CSE 638: Advanced Algorithms

Department of Computer Science
SUNY Stony Brook
Spring 2013

"For every complex problem, there is a solution that is simple, neat, and wrong."

— Henry Louis Mencken

Course Information

- **Lecture Time:** TuTh 2:30 pm - 3:50 pm
- **Location:** Melville Library E4540, West Campus

- **Instructor:** Rezaul A. Chowdhury
- **Office Hours:** TuTh 12:30 pm - 2:00 pm, 1421 Computer Science
- **Email:** rezaul@cs.stonybrook.edu

- **TA:** Vikas Ganjigunte Ashok
- **TA Office Hours:** Tu 4:00 pm - 5:00 pm, 2110 Computer Science
- **TA Email:** vganjiguntea@cs.stonybrook.edu

- **Class Webpage:** http://www.cs.sunysb.edu/~cse638

Prerequisites

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

Topics to be Covered

Various topics from the following areas will be covered

- **Parallel algorithms (most emphasis)**
- Randomized algorithms
- External-memory and cache-efficient algorithms
- Streaming algorithms
- Resilient algorithms

Grading Policy

- Homeworks (three: highest 15%, lowest 5%, other 10%): 30%
- Exam (one): 25%
    - Final (in-class): May 9
- Group project (one): 30%
    - Proposal: Feb 28
    - Progress report (in-class): April 2 - 4
    - Final report: May 10
- Scribe note (one lecture): 10%
- Class participation & attendance: 5%

Programming Environment

This course is supported by educational grants from

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

We have access to the following supercomputers

- **Lonestar (Texas Advanced Computing Center):** 1,800+ nodes with 12 cores (two Intel Westmere processors) per node
- **Trestles (San Diego Supercomputer Center):** 300+ nodes with 32 cores (four AMD Magny Cours processors) per node
- **Kraken (National Institute for Computational Sciences):** 9,000+ nodes with 12 cores (two AMD Opteron Istanbul processors) per node
- **Keeneland KIDS (Georgia Tech):** 120 nodes with 16 cores (two Intel Sandy Bridge processors) and three NVIDIA Fermi GPU's per node

Programming Environment

World's Most Powerful Supercomputers in June, 2008

(www.top500.org)

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/NNSA/LANL United States	**Roadrunner** - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband IBM	122400	1026.0	1375.8	2345
2	DOE/NNSA/LLNL United States	**BlueGene/L** - eServer Blue Gene Solution IBM	212992	478.2	596.4	2329
3	National Institute for Computational Sciences/University of Tennessee United States	**Kraken XT5** - Cray XT5 QC 2.3 GHz Cray Inc.	66000	463.3	607.2	

Programming Environment

World's Most Powerful Supercomputers in November, 2012 (www.top500.org)

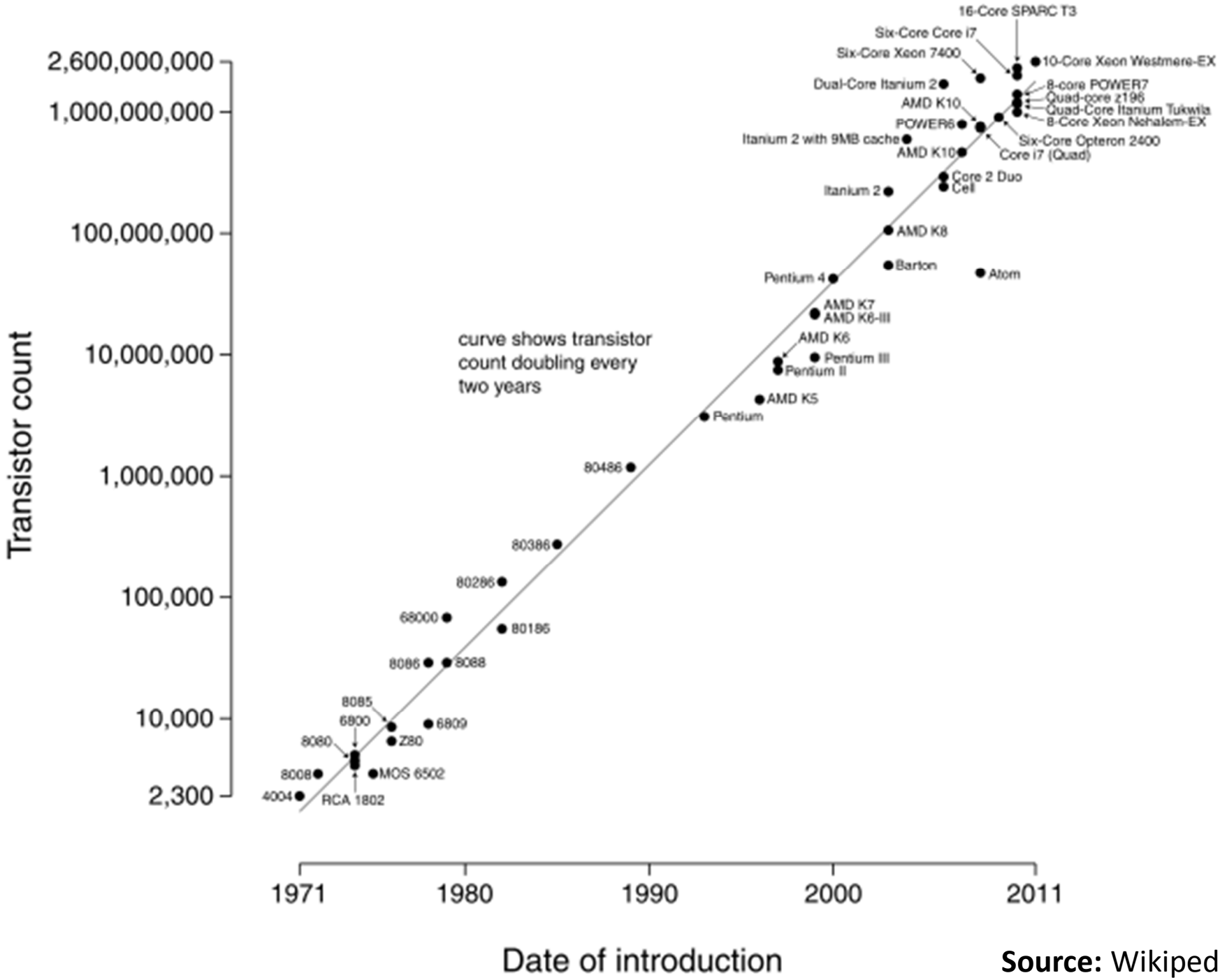
Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
2	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	16324.8	20132.7	7890
3	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
25	National Institute for Computational Sciences/University of Tennessee United States	**Kraken XT5** - Cray XT5-HE Opteron Six Core 2.6 GHz Cray Inc.	112800	919.1	1173.0	3090
96	Texas Advanced Computing Center/Univ. of Texas United States	**Lonestar 4** - Dell PowerEdge M610 Cluster, Xeon 5680 3.3Ghz, Infiniband QDR Dell	22656	251.8	301.8	

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.
- R. Motwani and P. Raghavan. ***Randomized Algorithms*** (1st Edition), Cambridge University Press, 1995.
- J. Vitter. ***Algorithms and Data Structures for External Memory***, Series on Foundations and Trends in Theoretical Computer Science, Now Publishers, Hanover, MA, 2008.
- U. Meyer, P. Sanders, and J. Sibeyn (Editors). ***Algorithms for Memory Hierarchies: Advanced Lectures*** (1st Edition), Lecture Notes in Computer Science, Springer, 2003.

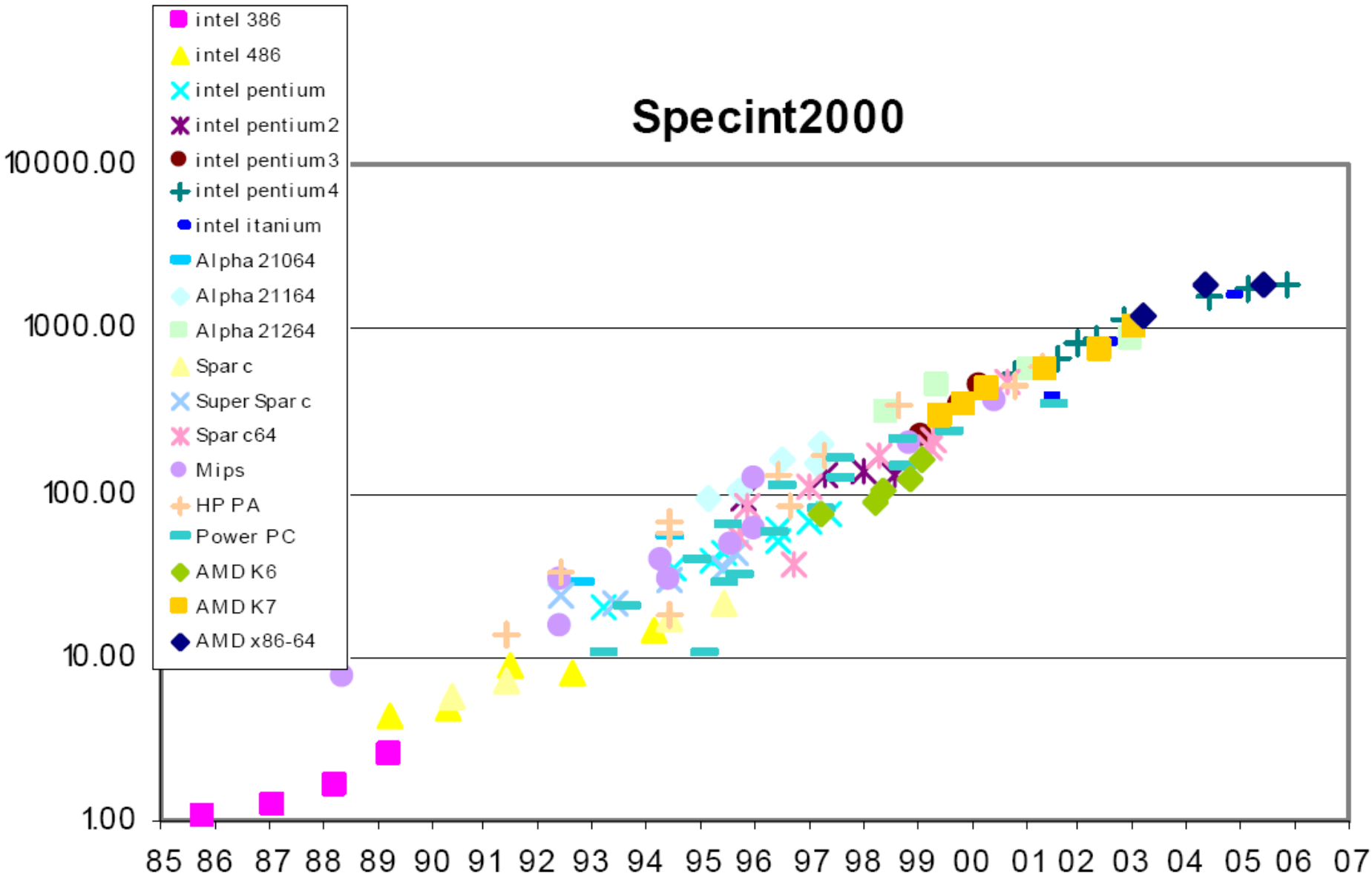
Why Parallelism?

Moore's Law



Source: Wikipedia

Unicore Performance



Source: Chung-Ta King, Department of Computer Science, National Tsing Hua University

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."

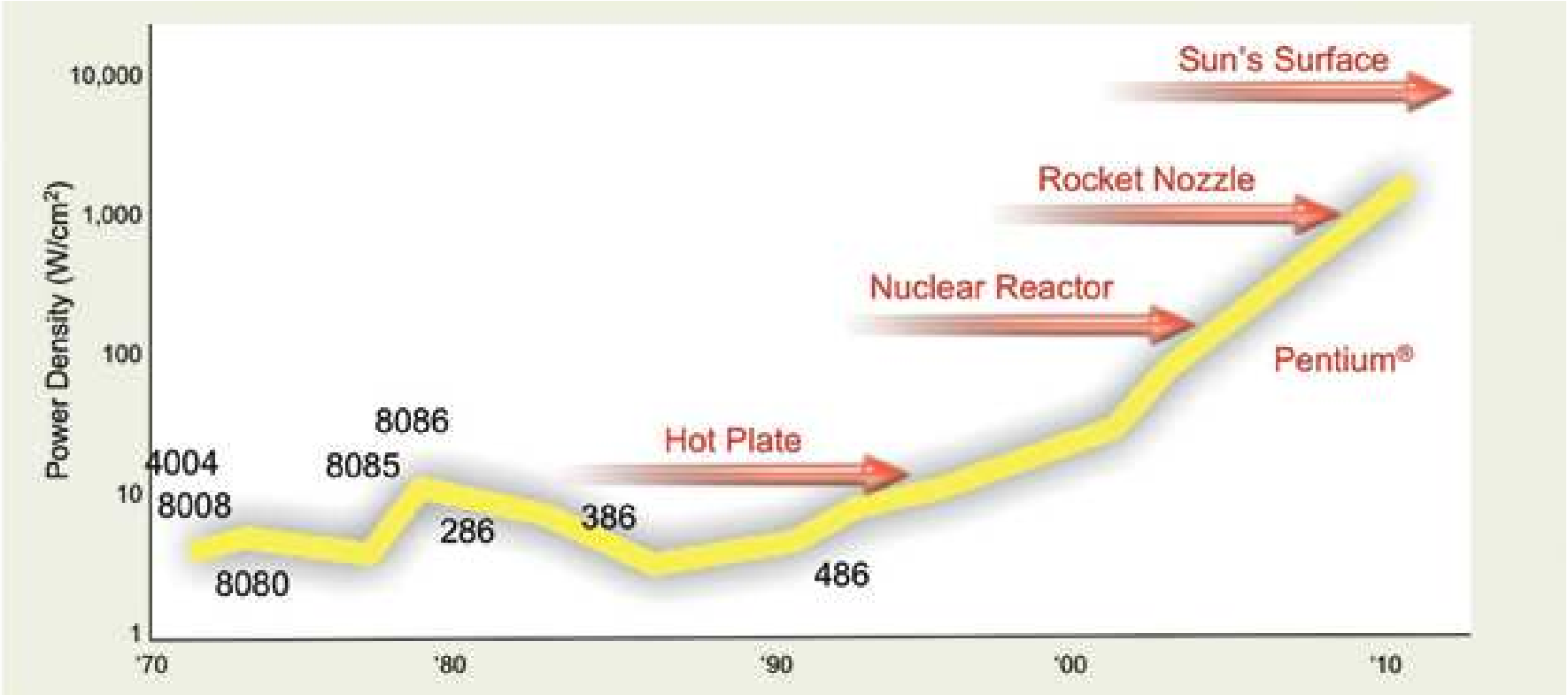
— Charles H. Duell
Commissioner, U.S. patent office, 1899

Exhausted all ideas to exploit hidden parallelism?

- Multiple simultaneous instructions
- Dynamic instruction scheduling
- Branch prediction
- Out-of-order instructions
- Speculative execution
- Pipelining
- Non-blocking caches, etc.

Unicore Performance: High Power Density

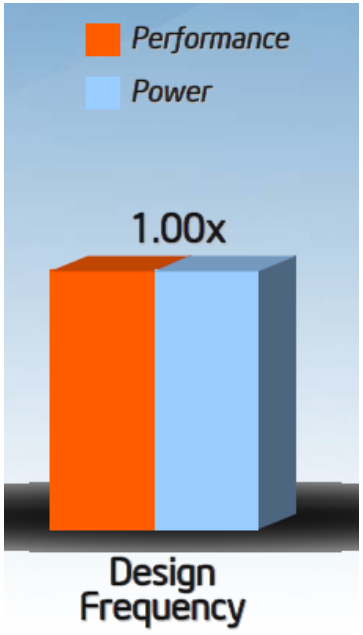
- Dynamic power, $P_d \propto V^2 f C$
  - *V = supply voltage*
  - *f = clock frequency*
  - *C = capacitance*
- But $V \propto f$
- Thus $P_d \propto f^3$



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

Unicore Performance: High Power Density

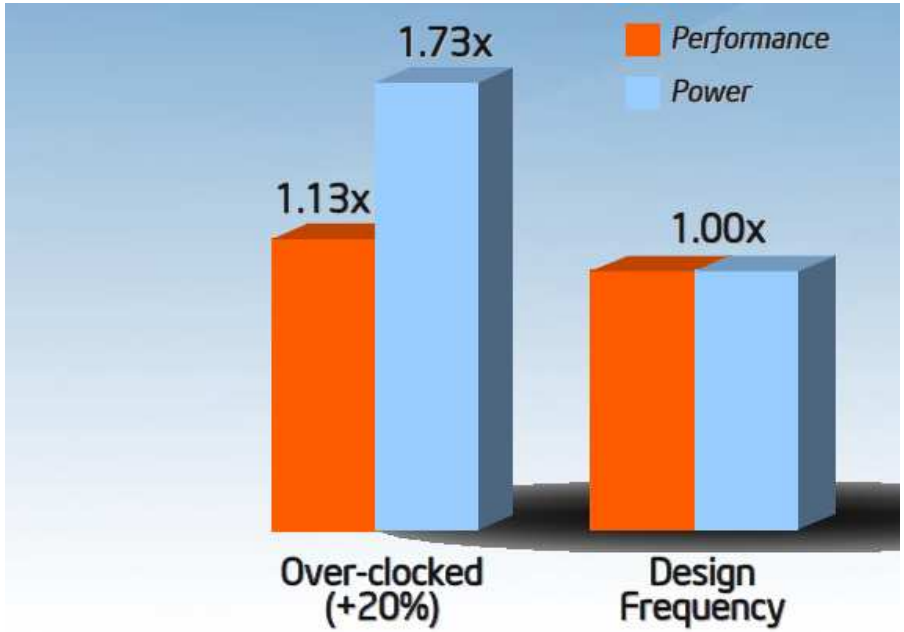
- Changing f by 20% changes performance by 13%
- So what happens if we overclock by 20%?
- And underclock by 20%?



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

Unicore Performance: High Power Density

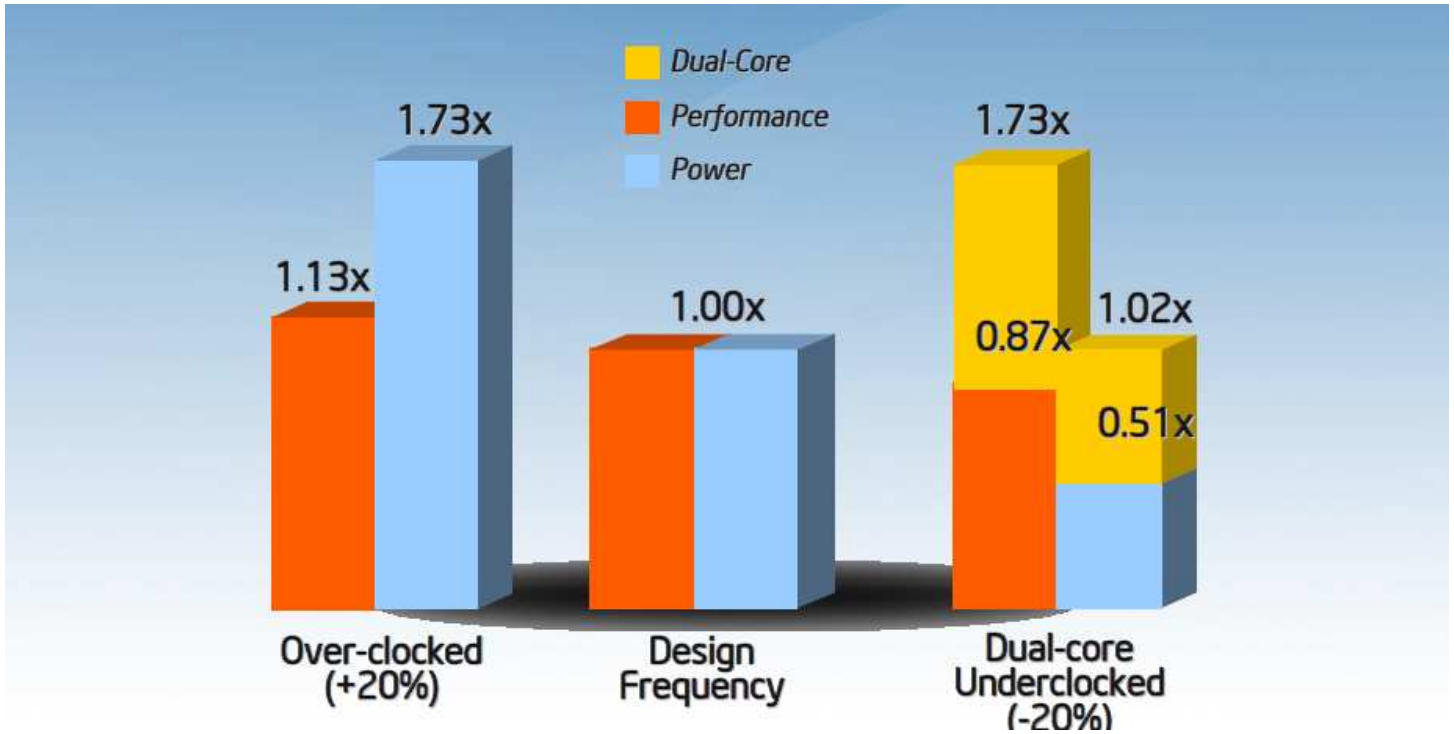
- Changing f by 20% changes performance by 13%
- So what happens if we overclock by 20%?
- And underclock by 20%?



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

Unicore Performance: High Power Density

- 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

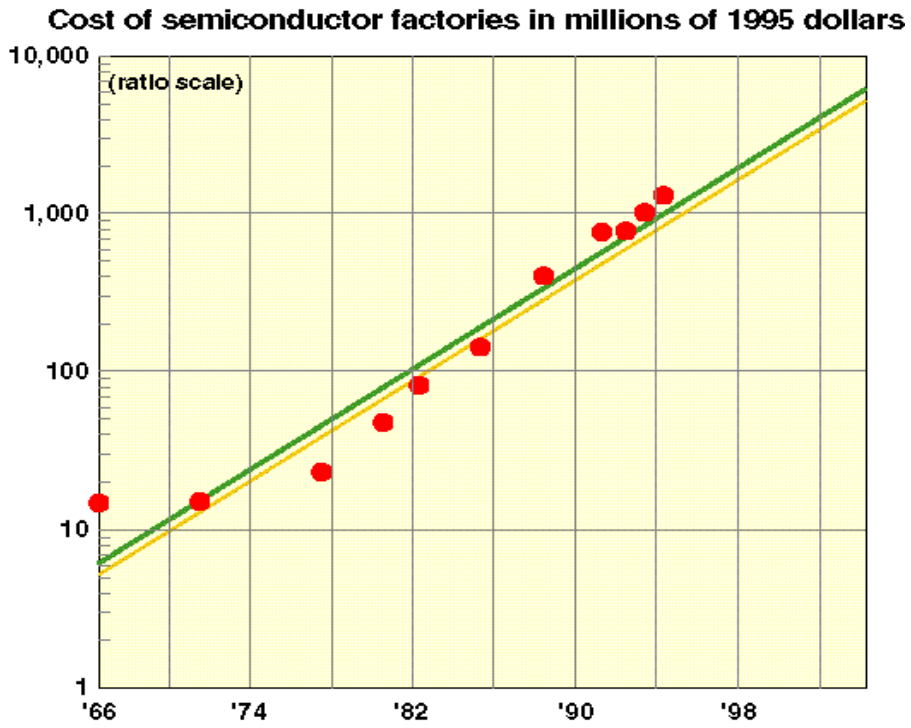
Unicore Performance: Manufacturing Issues

- 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:

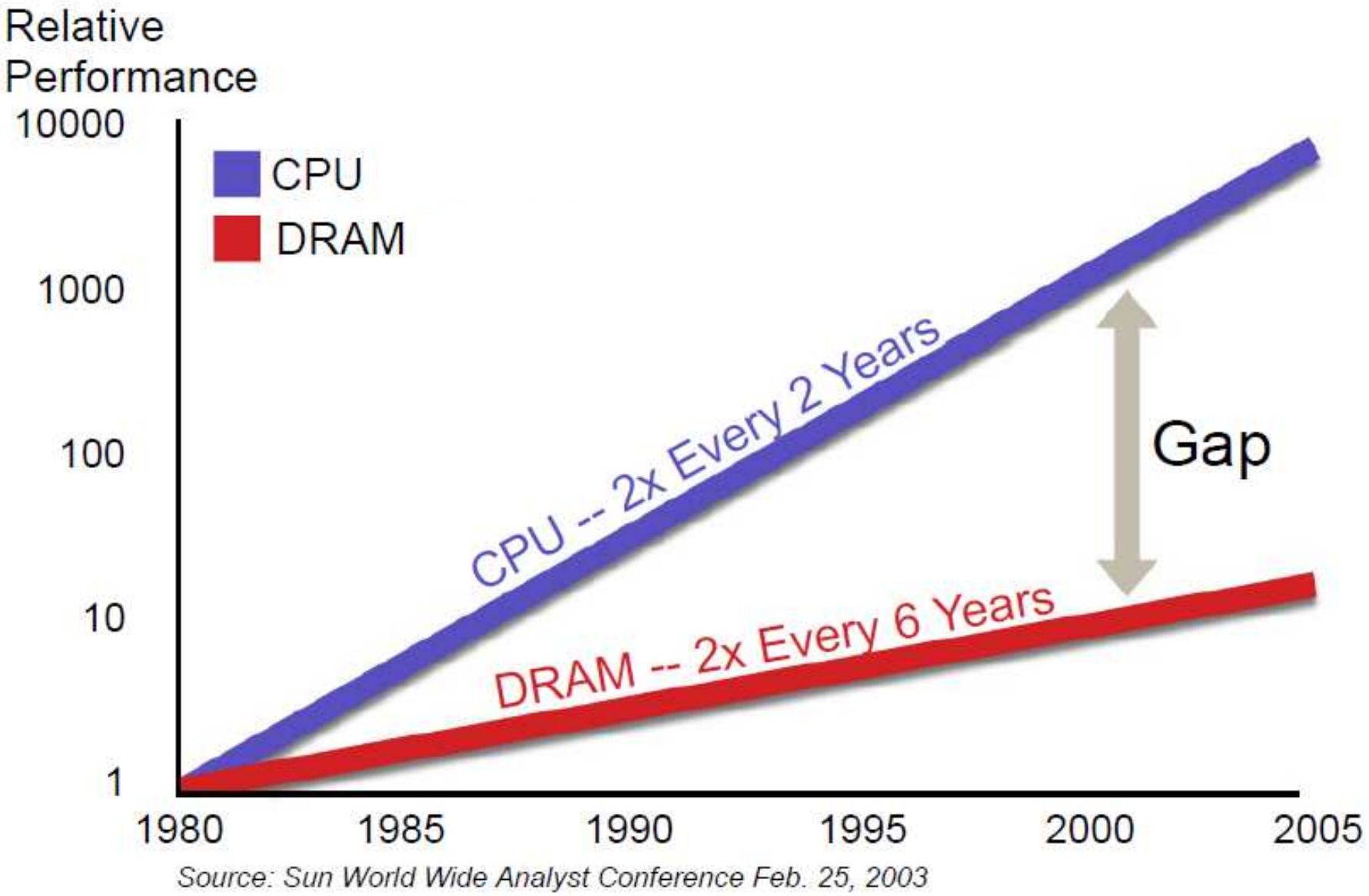
$$for\ (\ i = 0;\ i < 10^{12};\ ++i\ )$$
$$z[\ i\ ] = x[\ i\ ] + y[\ i\ ];$$

- We will have to access 3×10^{12} data items in one second
- Speed of light is, $c \approx 3\times10^{8}$ m/s
- So each data item must be within c / $3\times10^{12} \approx 0.1$ 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!)

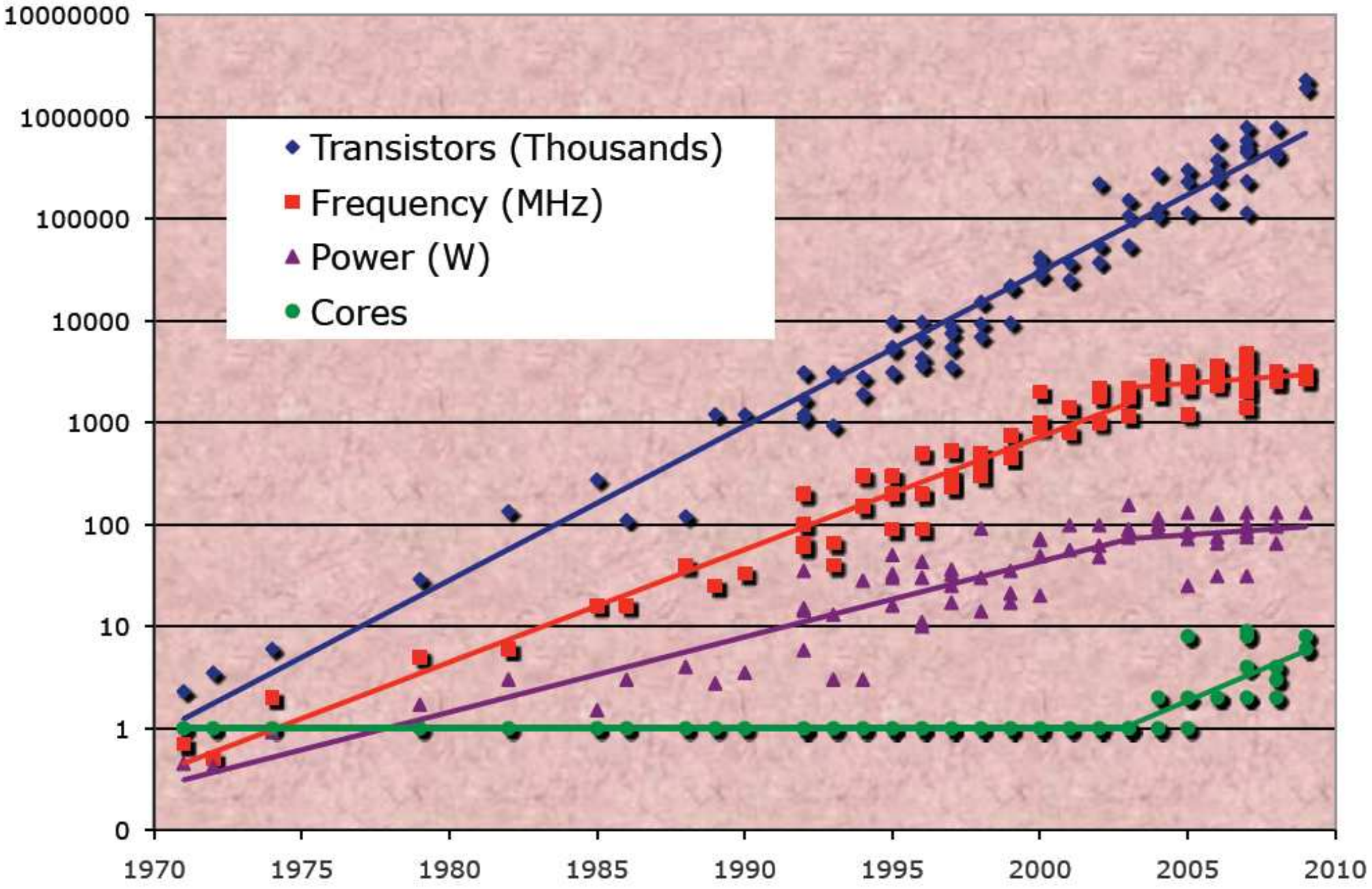
Source: Kathy Yelick and Jim Demmel, UC Berkeley

Unicore Performance: Memory Wall



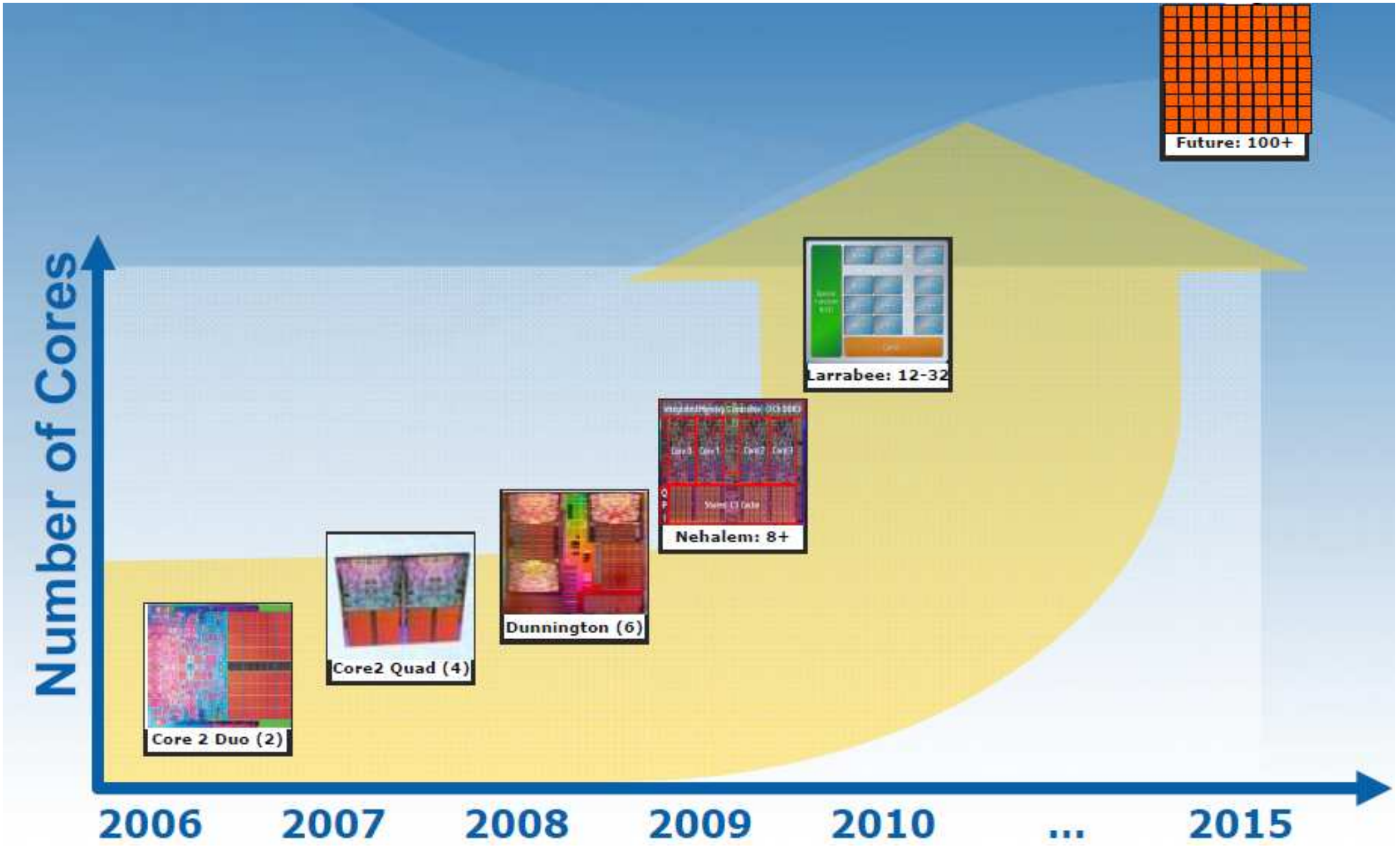
Source: Rick Hetherington, Chief Technology Officer, Microelectronics, Sun Microsystems

Moore's Law Reinterpreted



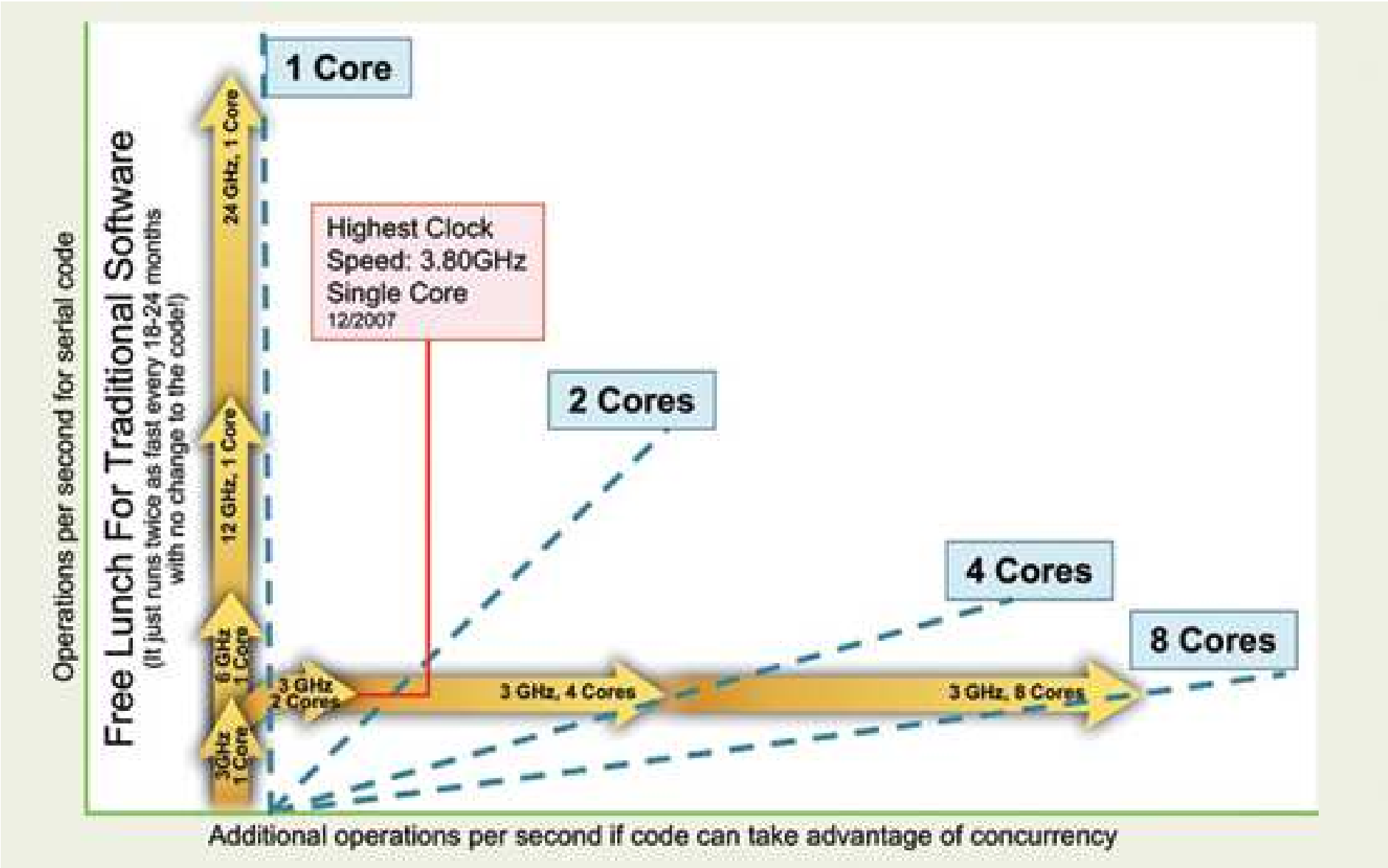
Source: Report of the 2011 Workshop on Exascale Programming Challenges

Cores / Processor (General Purpose)



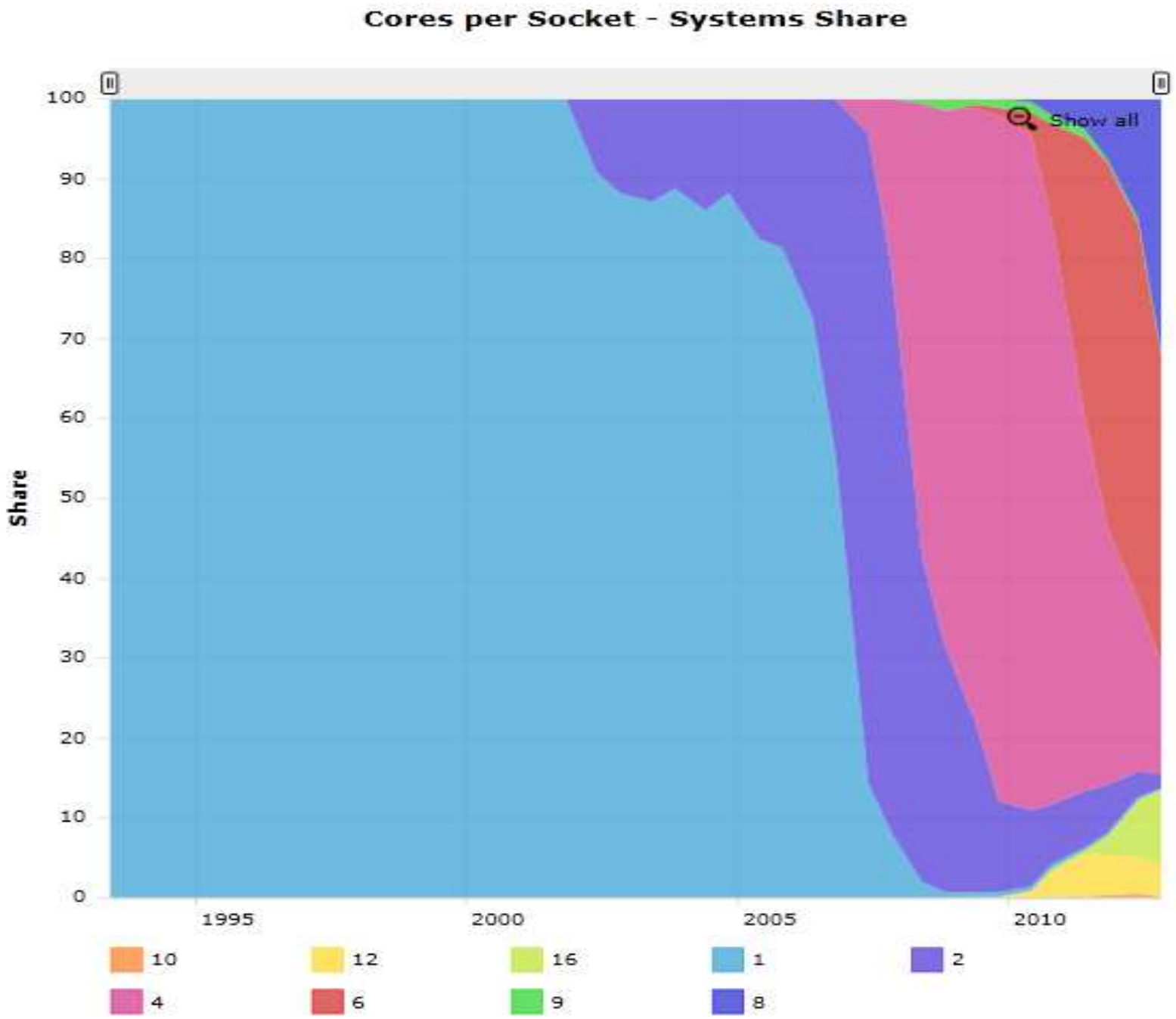
Source: Andrew A. Chien, Vice President of Research, Intel Corporation

No Free Lunch for Traditional Software



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

Top 500 Supercomputing Sites



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

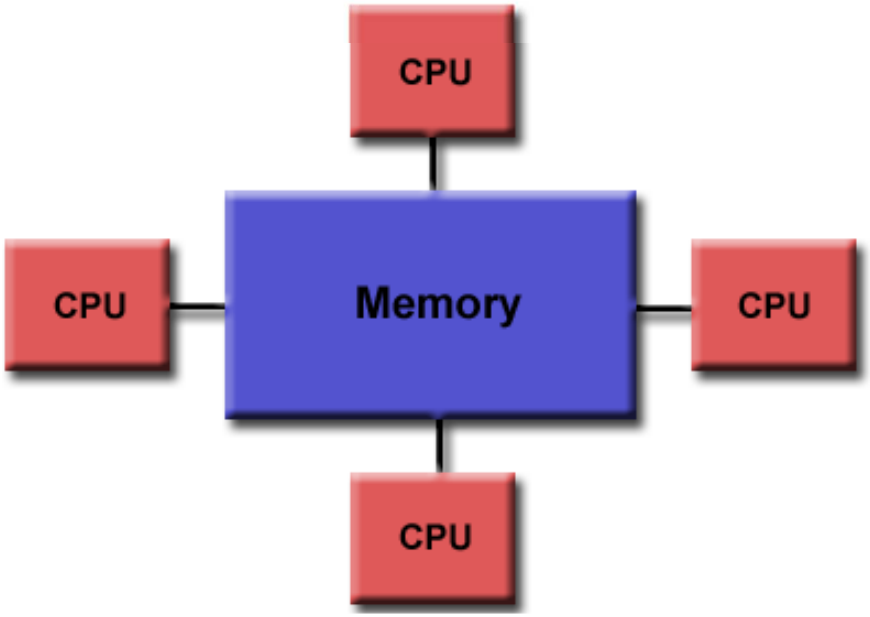
  $\approx 10^4$ time-steps to simulate
- 200 floating point operations (flop) / cell / time-step

  $\approx 4 \times 10^{15}$ floating point operations in total
- To predict in 1 hour ≈ 1 Tflop/s (Tera flop / sec)

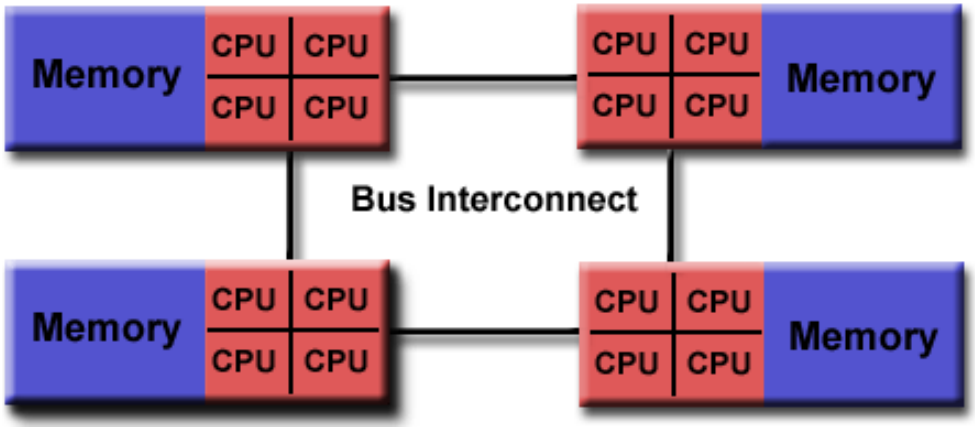
Some Useful Classifications of Parallel Computers

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
- Tow types:
  - Uniform Memory Access (UMA)
  - Non-Uniform Memory Access (NUMA)



UMA



NUMA

Source: Blaise Barney, LLNL

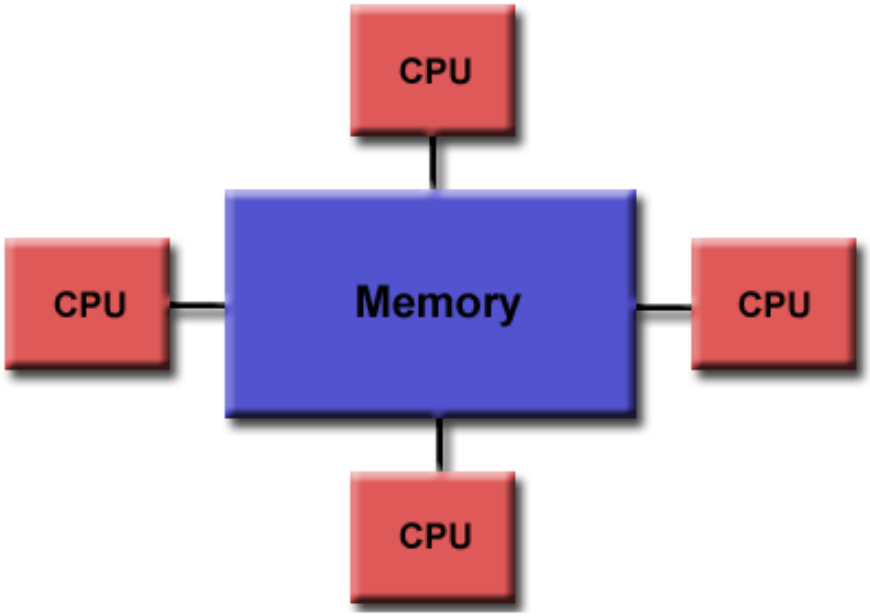
Parallel Computer Memory Architecture (Shared Memory)

Advantages

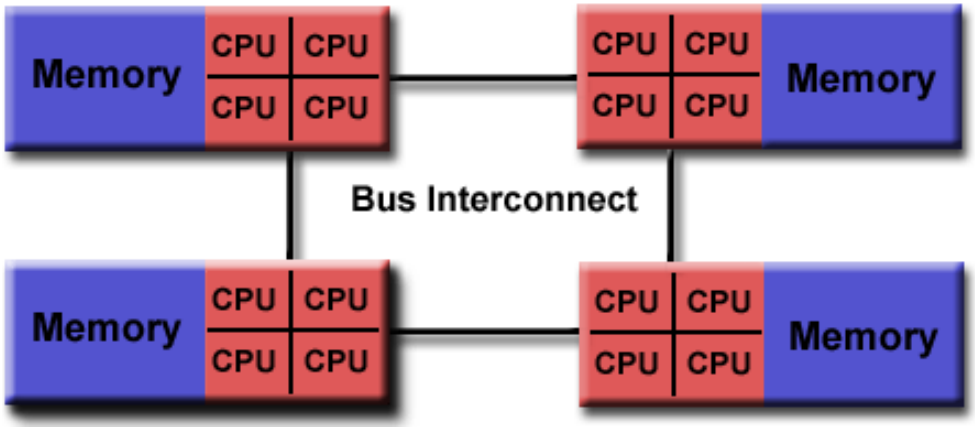
- User-friendly programming perspective to memory
- Fast data sharing

Disadvantages

- Difficult and expensive to scale
- Correct data access is user responsibility



UMA

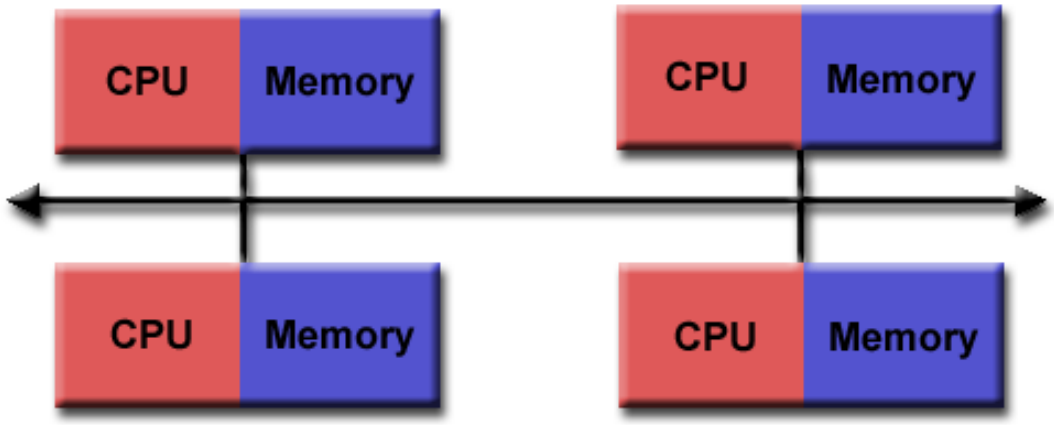


NUMA

Source: Blaise Barney, LLNL

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



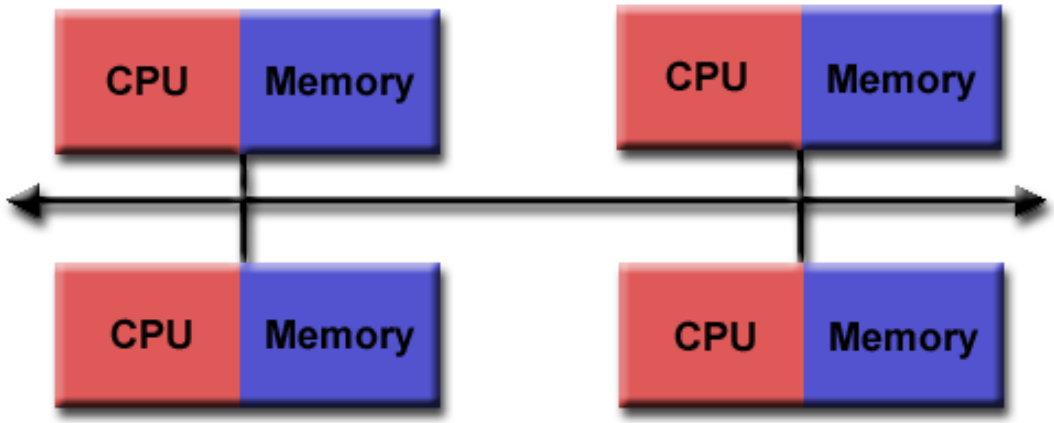
Source: Blaise Barney, LLNL

- Communication network to connect inter-processor memory

Parallel Computer Memory Architecture (Distributed Memory)

Advantages

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



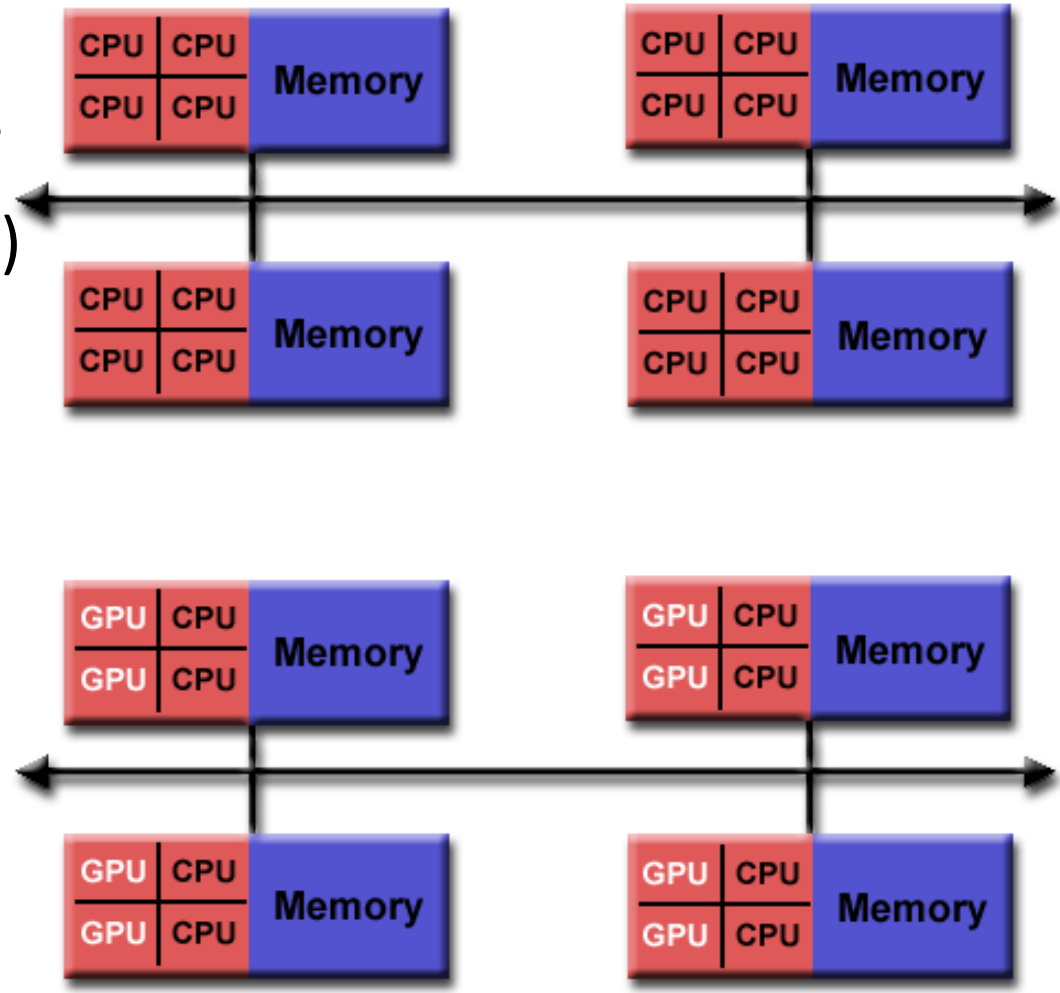
Source: Blaise Barney, LLNL

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 (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



Source: Blaise Barney, LLNL

Flynn's Taxonomy of Parallel Computers

Flynn's classical taxonomy (1966):

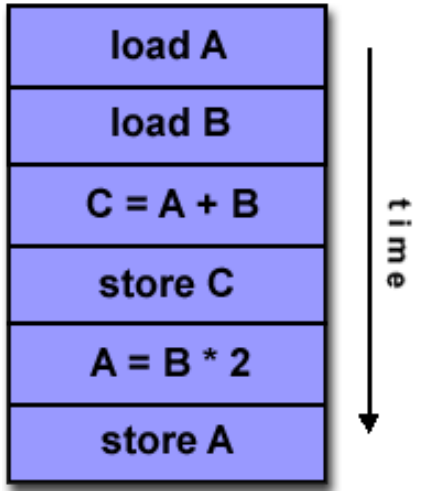
Classification of multi-processor computer architectures along two independent dimensions of *instruction* and *data*.

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

Flynn's Taxonomy of Parallel Computers

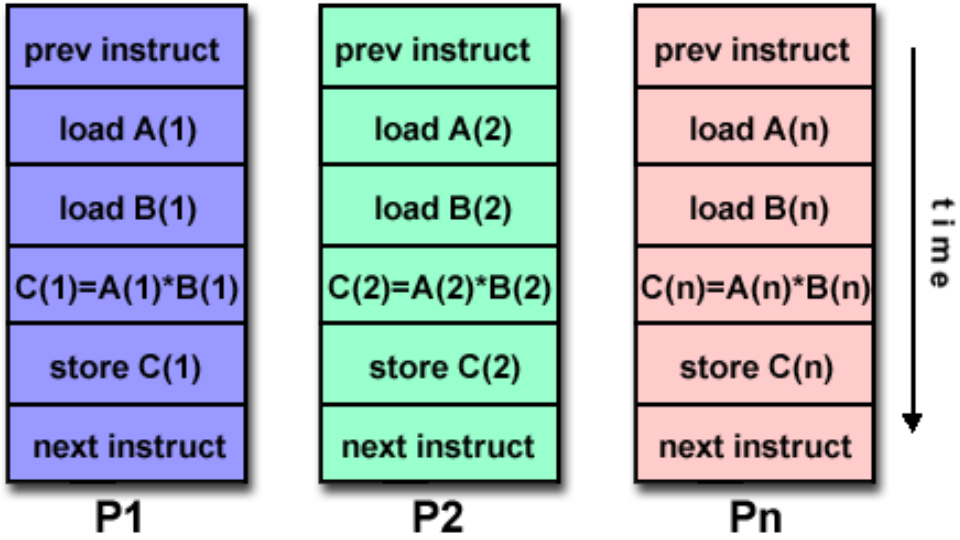
SISD

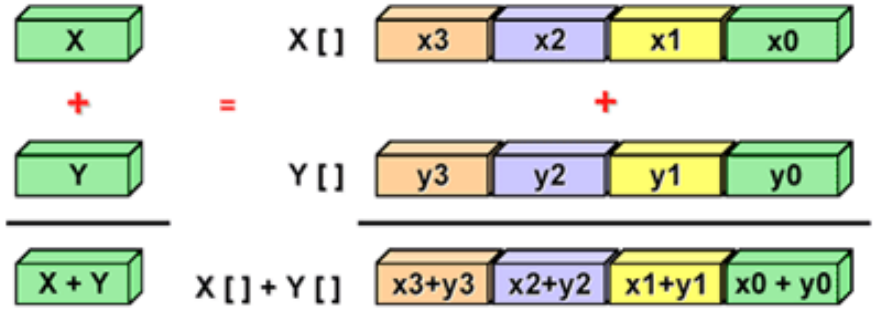
- A serial (non-parallel) computer
- The oldest and the most common type of computers
- Example: Uniprocessor unicore machines



Source: Blaise Barney, LLNL

Flynn's Taxonomy of Parallel Computers





Source: Blaise Barney, LLNL

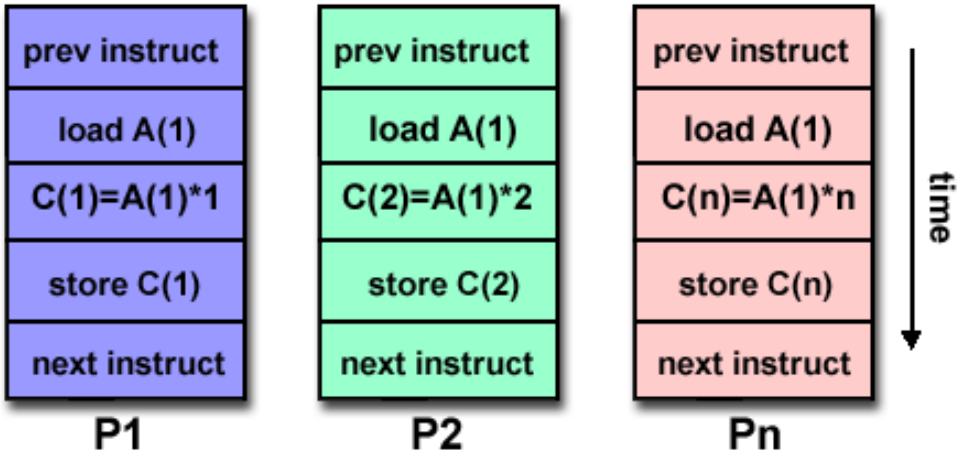
SIMD

- 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

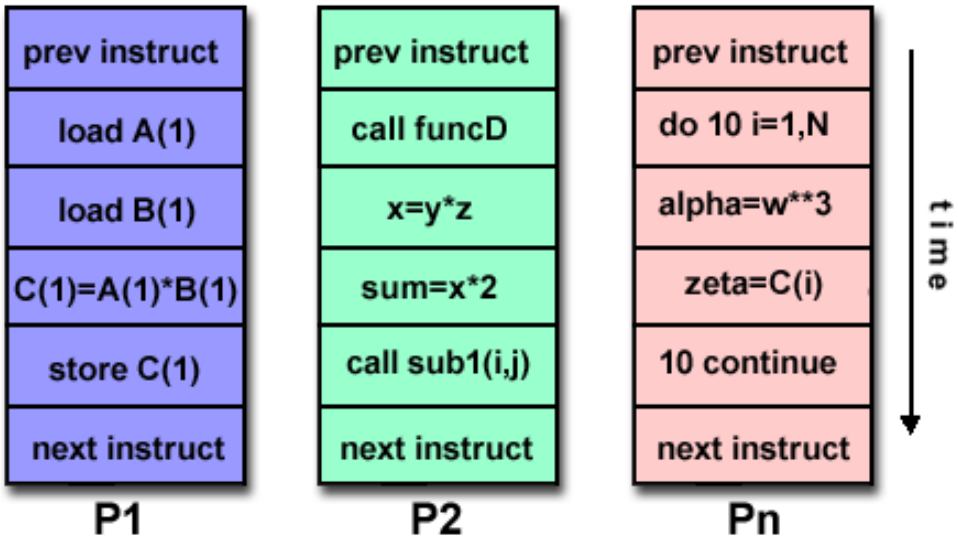
MISD

- 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:

for i = 1 *to n 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:

for $i = 1$ *to* n *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:

$c \leftarrow 0$

for $i = 1$ *to* n *do*

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

– How long will this loop take to complete execution?

Parallel Algorithms Warm-up (2)

Now consider the following loop:

$c \leftarrow 0$

for $i = 1$ *to* n *do*

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

- How long will this loop take to complete execution?
  - $O(\log n)$ time

Parallel Algorithms Warm-up (3)

Now consider quicksort:

$QSort(\ A\ )$

if $|A| \leq 1$ *return* A

else $p \leftarrow A[\ rand(\ |A|\ )\ ]$

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?

Parallel Algorithms Warm-up (3)

Now consider quicksort:

QSort(A)

if $|A| \leq 1$ *return* A

else $p \leftarrow A[\ rand(\ |A|\ )\ ]$

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)