

CSE 590: Special Topics Course (Supercomputing)

**Department of Computer Science
Stony Brook University
Spring 2016**

“To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.”

— Edsger Dijkstra, The Humble Programmer, CACM

Course Information

- **Lecture Time:** MoWe 4:00 pm - 5:20 pm
- **Location:** Earth & Space 079, West Campus
- **Instructor:** Rezaul A. Chowdhury
- **Office Hours:** MoWe 2:00 pm - 3:30 pm, 239 New CS Building
- **Email:** rezaul@cs.stonybrook.edu

- **TA:** TBA
- **TA Office Hours:** TBA
- **TA Email:** TBA

- **Class Webpage:**
<http://www.cs.sunysb.edu/~rezaul/CSE590-S16.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 (for supercomputing)

Course Organization

- **First Part:** 11 Lectures (tentative)
 - Introduction (2)
 - Shared-memory parallelism & Cilk (2)
 - Distributed-memory parallelism & MPI (2)
 - GPGPU computation & CUDA (2)
 - MapReduce & Hadoop (2)
 - Cloud computing (1)
- **Second Part:**
 - Paper presentations
 - Group projects

Grading Policy

- Programming assignments (best 3 of 4): 15%
- Paper presentation (one): 25%
- Report on a paper presented by another student (one): 10%
- Group project (one): 40%
 - Proposal (in-class): Feb 29
 - Progress report (in-class): April 11
 - Final presentation (in-class): May 4 - 6
- Class participation & attendance: 10%

Programming Environment

This course is supported by an educational grant from

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

We will use XSEDE for homeworks/projects involving

- Shared-memory parallelism
- Distributed-memory parallelism
- GPGPU and Intel MIC

Programming Environment

On XSEDE we have access to

- Stampede: 6,400 nodes with 16 cores/node & Intel MIC / NVIDIA GPU
- Comet: \approx 2,000 compute nodes with 24 cores/node
- SuperMIC: 360 nodes with 20 cores/node & Intel MIC / NVIDIA GPU

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.	560,640	17,590.0	27,112.5	8,209
2	DOE/NNSA/LLNL United States	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	16,324.8	20,132.7	7,890
3	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
4	DOE/SC/Argonne National Laboratory United States	Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,162.4	10,066.3	3,945
5	Forschungszentrum Juelich (FZJ) Germany	JUQUEEN - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393,216	4,141.2	5,033.2	1,970
6	Leibniz Rechenzentrum Germany	SuperMUC - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM/Lenovo	147,456	2,897.0	3,185.1	3,423
7	Texas Advanced Computing Center/Univ. of Texas United States	Stampede - PowerEdge C8220, Xeon E5- 2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi Dell	204,900	2,660.3	3,959.0	

Recommended Texts

No required textbook.

Some useful ones are as follows

- A. Grama, G. Karypis, V. Kumar, and A. Gupta. ***Introduction to Parallel Computing*** (2nd Edition), Addison Wesley, 2003.
- 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.
- D. and W. Hwu. ***Programming Massively Parallel Processors: A Hands-on Approach*** (1st Edition), Morgan Kaufmann, 2010.
- J. Lin and C. Dyer. ***Data-Intensive Text Processing with MapReduce***, Morgan and Claypool Publishers, 2010.
- T. White. ***Hadoop: The Definitive Guide*** (2nd Edition), Yahoo Press, 2010.
- T. Velte, A. Velte, and R. Elsenpeter. ***Cloud Computing, A Practical Approach*** (1st Edition), McGraw-Hill Osborne Media, 2009.

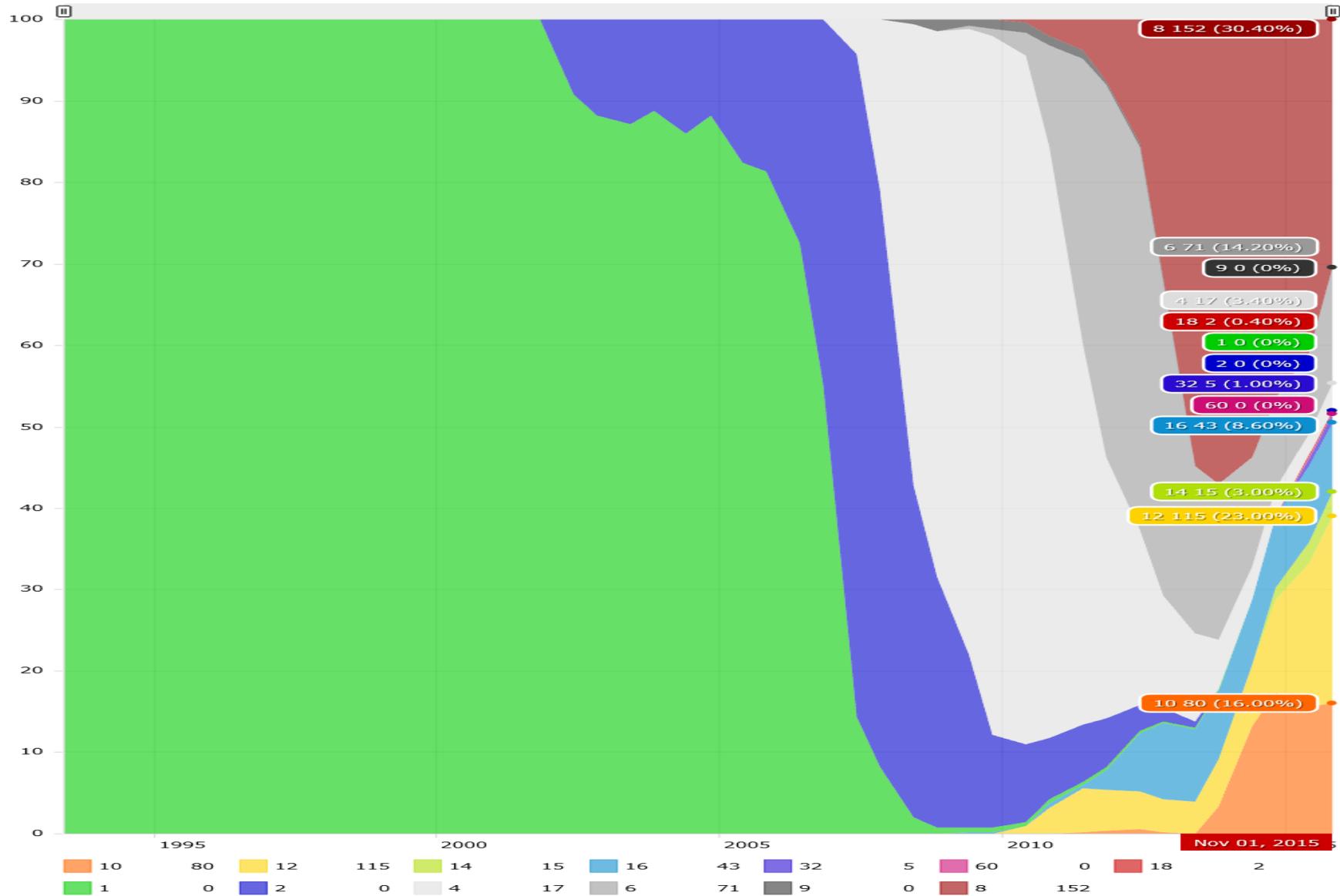
**Supercomputing
&
Parallel Computing**

Top 10 Supercomputing Sites in Nov. 2015

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	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
2	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
3	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
4	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
5	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
6	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	
7	Swiss National Supercomputing Centre (CSCS) Switzerland	Piz Daint - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325
8	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	
9	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
10	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

Source: www.top500.org

Top 500 Supercomputing Sites (Cores per Socket – Systems Share)



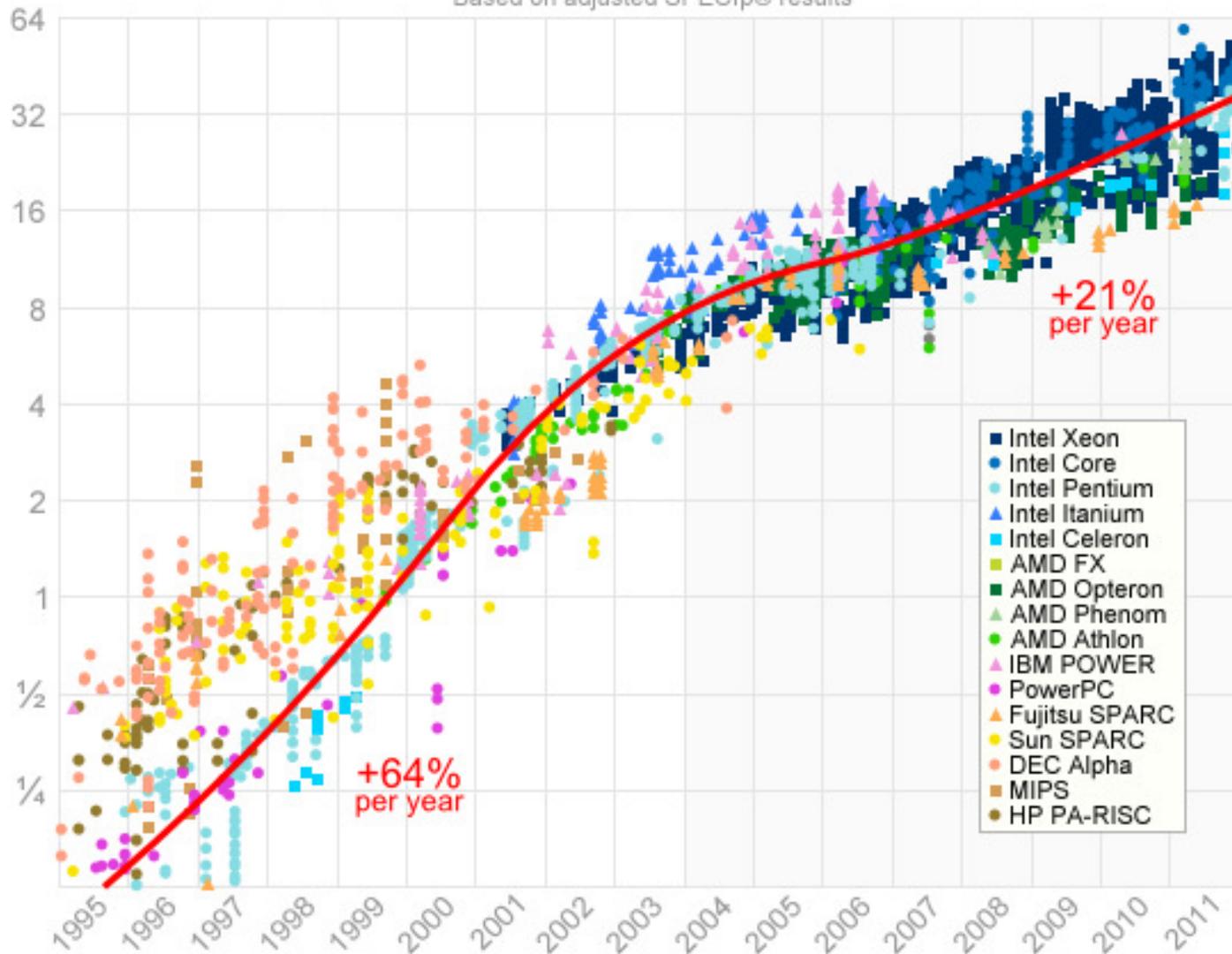
Source: www.top500.org

Why Parallelism?

Unicore Performance

Single-Threaded Floating-Point Performance

Based on adjusted SPECfp® results



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*

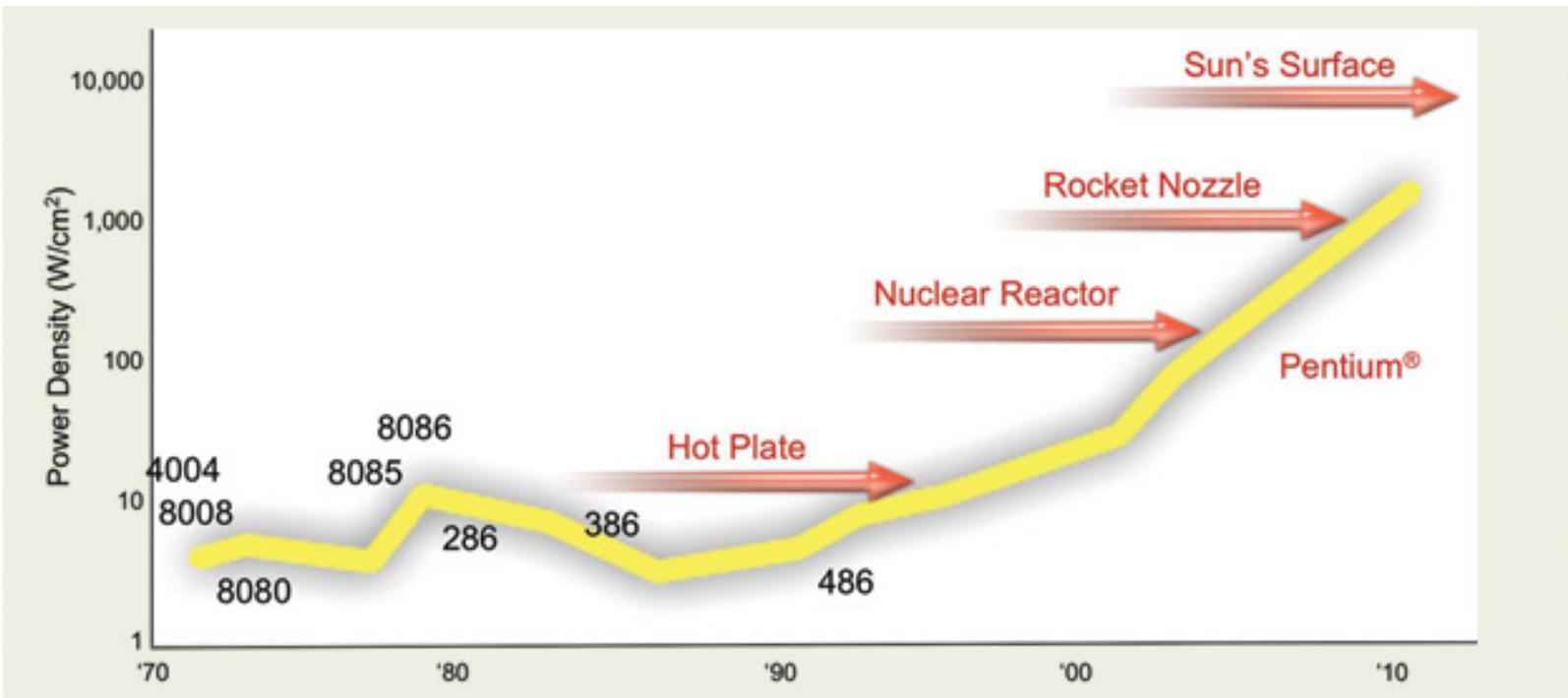
Commissioner, U.S. patent office, 1899

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, $P_d \propto V^2 f C$
 - $V = \text{supply voltage}$
 - $f = \text{clock frequency}$
 - $C = \text{capacitance}$
- But $V \propto f$
- Thus $P_d \propto f^3$



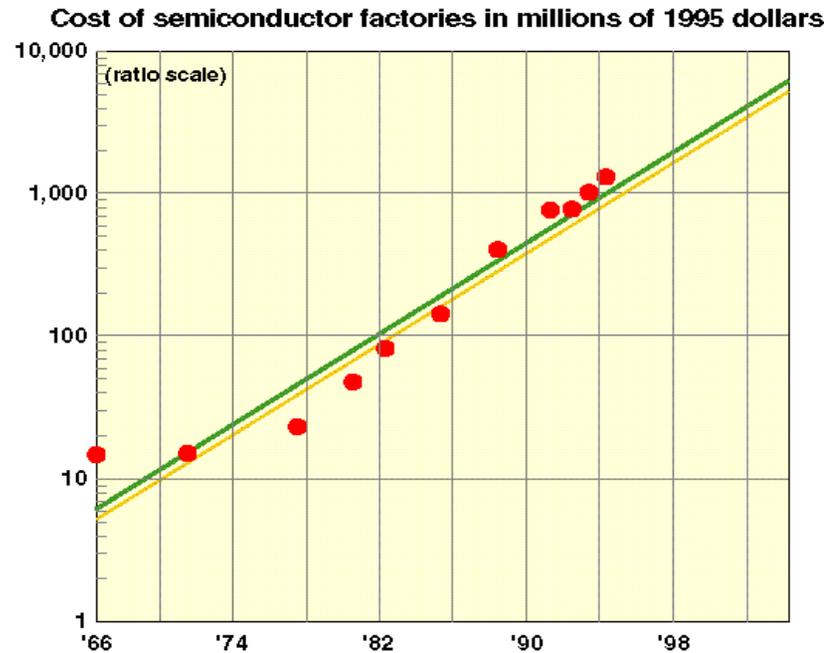
Source: Patrick Gelsinger, Intel Developer Forum, Spring 2004 (Simon Floyd)

Unicore Performance: Manufacturing Issues

- Frequency, $f \propto 1 / s$
 - $s = \text{feature size (transistor dimension)}$
- 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!

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

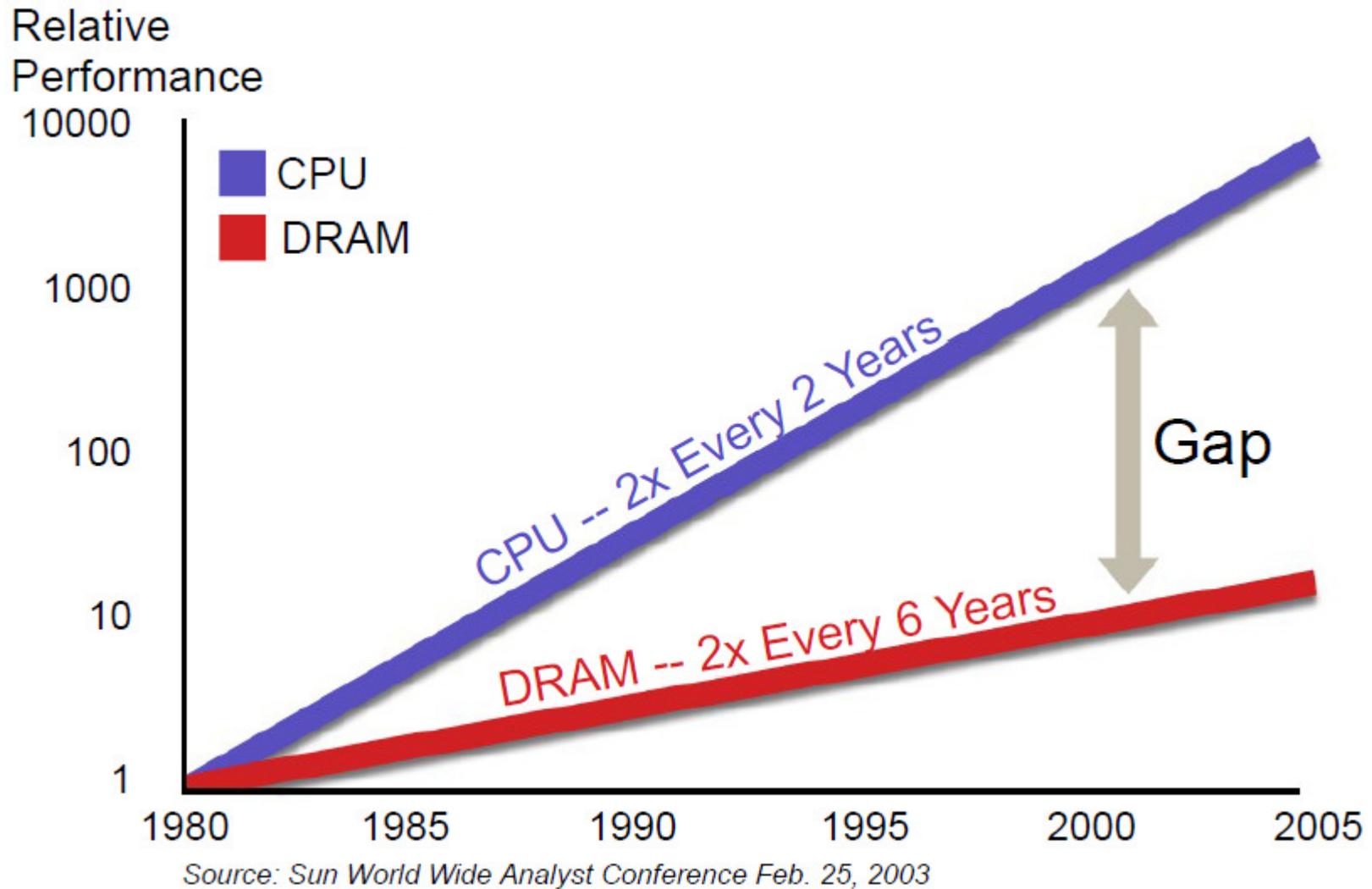
Unicore Performance: Physical Limits

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

```
for ( i = 0; i < 1012; ++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 \times 0.2 mm square
- Each data item (≥ 8 bytes) can occupy only 1 \AA^2 space!
(size of a small atom!)

Unicore Performance: Memory Wall



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

Unicore Performance Has Hit a Wall!

Some Reasons

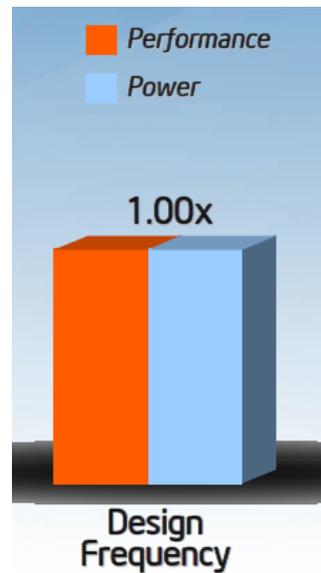
- Lack of additional ILP
(Instruction Level Hidden Parallelism)
- High power density
- Manufacturing issues
- Physical limits
- Memory speed

“Oh Sinnerman, where you gonna run to?”

— Sinnerman (recorded by Nina Simone)

Where You Gonna Run To?

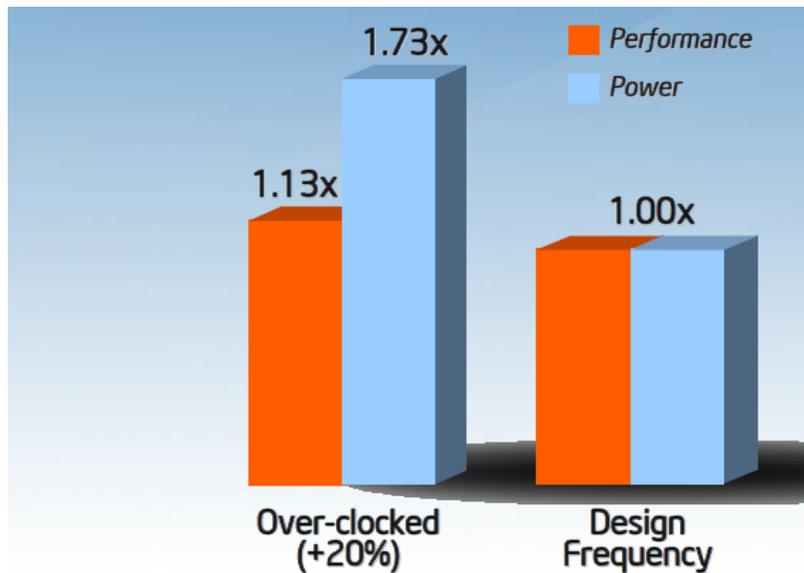
- Changing f by 20% changes performance by 13%
- So what happens if we overclock by 20%?



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

Where You Gonna Run To?

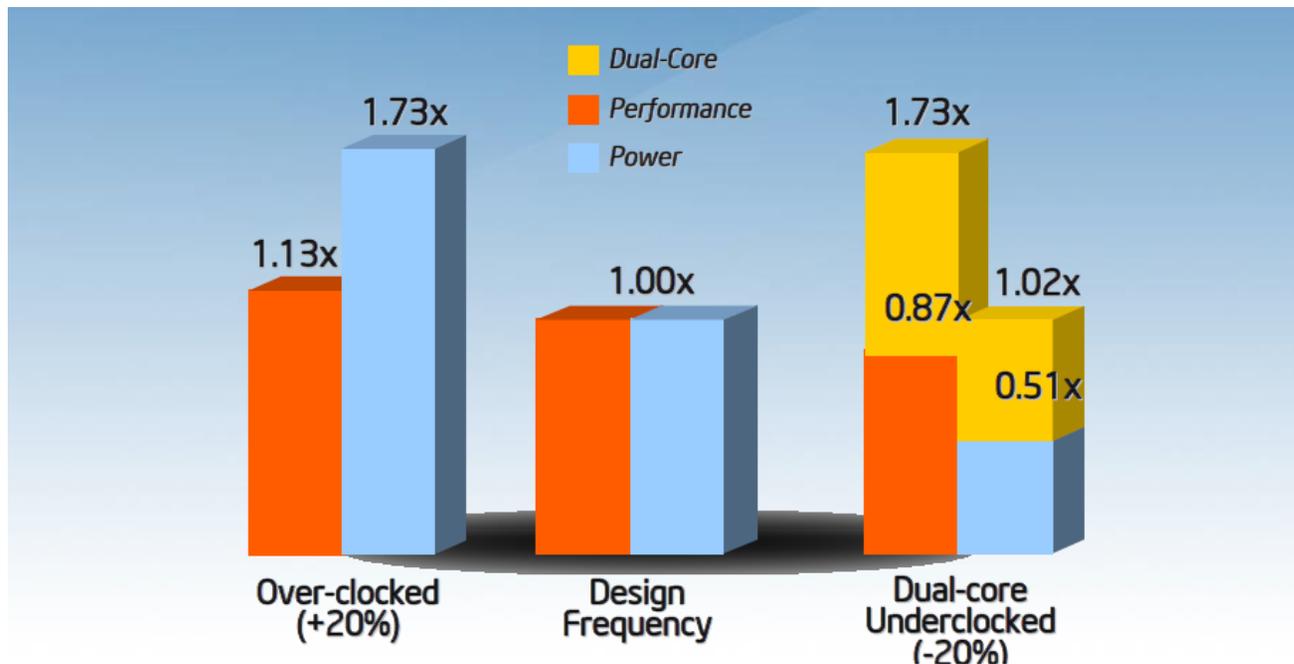
- Changing f by 20% changes performance by 13%
- So what happens if we overclock by 20%?
- And underclock by 20%?



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

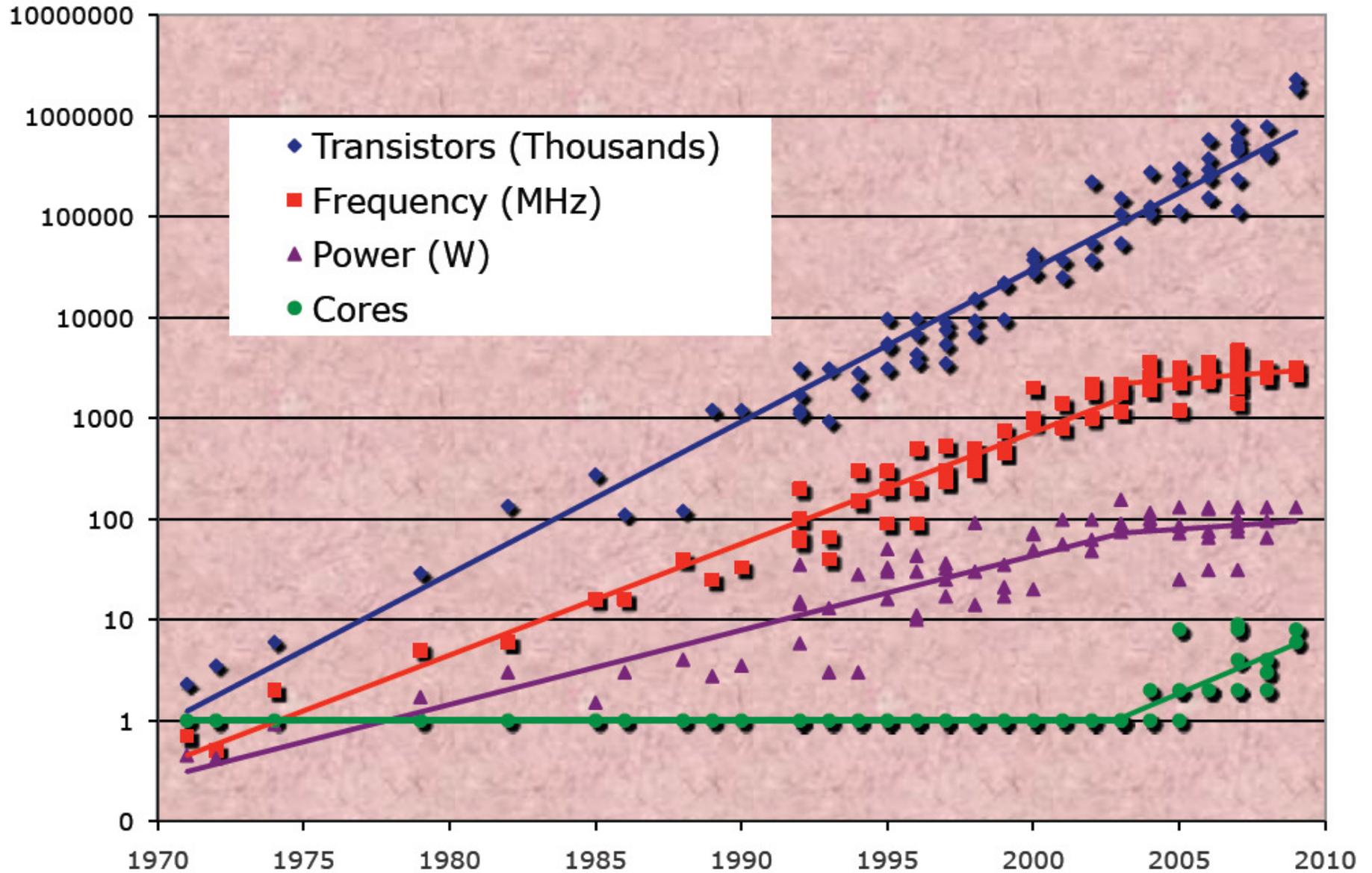
Where You Gonna Run To?

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



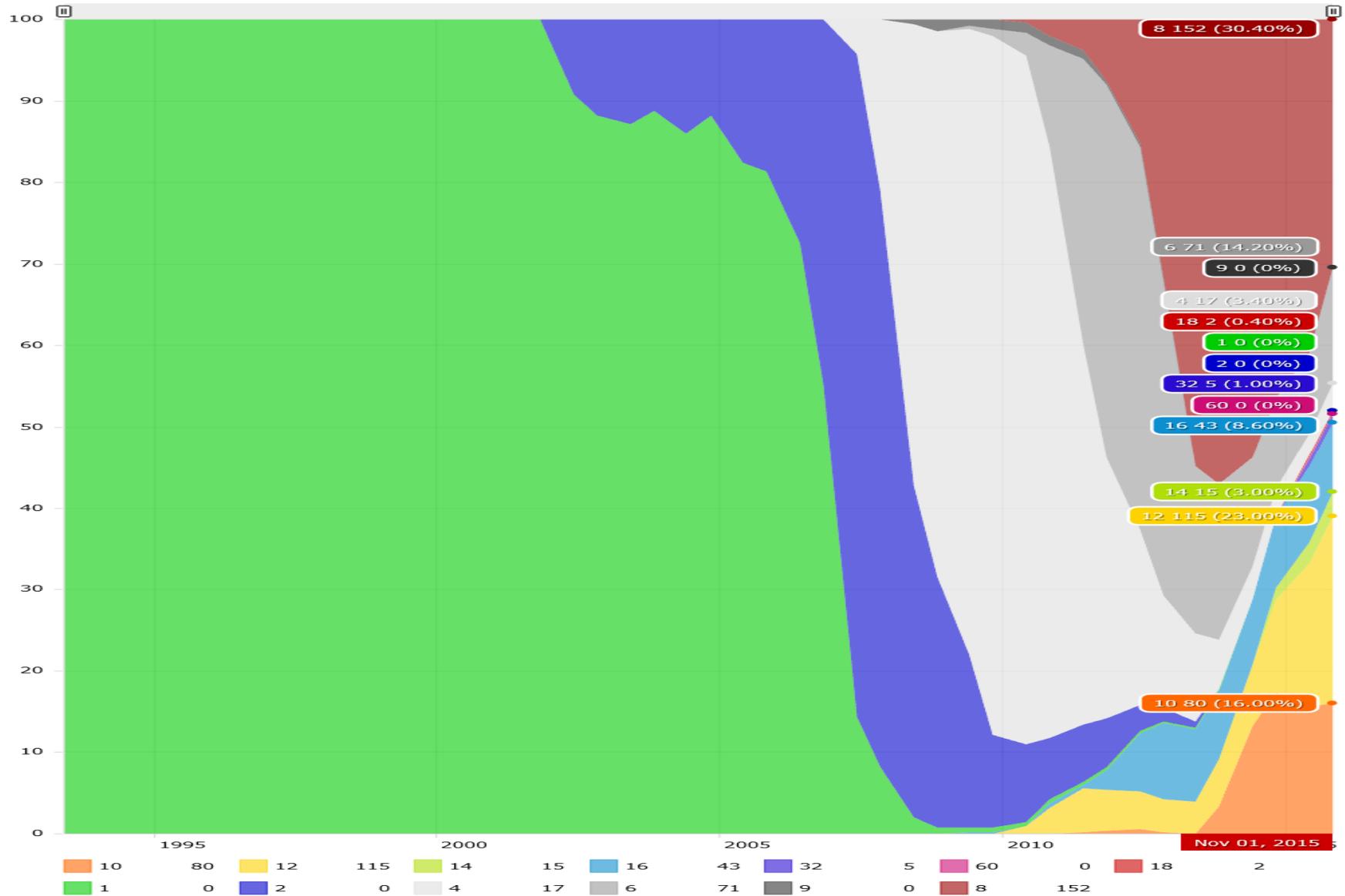
Source: Andrew A. Chien, Vice President of Research, Intel Corporation

Moore's Law Reinterpreted



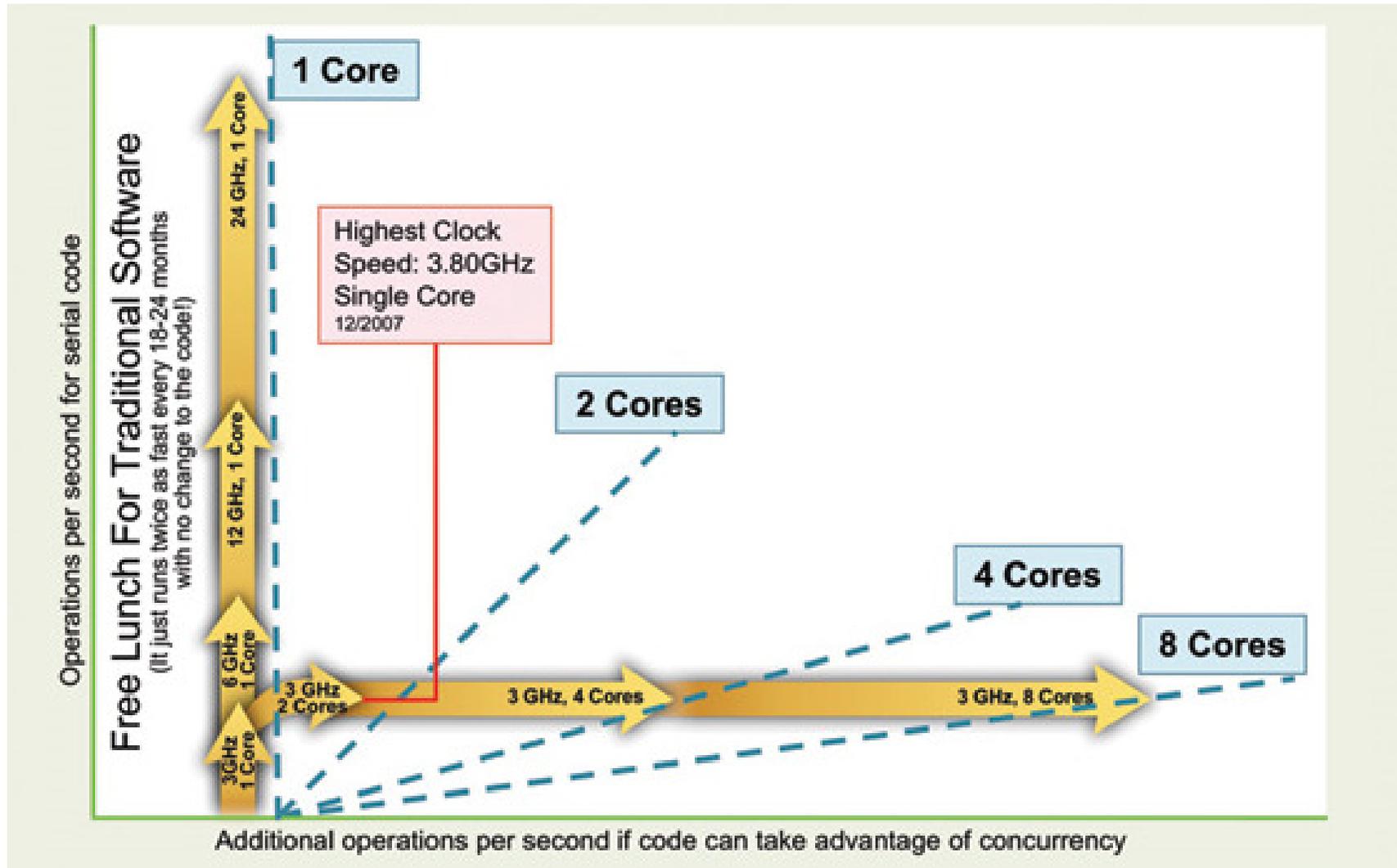
Source: Report of the 2011 Workshop on Exascale Programming Challenges

Top 500 Supercomputing Sites (Cores per Socket – Systems Share)



Source: www.top500.org

No Free Lunch for Traditional Software

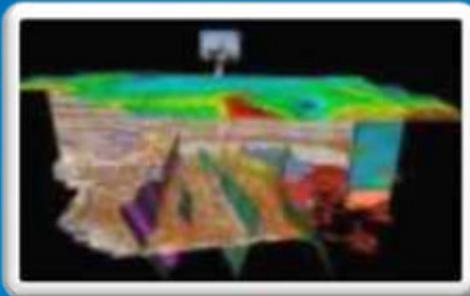


Source: Simon Floyd, Workstation Performance: Tomorrow's Possibilities (Viewpoint Column)

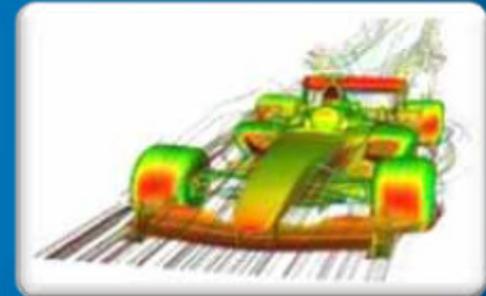
Insatiable Demand for Performance



Weather Prediction



Oil Exploration



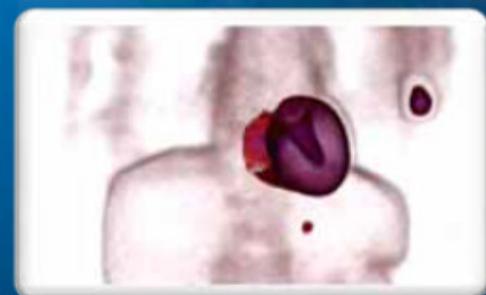
Design Simulation



Genomics Research



Financial Analysis

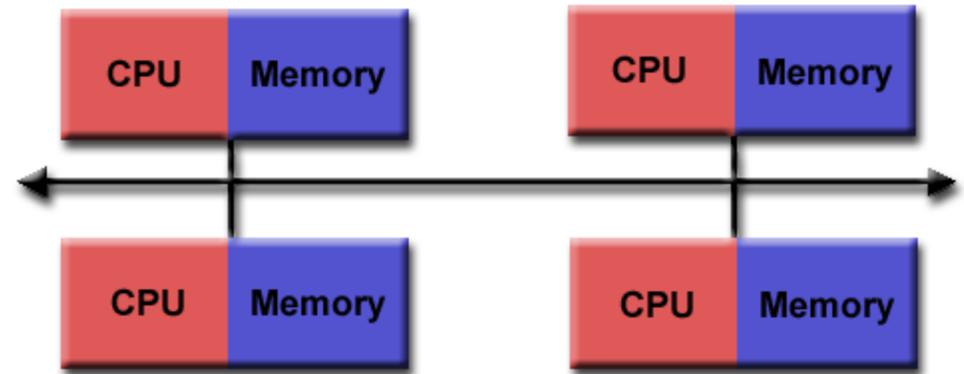


Medical Imaging

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
- Communication network to connect inter-processor memory
- Programming
 - Message Passing Interface (MPI)
 - Many once available: PVM, Chameleon, MPL, NX, etc.

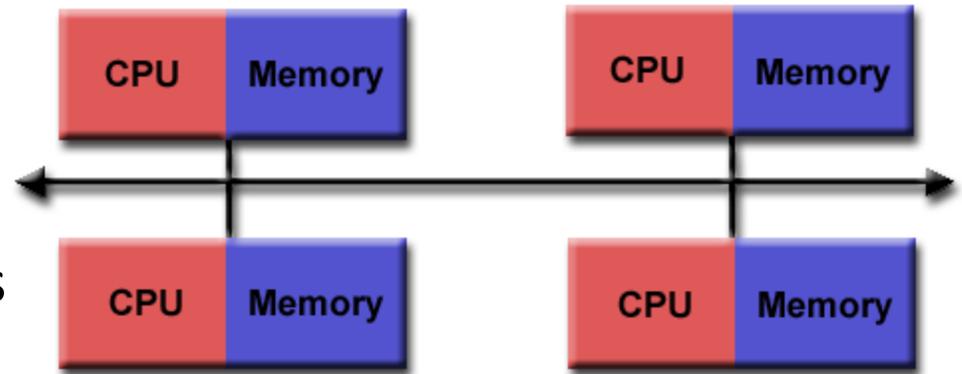


Source: Blaise Barney, LLNL

Parallel Computer Memory Architecture (Distributed Memory)

Advantages

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



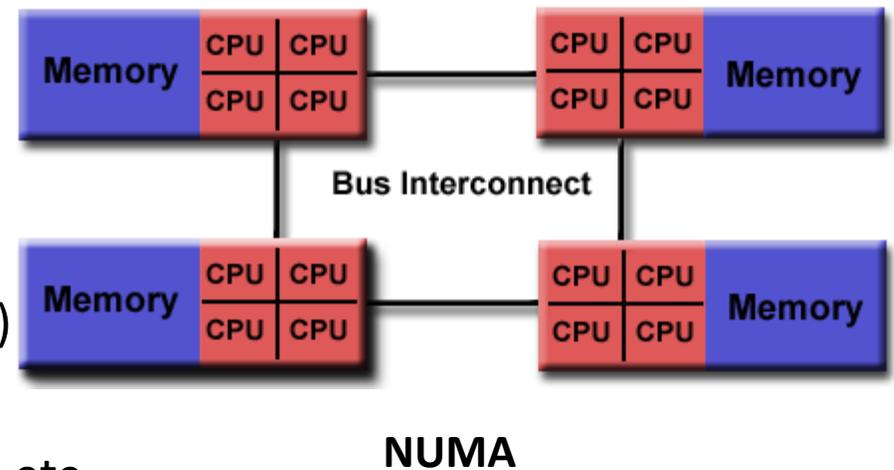
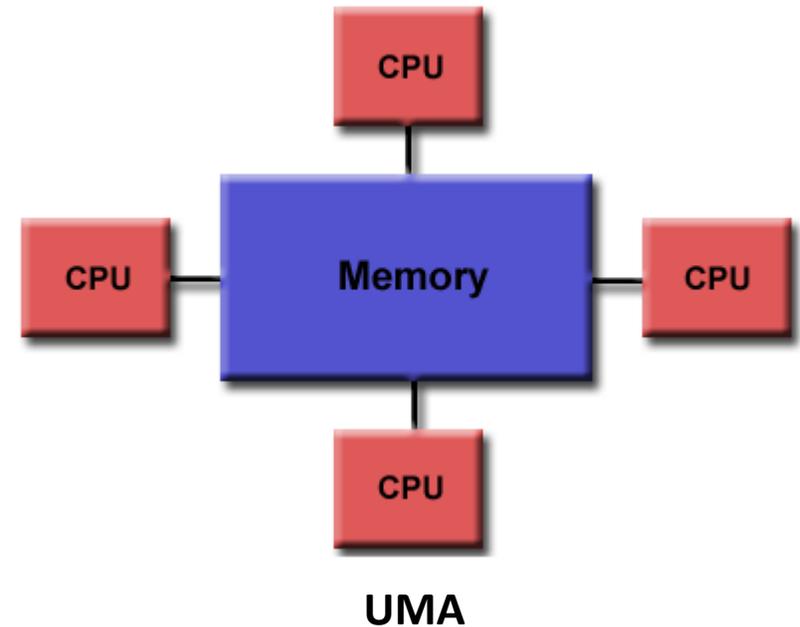
Source: Blaise Barney, LLNL

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.



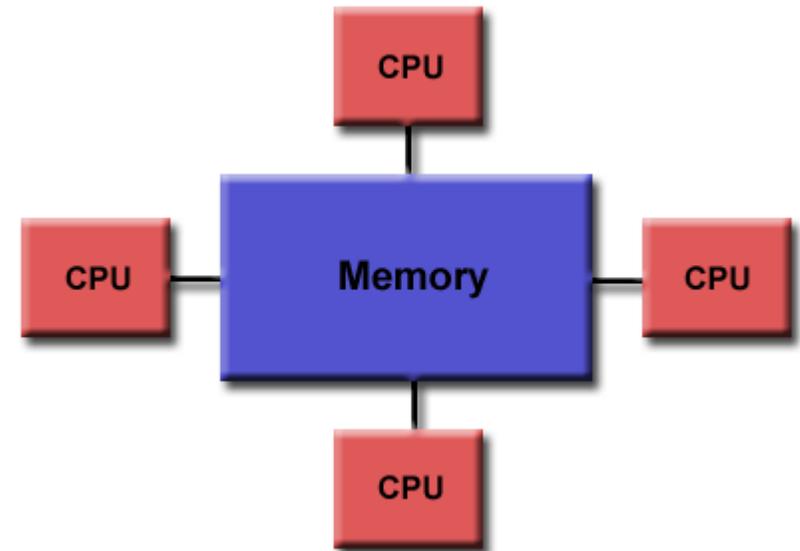
Parallel Computer Memory Architecture (Shared Memory)

Advantages

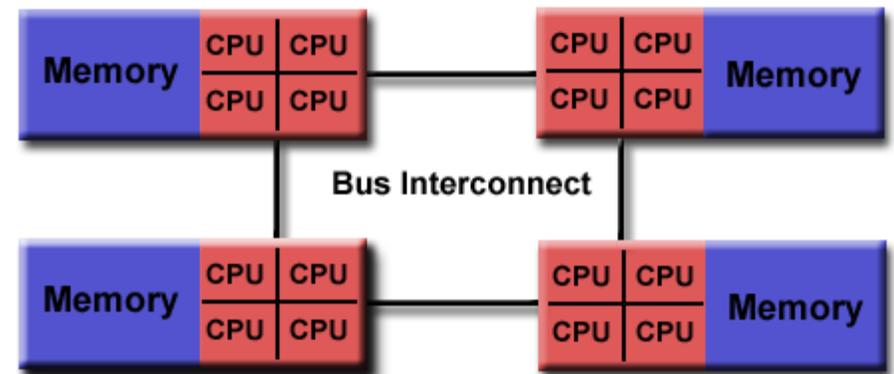
- User-friendly programming perspective to memory
- Fast data sharing

Disadvantages

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



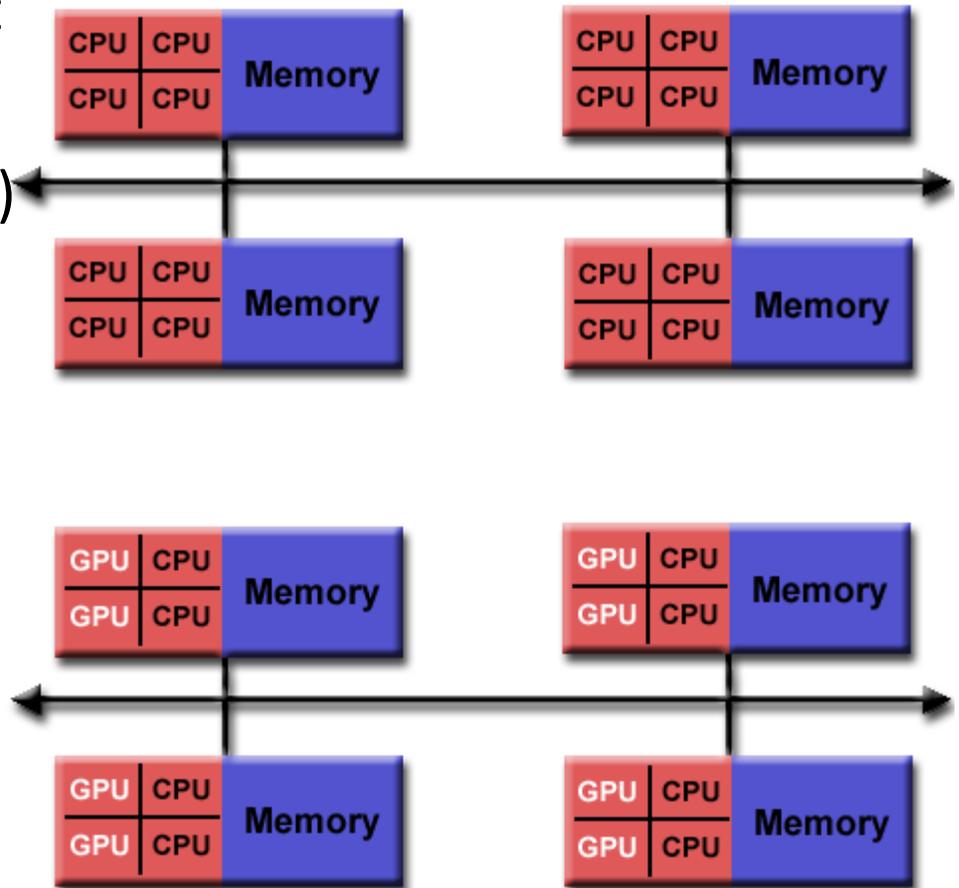
UMA



NUMA

Parallel Computer Memory Architecture (Hybrid Distributed-Shared Memory)

- The shared-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.



Flynn's Taxonomy of Parallel Computers

Flynn's classical taxonomy (1966):

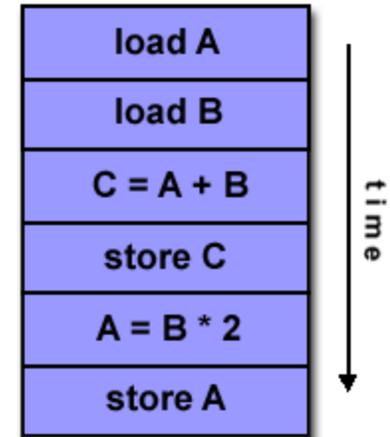
Classification of multi-processor computer architectures along two independent dimensions of *instruction* and *data*.

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

Flynn's Taxonomy of Parallel Computers

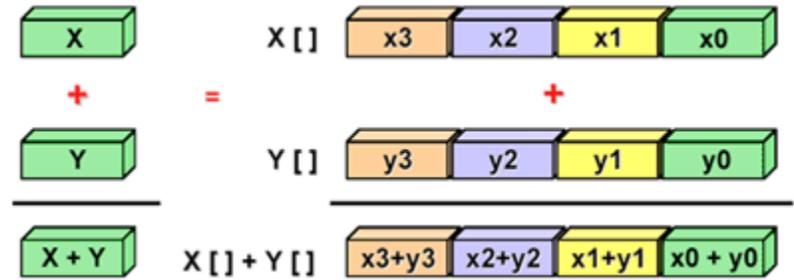
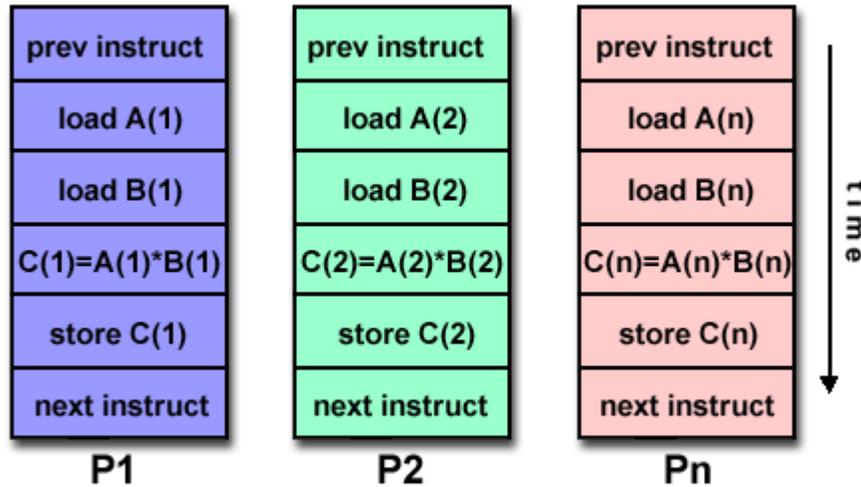
SISD

- 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

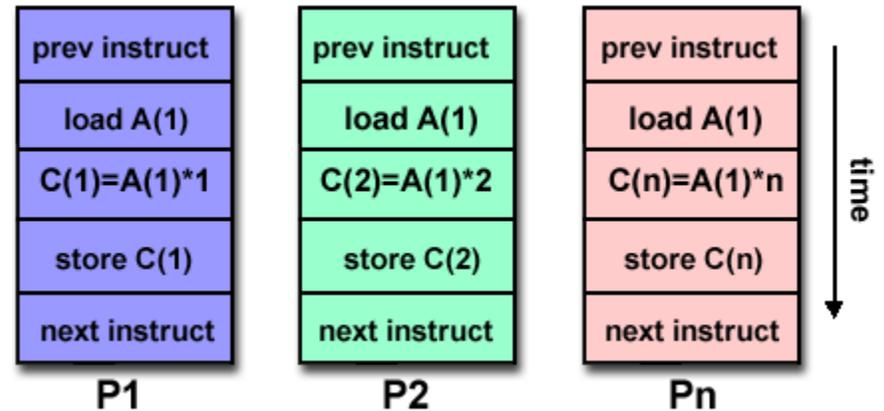
SIMD

- 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

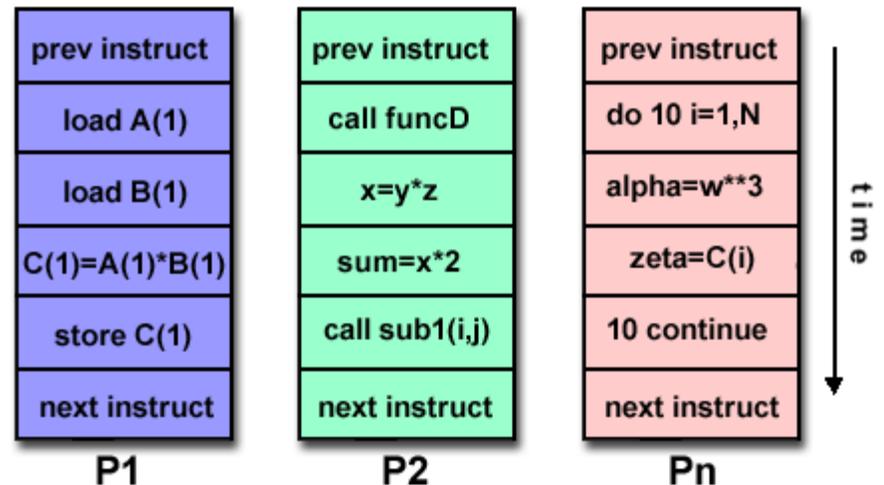
MISD

- 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



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] ← A[i] × 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] ← A[i] × 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 $p \leftarrow A[\text{rand}(|A|)]$

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

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)