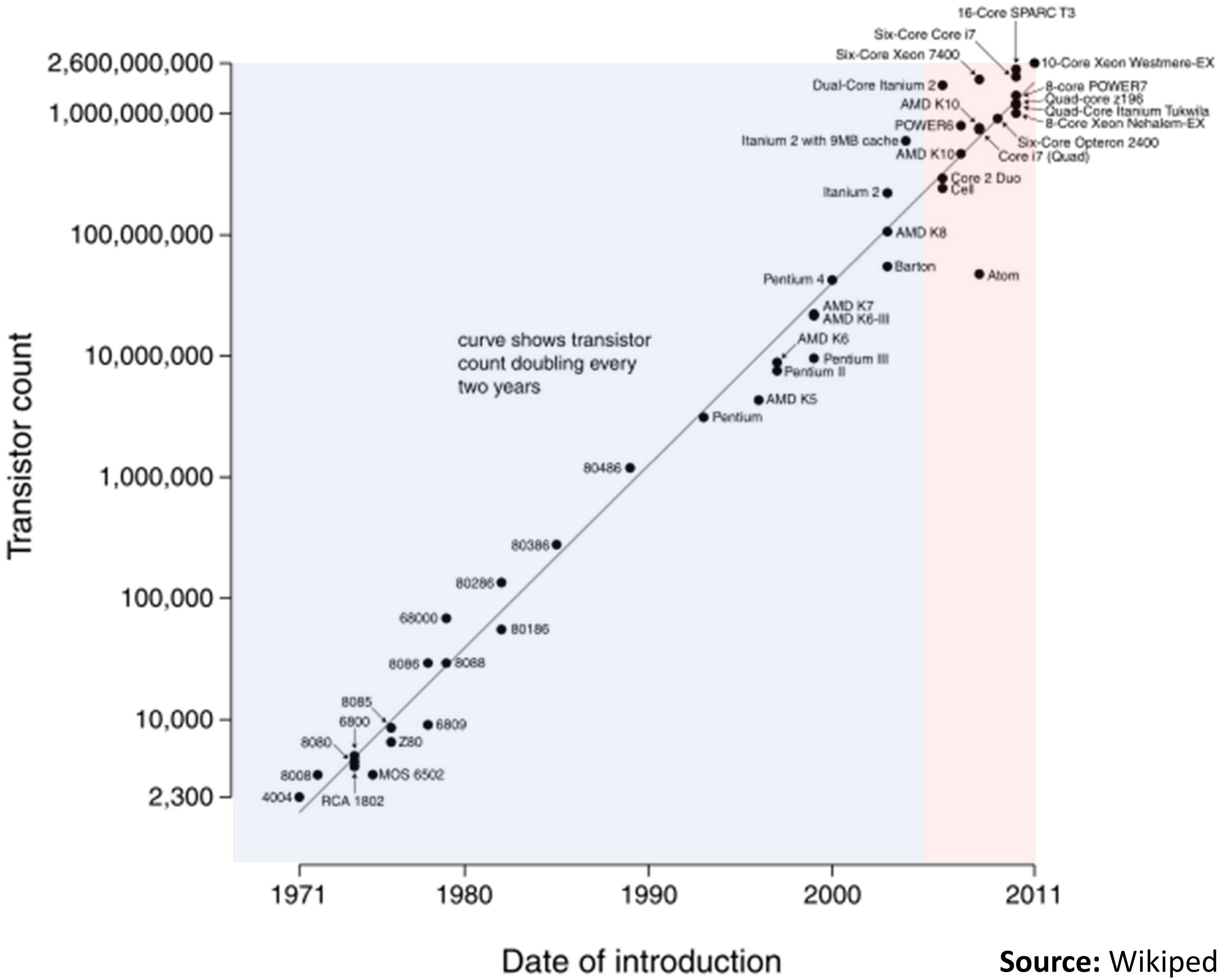
Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
11	United Kingdom Meteorological Office United Kingdom	Cray XC40, Xeon E5-2695v4 18C 2.1GHz, Aries interconnect Cray Inc.	241,920	6,765.2	8,128.5	
12	CINECA Italy	**Marconi Intel Xeon Phi** - CINECA Cluster, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Lenovo	241,808	6,223.0	10,833.0	
13	NASA/Ames Research Center/NAS United States	**Pleiades** - SGI ICE X, Intel Xeon E5-2670/E5-2680v2/E5-2680v3/E5-2680v4 2.6/2.8/2.5/2.4 GHz, Infiniband FDR HPE/SGI	241,108	5,951.6	7,107.1	4,407
14	HLRS - Höchstleistungsrechenzentrum Stuttgart Germany	**Hazel Hen** - Cray XC40, Xeon E5-2680v3 12C 2.5GHz, Aries interconnect Cray Inc.	185,088	5,640.2	7,403.5	3,615
15	King Abdullah University of Science and Technology Saudi Arabia	**Shaheen II** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	196,608	5,537.0	7,235.2	2,834
16	Total Exploration Production France	**Pangea** - SGI ICE X, Xeon Xeon E5-2670/ E5-2680v3 12C 2.5GHz, Infiniband FDR HPE/SGI	220,800	5,283.1	6,712.3	4,150
17	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462,462	5,168.1	8,520.1	4,510
18	DOE/SC/Argonne National Laboratory United States	**Theta** - Cray XC40, Intel Xeon Phi 7230 64C 1.3GHz, Aries interconnect Cray Inc.	207,360	5,095.8	8,626.2	1,087
19	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458,752	5,008.9	5,872.0	2,301
20	National Center for Atmospheric Research (NCAR) United States	**Cheyenne** - SGI ICE XA, Xeon E5-2697v4 18C 2.3GHz, Infiniband EDR HPE/SGI	144,900	4,788.2	5,332.3	1,727

Recommended Textbooks

- A. Grama, G. Karypis, V. Kumar, and A. Gupta. ***Introduction to Parallel Computing*** (2nd Edition), Addison Wesley, 2003.
- J. JáJá. ***An Introduction to Parallel Algorithms*** (1st Edition), Addison Wesley, 1992.
- T. Cormen, C. Leiserson, R. Rivest, and C. Stein. ***Introduction to Algorithms*** (3rd Edition), MIT Press, 2009.
- M. Herlihy and N. Shavit. ***The Art of Multiprocessor Programming*** (1st Edition), Morgan Kaufmann, 2008.
- P. Pacheco. ***Parallel Programming with MPI*** (1st Edition), Morgan Kaufmann, 1996.

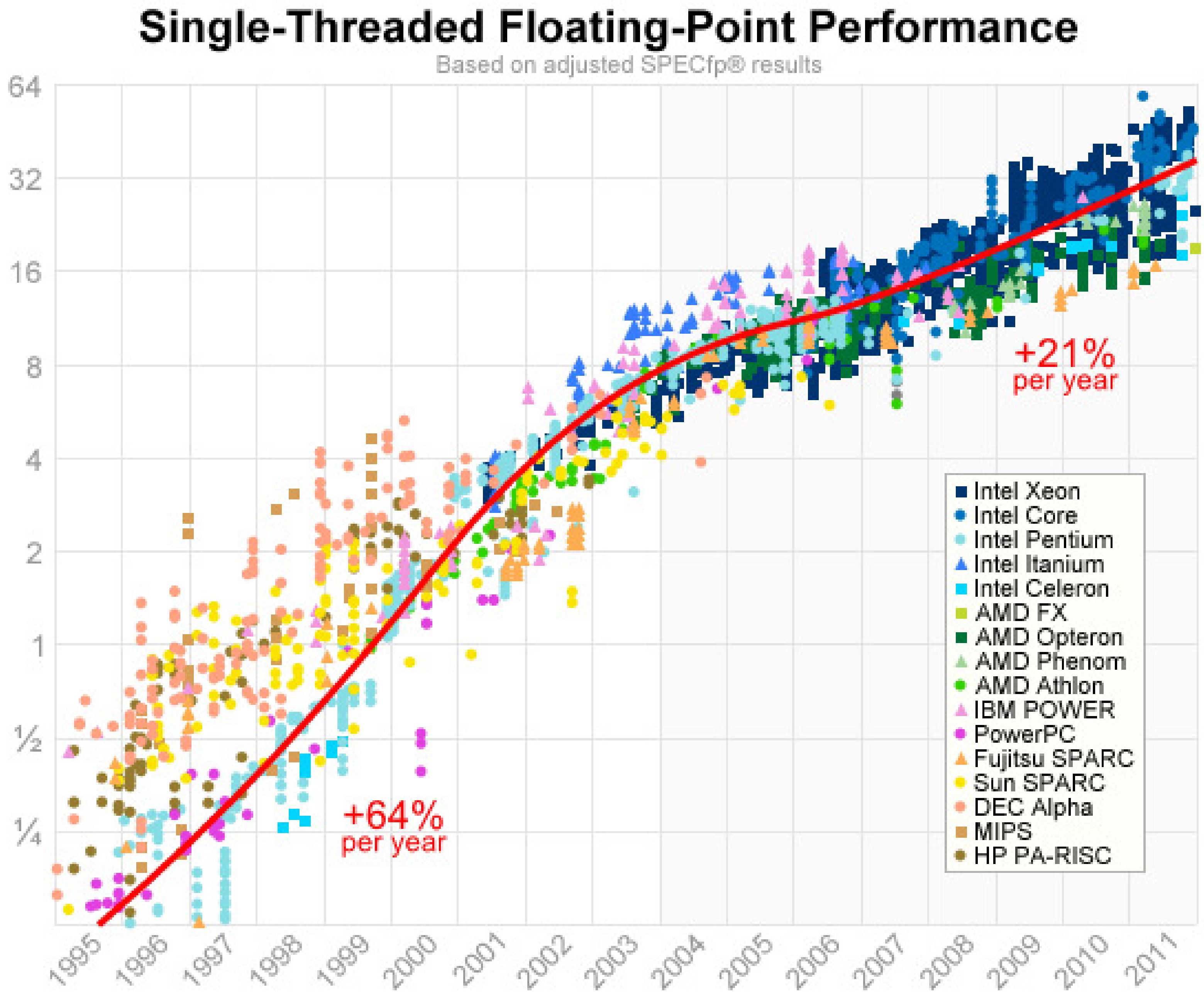
Why Parallelism?

Moore's Law



Source: Wikipedia

Unicore Performance



Source: Jeff Preshing, 2012, http://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance/

Unicore Performance Has Hit a Wall!

Some Reasons

- Lack of additional ILP
  (Instruction Level Hidden Parallelism)
- High power density
- Manufacturing issues
- Physical limits
- Memory speed

Unicore Performance: No Additional ILP

"Everything that can be invented has been invented."

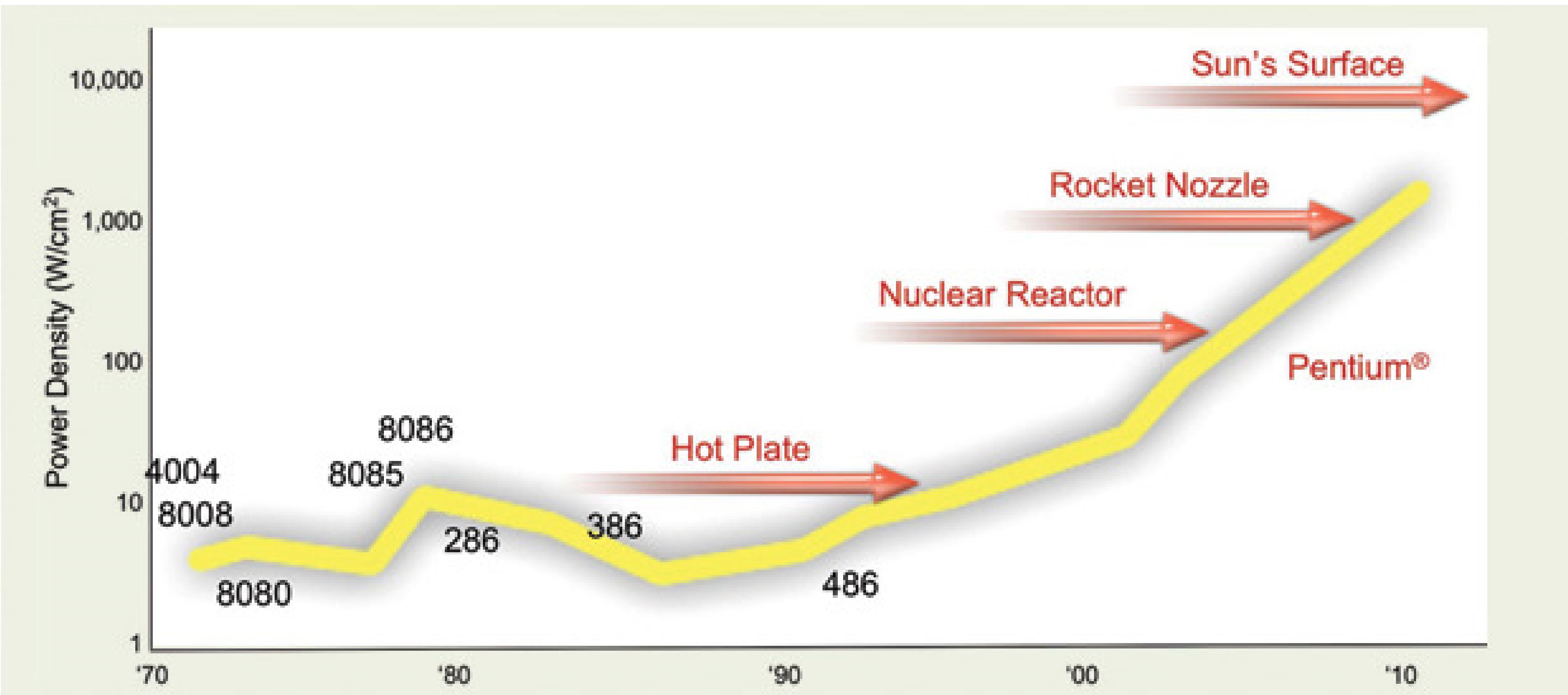
— Charles H. Duell
Commissioner, U.S. patent office, 1899

Exhausted all ideas to exploit hidden parallelism?

– Multiple simultaneous instructions
– Instruction Pipelining
– Out-of-order instructions
– Speculative execution
– Branch prediction
– Register renaming, etc.

Unicore Performance: High Power Density

- Dynamic power, $P_d \propto V^2 f C$
  - V = *supply voltage*
  - f = *clock frequency*
  - C = *capacitance*
- But $V \propto f$
- Thus $P_d \propto f^3$



Source: Patrick Gelsinger, Intel Developer Forum, Spring 2004 (Simon Floyd)

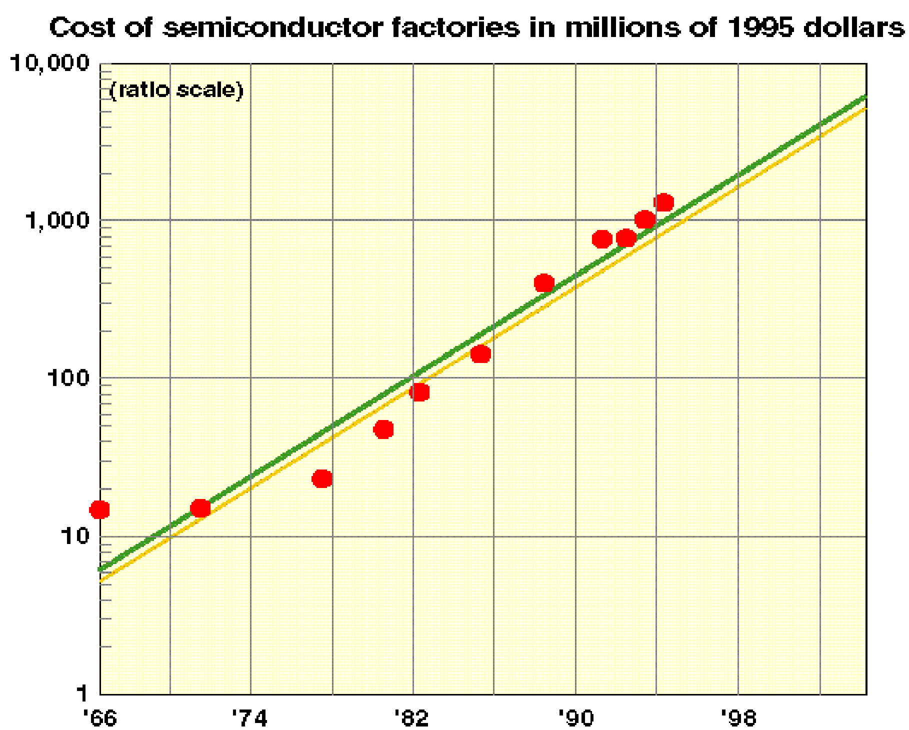
Unicore Performance: Manufacturing Issues

- Frequency, $f \propto 1 / s$
  - *s = feature size (transistor dimension)*
- Transistors / unit area $\propto 1 / s^2$
- Typically, die size $\propto 1 / s$
- So, what happens if feature size goes down by a factor of x?
  - Raw computing power goes up by a factor of x^4 !
  - Typically most programs run faster by a factor of x^3 without any change!

Source: Kathy Yelick and Jim Demmel, UC Berkeley

Unicore Performance: Manufacturing Issues

- Manufacturing cost goes up as feature size decreases
  - Cost of a semiconductor fabrication plant doubles every 4 years (Rock's Law)
- CMOS feature size is limited to 5 nm (at least 10 atoms)



Source: Kathy Yelick and Jim Demmel, UC Berkeley

Unicore Performance: Physical Limits

Execute the following loop on a serial machine in 1 second:

for ($i = 0$; $i < 10^{12}$; $++i$)

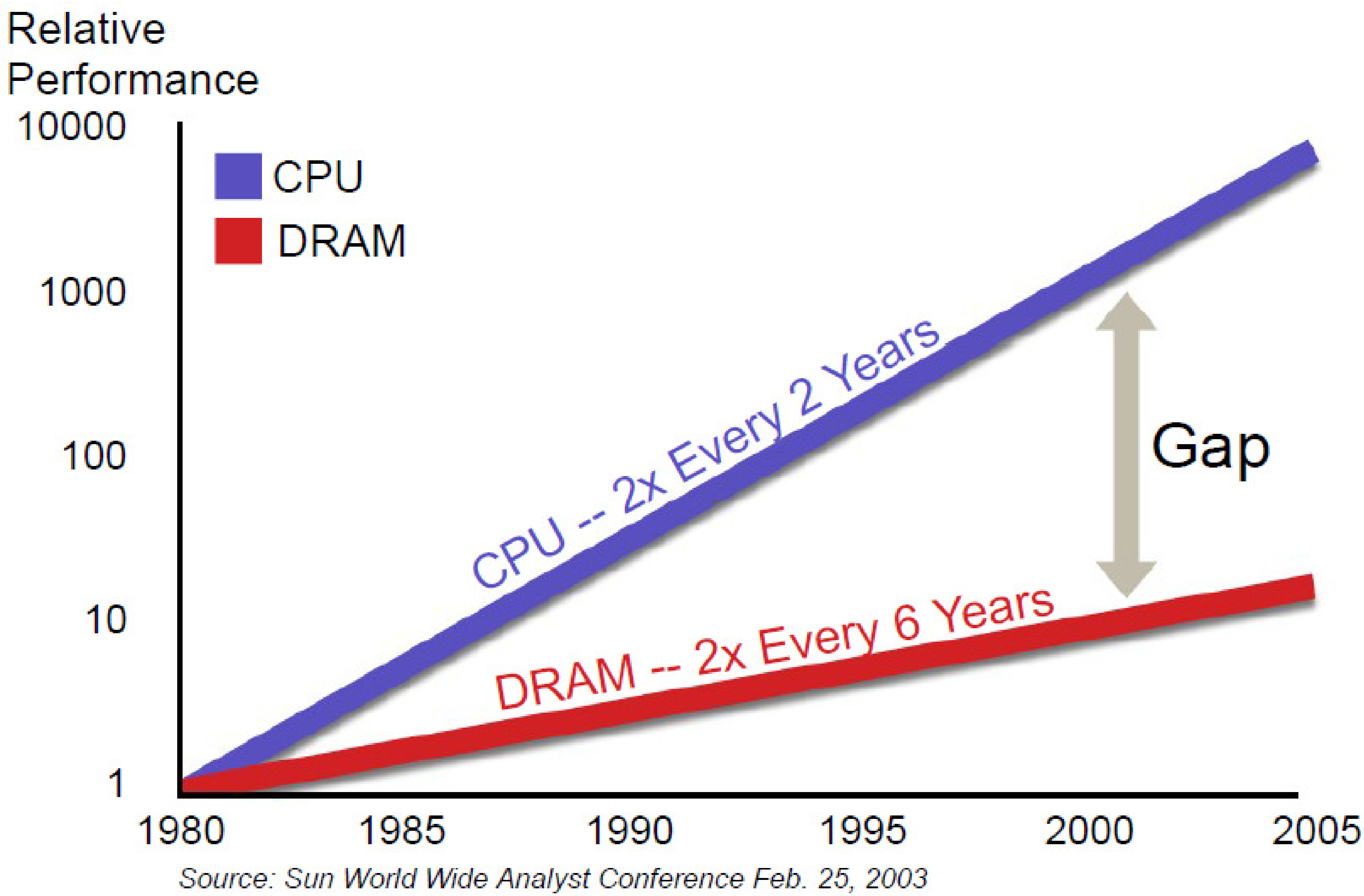
$z[i] = x[i] + y[i]$;

- We will have to access 3×10^{12} data items in one second
- Speed of light is, $c \approx 3 \times 10^{8}$ m/s
- So each data item must be within $c / 3 \times 10^{12} \approx 0.1$ mm from the CPU on the average
- All data must be put inside a 0.2 mm × 0.2 mm square
- Each data item (≥ 8 bytes) can occupy only 1 Å^2 space!

  (size of a small atom!)

Source: Kathy Yelick and Jim Demmel, UC Berkeley

Unicore Performance: Memory Wall



Source: Rick Hetherington, Chief Technology Officer, Microelectronics, Sun Microsystems

Unicore Performance Has Hit a Wall!

Some Reasons

- Lack of additional ILP
  (Instruction Level Hidden Parallelism)
- High power density
- Manufacturing issues
- Physical limits
- Memory speed

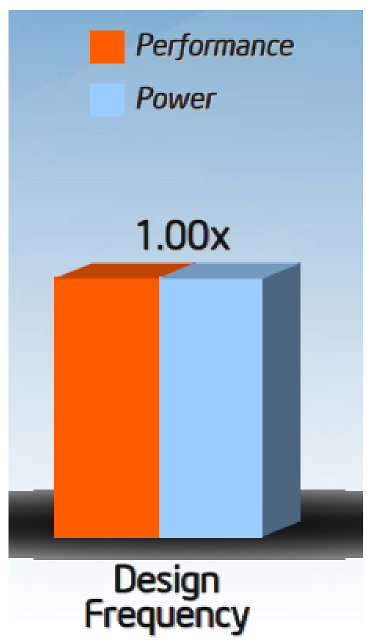
"Oh Sinnerman, where you gonna run to?"

— Sinnerman (recorded by Nina Simone)

Where You Gonna Run To?

– Changing f by 20% changes performance by 13%

– So what happens if we overclock by 20%?



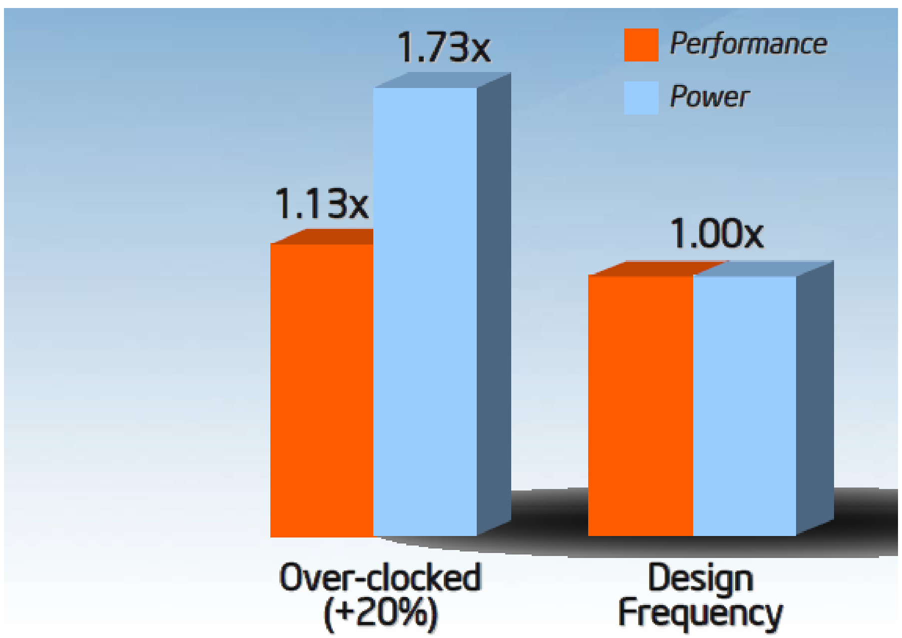
Source: Andrew A. Chien, Vice President of Research, Intel Corporation

Where You Gonna Run To?

– Changing f by 20% changes performance by 13%

– So what happens if we overclock by 20%?

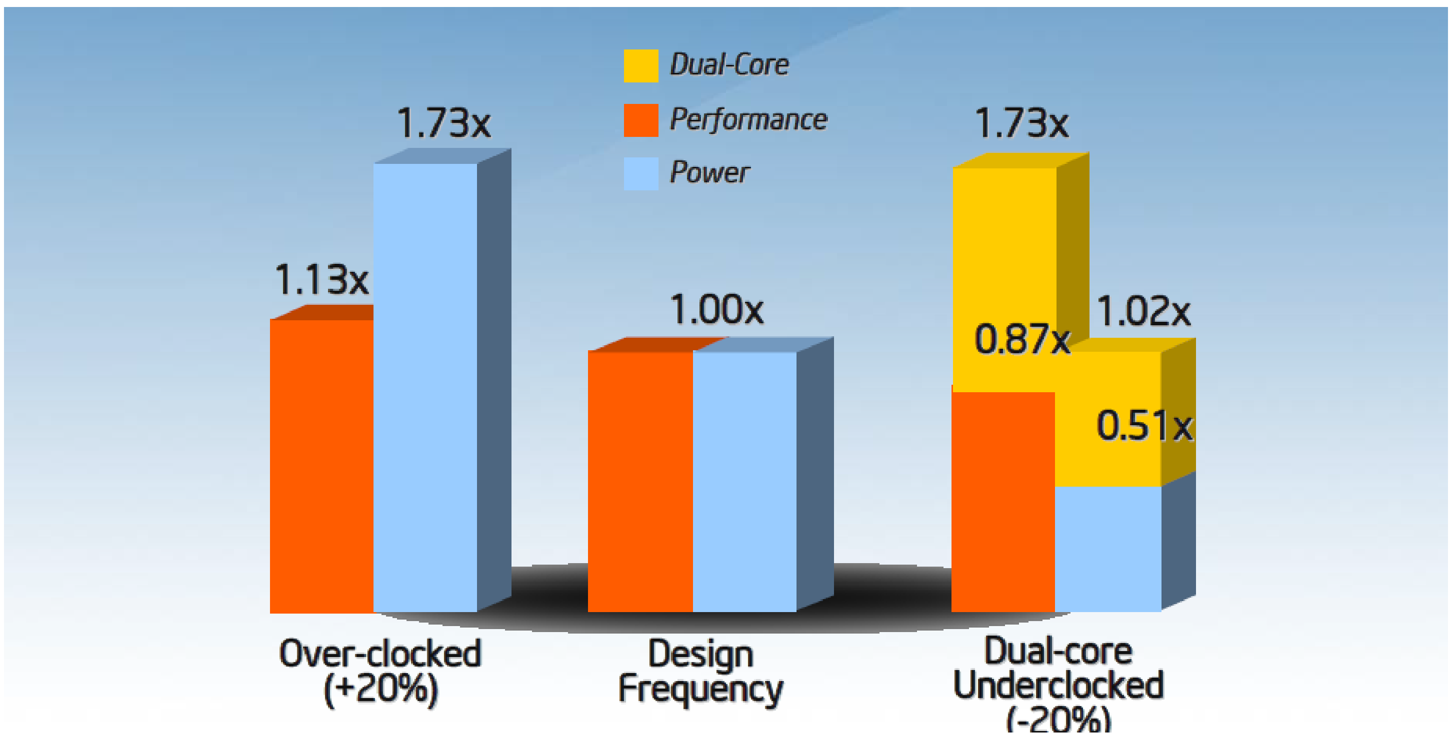
– And underclock by 20%?



Source: Andrew A. Chien, Vice President of Research, Intel Corporation

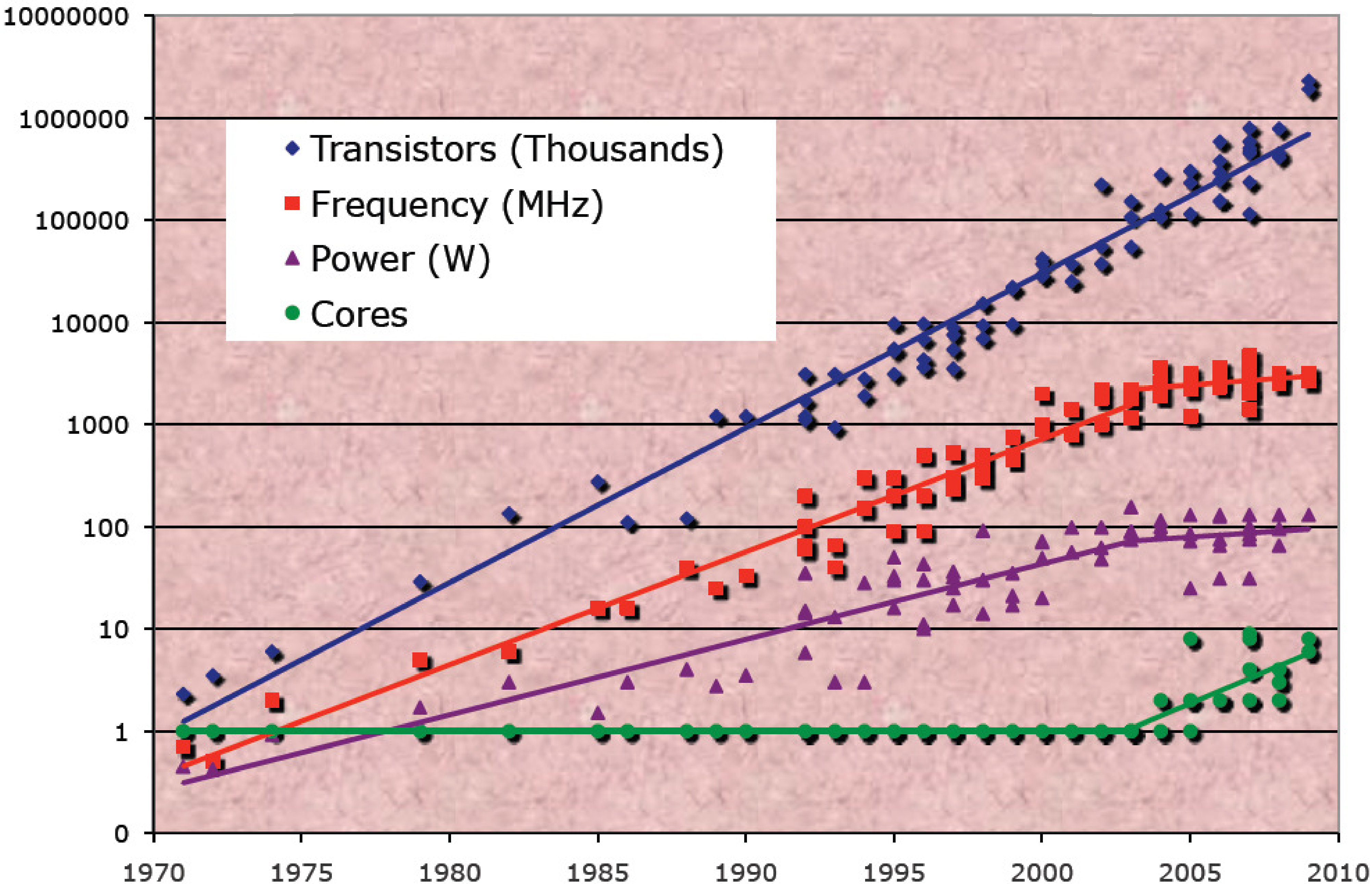
Where You Gonna Run To?

- Changing f by 20% changes performance by 13%
- So what happens if we overclock by 20%?
- And underclock by 20%?



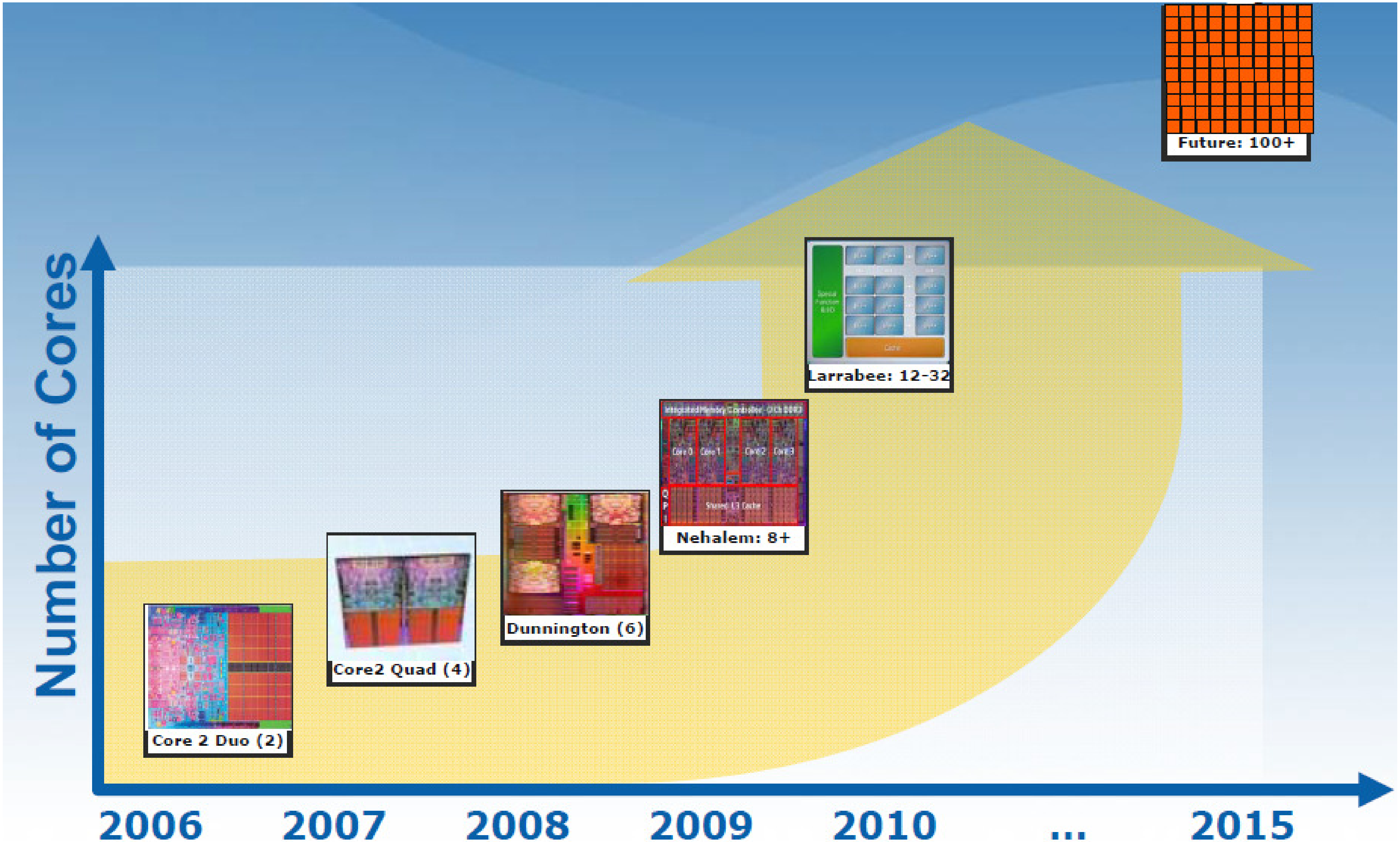
Source: Andrew A. Chien, Vice President of Research, Intel Corporation

Moore's Law Reinterpreted



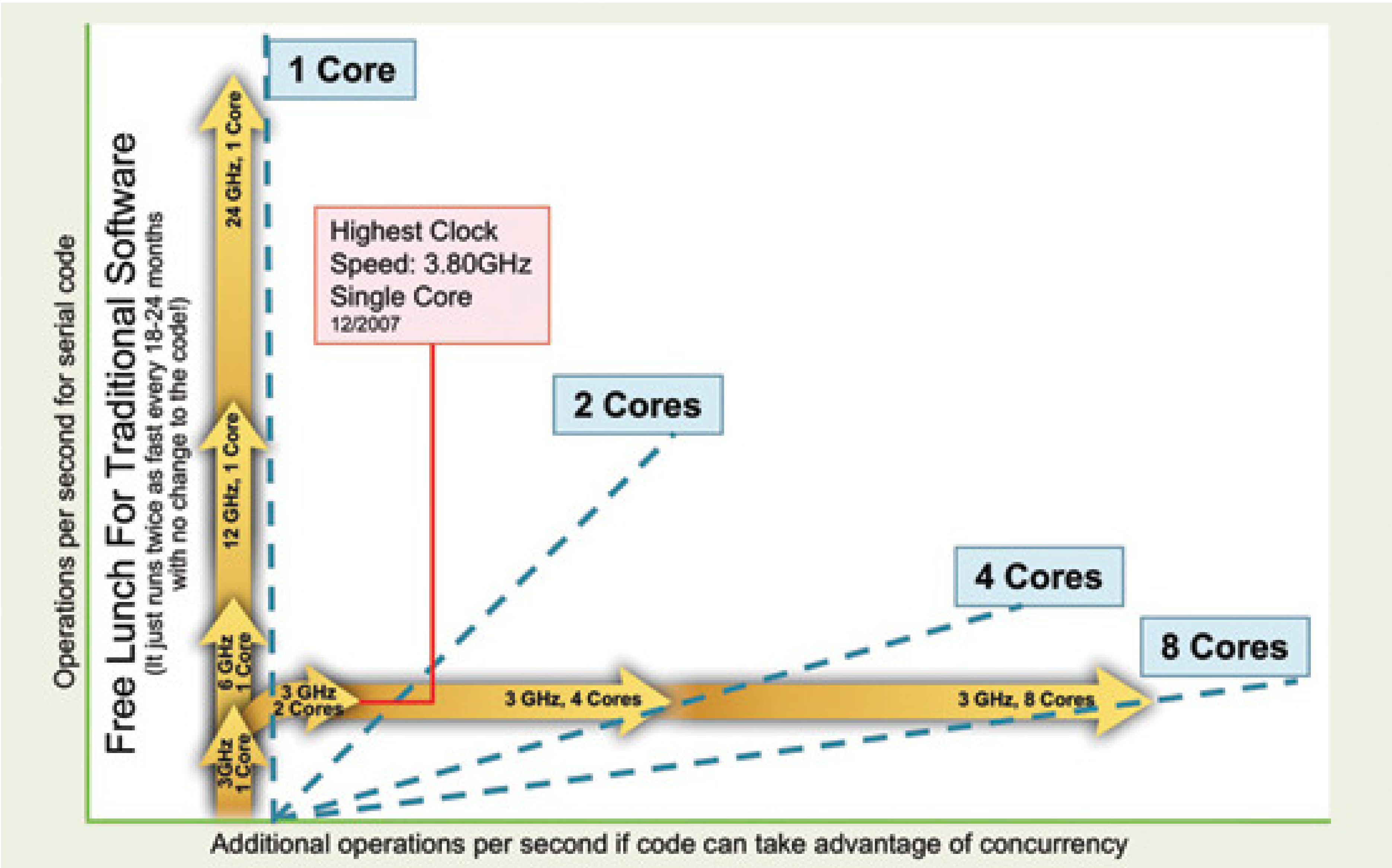
Source: Report of the 2011 Workshop on Exascale Programming Challenges

Cores / Processor (General Purpose)



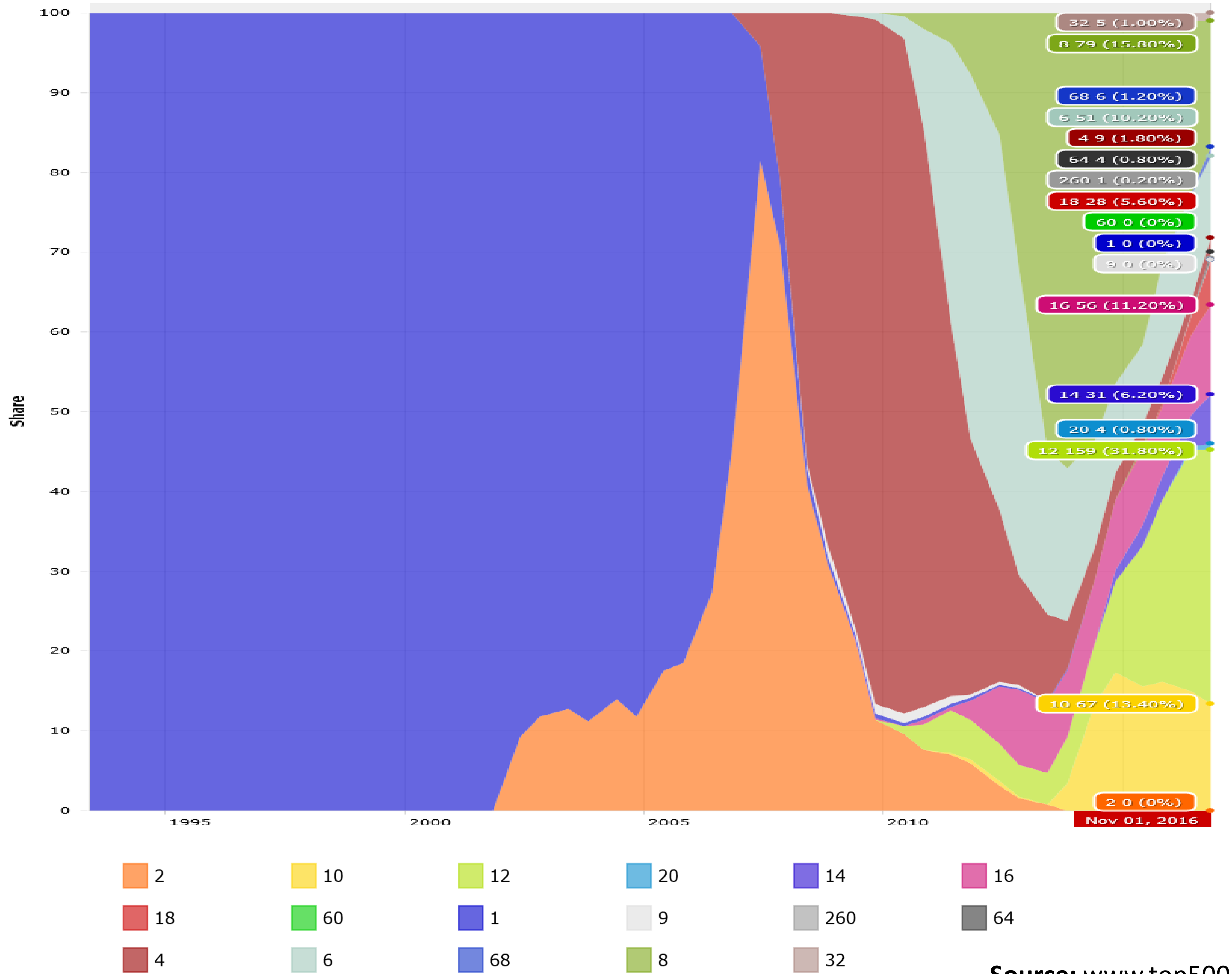
Source: Andrew A. Chien, Vice President of Research, Intel Corporation

No Free Lunch for Traditional Software



Source: Simon Floyd, Workstation Performance: Tomorrow's Possibilities (Viewpoint Column)

Top 500 Supercomputing Sites (Cores / Socket)



Source: www.top500.org

Insatiable Demand for Performance

Source: Patrick Gelsinger, Intel Developer Forum, 2008

Numerical Weather Prediction

Problem: (*temperature*, *pressure*, …, *humidity*, *wind velocity*)
$\leftarrow$ *f*(*longitude*, *latitude*, *height*, *time*)

Approach (very coarse resolution):

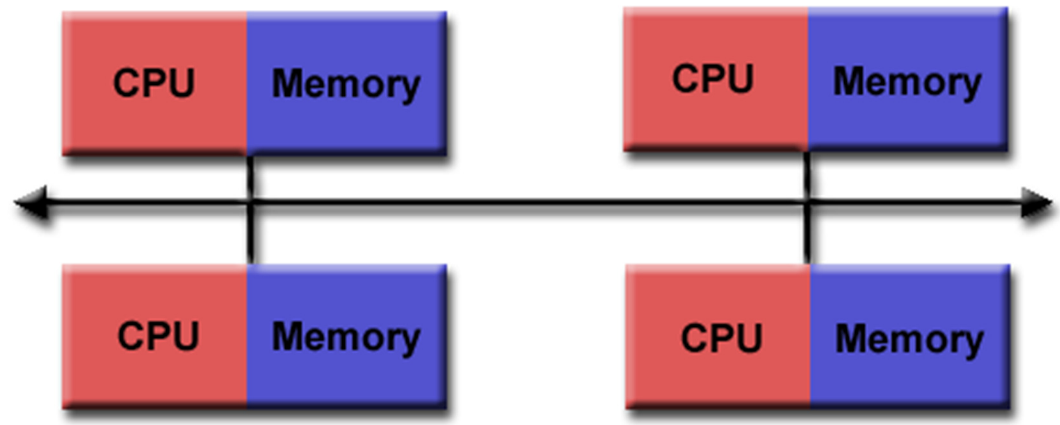
- Consider only modeling fluid flow in the atmosphere
- Divide the entire global atmosphere into cubic cells of size 1 mile × 1 mile × 1 mile each to a height of 10 miles
  $\approx 2 \times 10^9$ cells
- Simulate 7 days in 1 minute intervals
  $\approx 10^4$ time-steps to simulate
- 200 floating point operations (flop) / cell / time-step
  $\approx 4 \times 10^{15}$ floating point operations in total
- To predict in 1 hour $\approx$ 1 Tflop/s (Tera flop / sec)

Some Useful Classifications of Parallel Computers

Parallel Computer Memory Architecture
(Distributed Memory)

– Each processor has its own local memory — no global address space

– Changes in local memory by one processor have no effect on memory of other processors



Source: Blaise Barney, LLNL

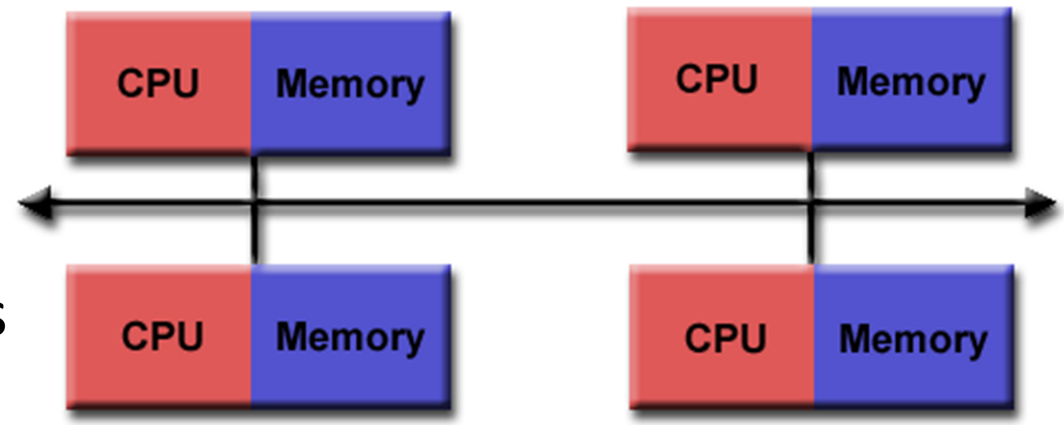
– Communication network to connect inter-processor memory

– Programming
  – Message Passing Interface (MPI)
  – Many once available: PVM, Chameleon, MPL, NX, etc.

Parallel Computer Memory Architecture (Distributed Memory)

Advantages

- Easily scalable
- No cache-coherency needed among processors
- Cost-effective



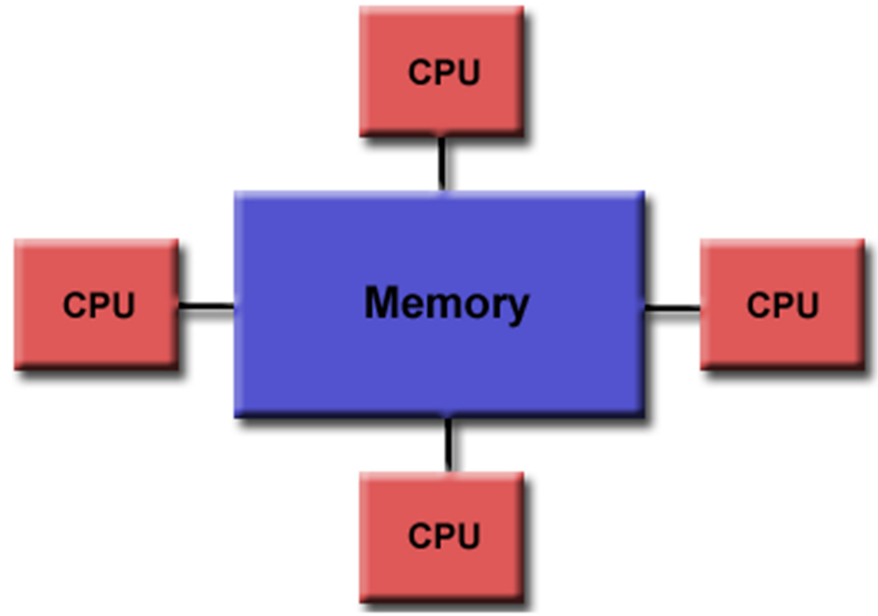
Source: Blaise Barney, LLNL

Disadvantages

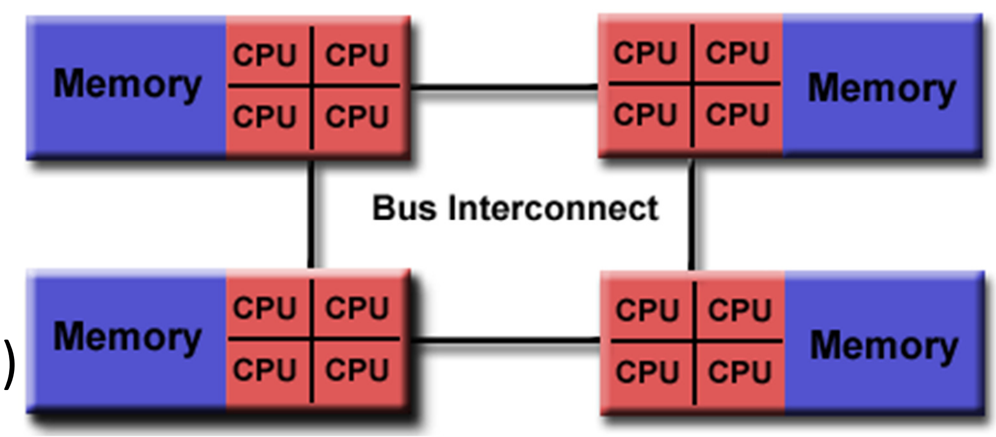
- Communication is user responsibility
- Non-uniform memory access
- May be difficult to map shared-memory data structures to this type of memory organization

Parallel Computer Memory Architecture (Shared Memory)

– All processors access all memory as global address space
– Changes in memory by one processor are visible to all others
– Two types
  – Uniform Memory Access (UMA)
  – Non-Uniform Memory Access (NUMA)
– Programming
  – Open Multi-Processing (OpenMP)
  – Cilk/Cilk++ and Intel Cilk Plus
  – Intel Thread Building Block (TBB), etc.



UMA

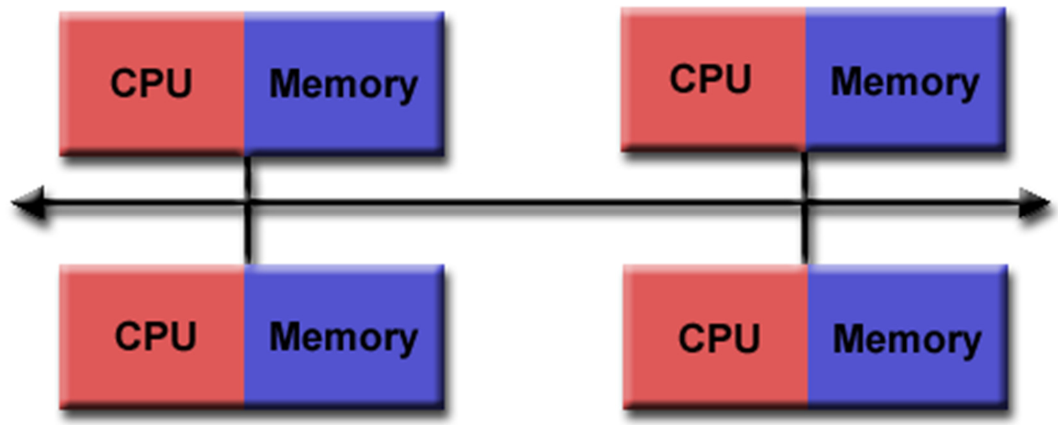


NUMA

Source: Blaise Barney, LLNL

Parallel Computer Memory Architecture (Distributed Memory)

Advantages



Source: Blaise Barney, LLNL

– Easily scalable
– No cache-coherency needed among processors
– Cost-effective

Disadvantages

– Communication is user responsibility
– Non-uniform memory access
– May be difficult to map shared-memory data structures to this type of memory organization

Parallel Computer Memory Architecture (Hybrid Distributed-Shared Memory)

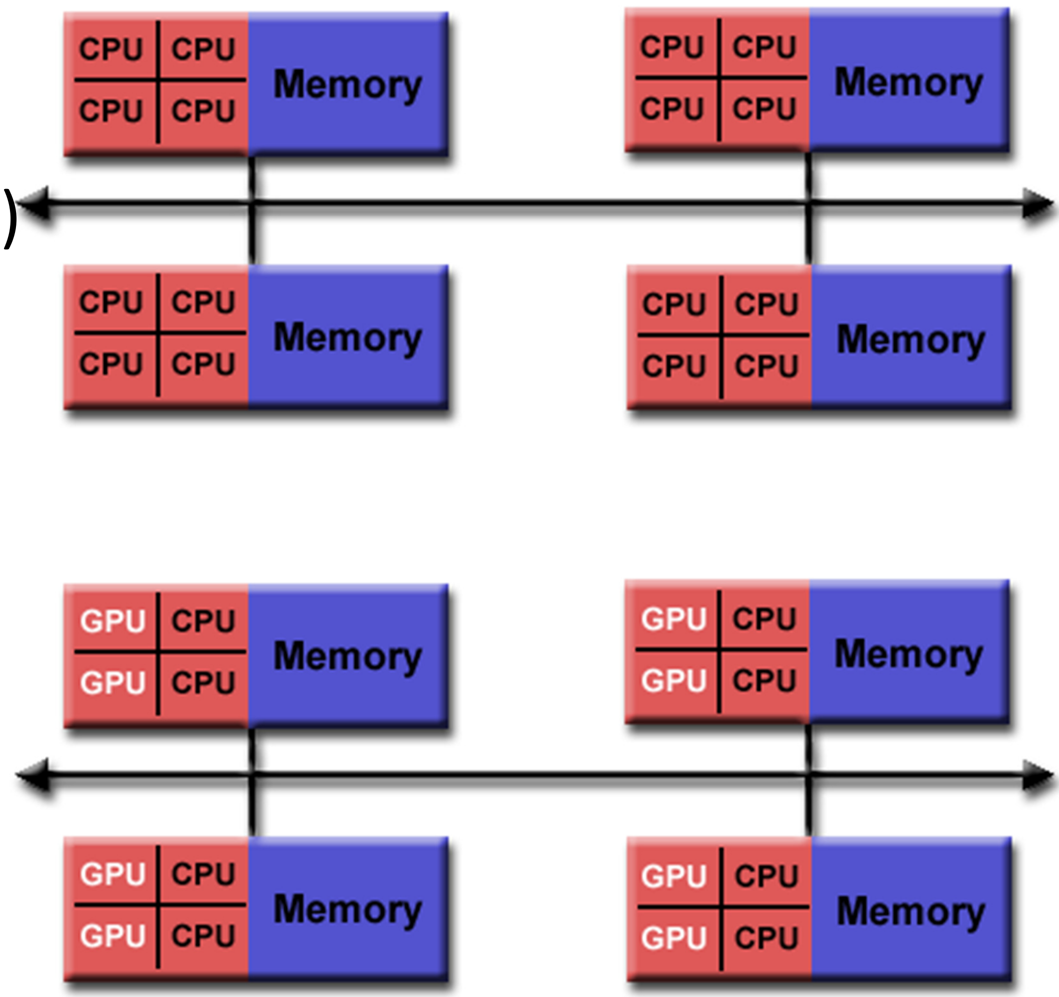
– The share-memory component can be a cache-coherent SMP or a Graphics Processing Unit (GPU)

– The distributed-memory component is the networking of multiple SMP/GPU machines

– Most common architecture for the largest and fastest computers in the world today

– Programming

  – OpenMP / Cilk + CUDA / OpenCL + MPI, etc.



Source: Blaise Barney, LLNL

Flynn's Taxonomy of Parallel Computers

Flynn's classical taxonomy (1966):

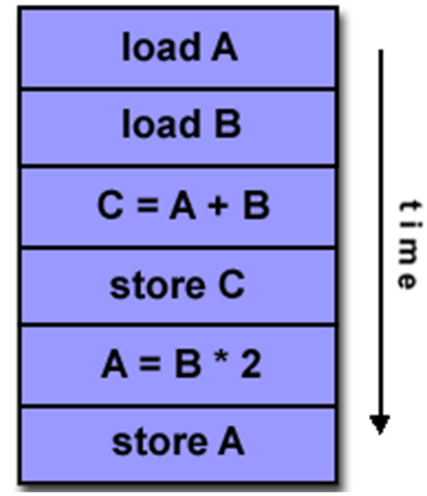
Classification of multi-processor computer architectures along two independent dimensions of *instruction* and *data*.

	Single Data (SD)	Multiple Data (MD)
Single Instruction (SI)	**SISD**	**SIMD**
Multiple Instruction (MI)	**MISD**	**MIMD**

Flynn's Taxonomy of Parallel Computers

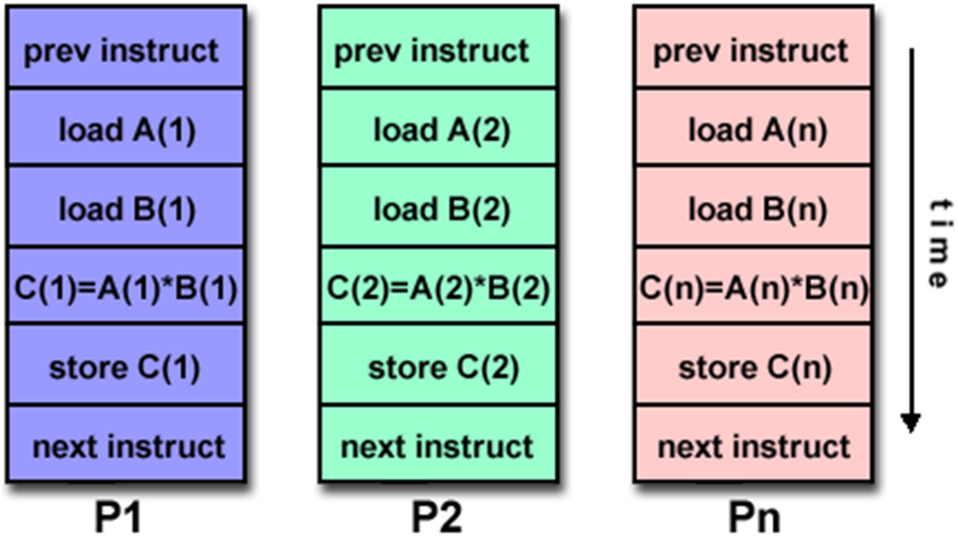
SISD

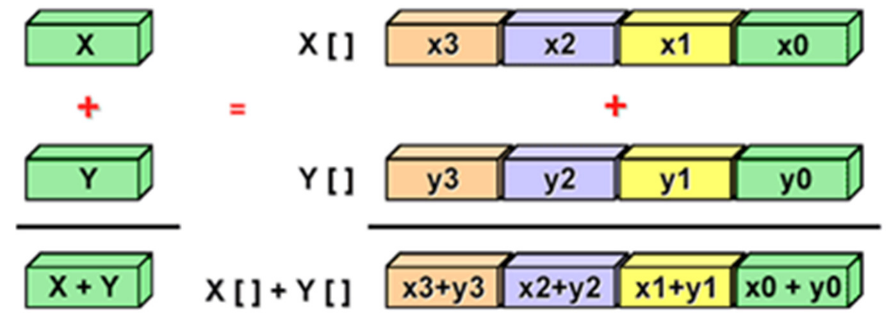
- A serial (non-parallel) computer
- The oldest and the most common type of computers
- Example: Uniprocessor unicore machines



Source: Blaise Barney, LLNL

Flynn's Taxonomy of Parallel Computers





Source: Blaise Barney, LLNL

SIMD

– A type of parallel computer

– All PU's run the same instruction at any given clock cycle

– Each PU can act on a different data item

– Synchronous (lockstep) execution

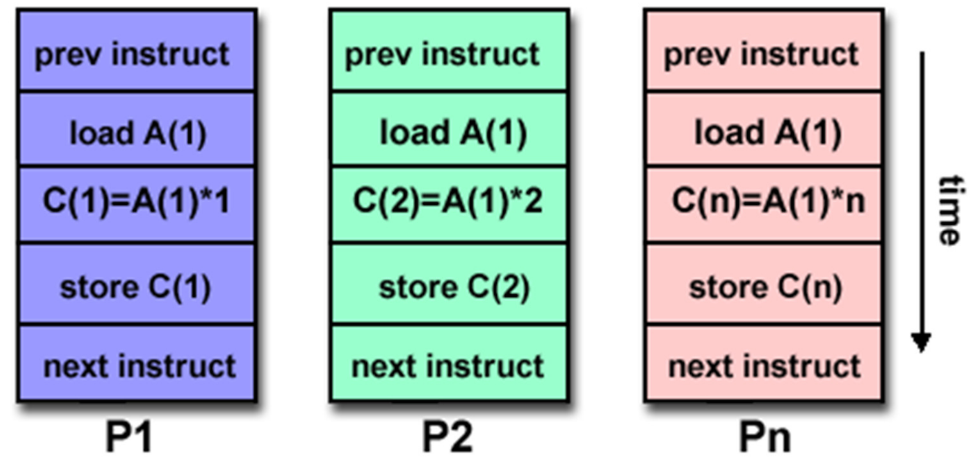
– Two types: processor arrays and vector pipelines

– Example: GPUs (Graphics Processing Units)

Flynn's Taxonomy of Parallel Computers

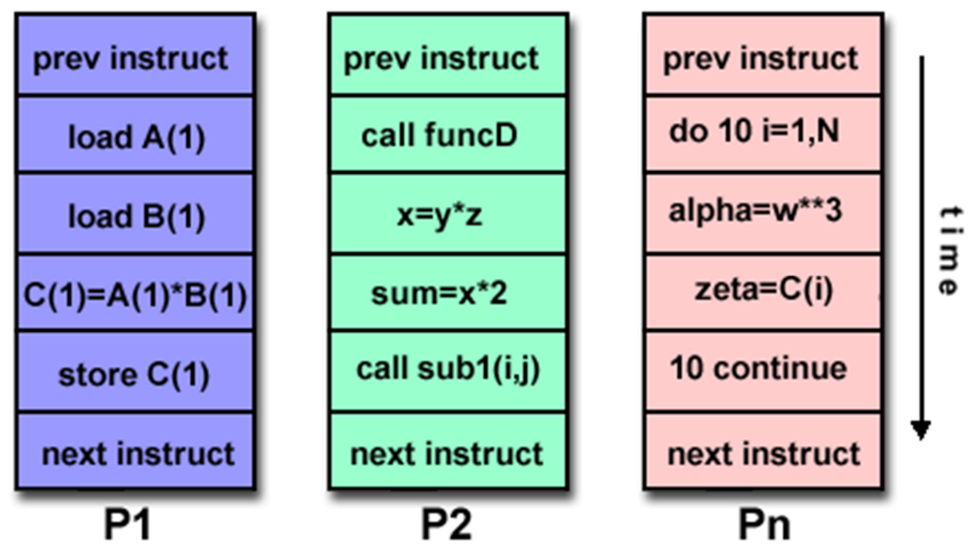
MISD

- A type of parallel computer
- Very few ever existed



MIMD

- A type of parallel computer
- Synchronous /asynchronous execution
- Examples: most modern supercomputers, parallel computing clusters, multicore PCs



Source: Blaise Barney, LLNL

Parallel Algorithms Warm-up

"The way the processor industry is going, is to add more and more cores, but nobody knows how to program those things. I mean, two, yeah; four, not really; eight, forget it."

— Steve Jobs, NY Times interview, June 10 2008

Parallel Algorithms Warm-up (1)

Consider the following loop:

$$\textit{for } i = 1 \textit{ to } n \textit{ do}$$
$$C[\,i\,] \leftarrow A[\,i\,] \times B[\,i\,]$$

– Suppose you have an infinite number of processors/cores

– Ignore all overheads due to scheduling, memory accesses, communication, etc.

– Suppose each operation takes a constant amount of time

– How long will this loop take to complete execution?

Parallel Algorithms Warm-up (1)

Consider the following loop:

$$\textit{for } i = 1 \textit{ to } n \textit{ do}$$
$$C[\, i\,] \leftarrow A[\, i\,] \times B[\, i\,]$$

– Suppose you have an infinite number of processors/cores

– Ignore all overheads due to scheduling, memory accesses, communication, etc.

– Suppose each operation takes a constant amount of time

– How long will this loop take to complete execution?

  – $O(1)$ time

Parallel Algorithms Warm-up (2)

Now consider the following loop:

$$c \leftarrow 0$$

$$\textit{for } i = 1 \textit{ to } n \textit{ do}$$

$$\quad c \leftarrow c + A[\, i \,] \times B[\, i \,]$$

– How long will this loop take to complete execution?

Parallel Algorithms Warm-up (2)

Now consider the following loop:

$$c \leftarrow 0$$

$$\textit{for } i = 1 \textit{ to } n \textit{ do}$$

$$\quad c \leftarrow c + A[\,i\,] \times B[\,i\,]$$

– How long will this loop take to complete execution?

– $O(\log n)$ time

Parallel Algorithms Warm-up (3)

Now consider quicksort:

$QSort(A)$

$\text{if } |A| \leq 1 \text{ return } A$

$\text{else} \quad p \leftarrow A[\,rand(\,|A|\,)\,]$

$\quad\quad \text{return } QSort(\{\, x \in A : x < p \,\})$

$\quad\quad\quad\quad \# \{\, p \,\} \#$

$\quad\quad\quad QSort(\{\, x \in A : x > p \,\})$

– Assuming that A is split in the middle everytime, and the two recursive calls can be made in parallel, how long will this algorithm take?

Parallel Algorithms Warm-up (3)

Now consider quicksort:

$QSort(\ A\ )$

$\textit{if}\ |A| \leq 1\ \textit{return}\ A$

$\textit{else}\quad p \leftarrow A[\ \textit{rand}(\ |A|\ )\ ]$

$\text{return}\ QSort(\ \{\ x \in A : x < p\ \}\ )$

$\#\ \{\ p\ \}\ \#$

$QSort(\ \{\ x \in A : x > p\ \}\ )$

– Assuming that A is split in the middle everytime, and the two recursive calls can be made in parallel, how long will this algorithm take?

  – $O(\ \log^2 n\ )$ (if partitioning takes logarithmic time)
  – $O(\ \log n\ )$ (but can be partitioned in constant time)