Emerging computational aspects and challenges for GCMs II: Some basics of parallel computing

Dr. Richard Loft
Director, Technology Development
Computational and Information Systems Laboratory
NCAR

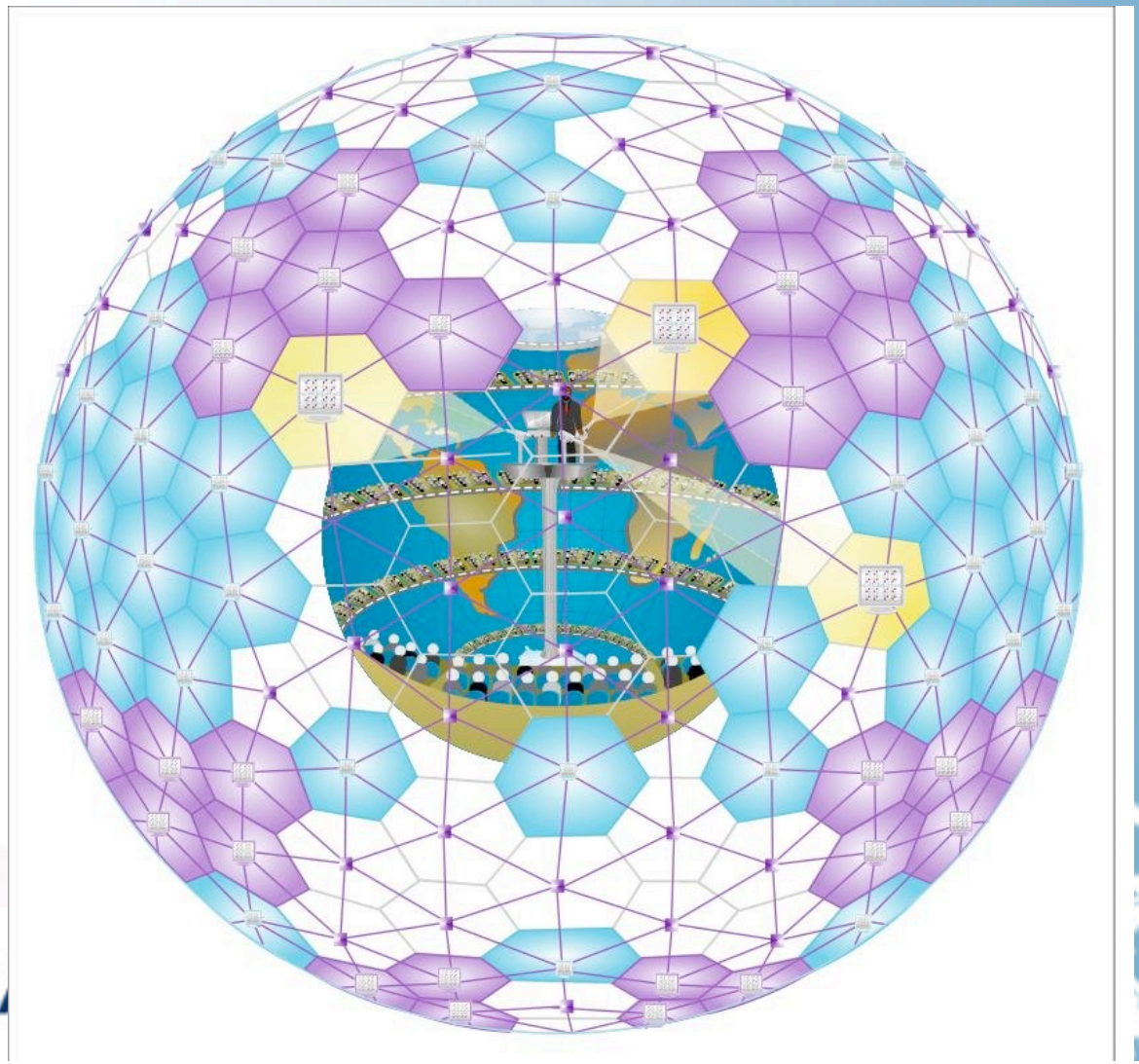
DCMIP Summer School
August 8, 2012

Richardson's Dream (1922)

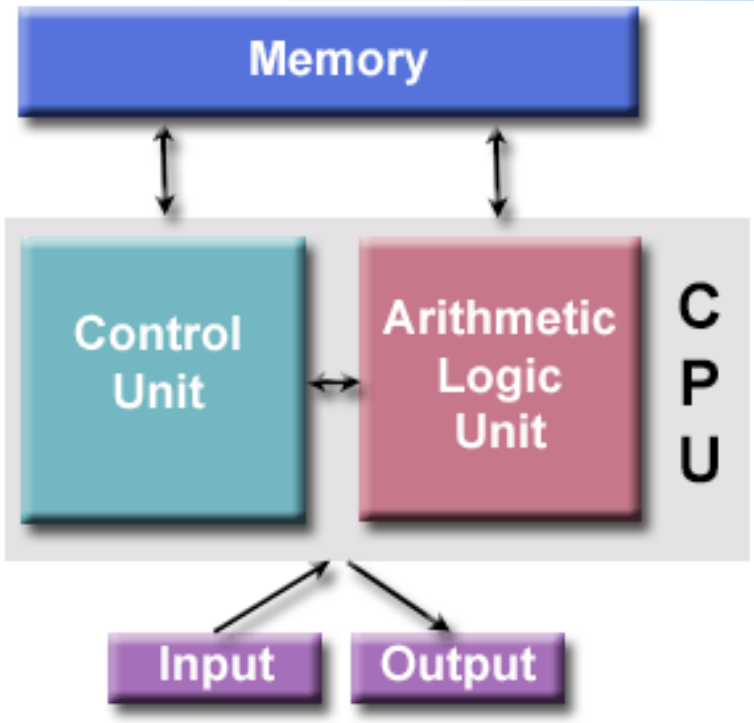
"Imagine a large hall like a theatre, except that the circles and galleries go right round through the space usually occupied by the stage. The walls of this chamber are painted to form a map of the globe. A myriad computers are at work upon the weather of the part of the map where each sits, but each computer attends only to one equation or part of an equation. The work of each region is coordinated by an official of higher rank. Numerous little "night signs" display the instantaneous values so that neighbouring computers can read them. Each number is thus displayed in three adjacent zones so as to maintain communication to the North and South on the map"

Lewis Fry Richardson

Richardson also imagined doing a global calculation – by hand and in parallel!



Von Neumann Architecture (1945)

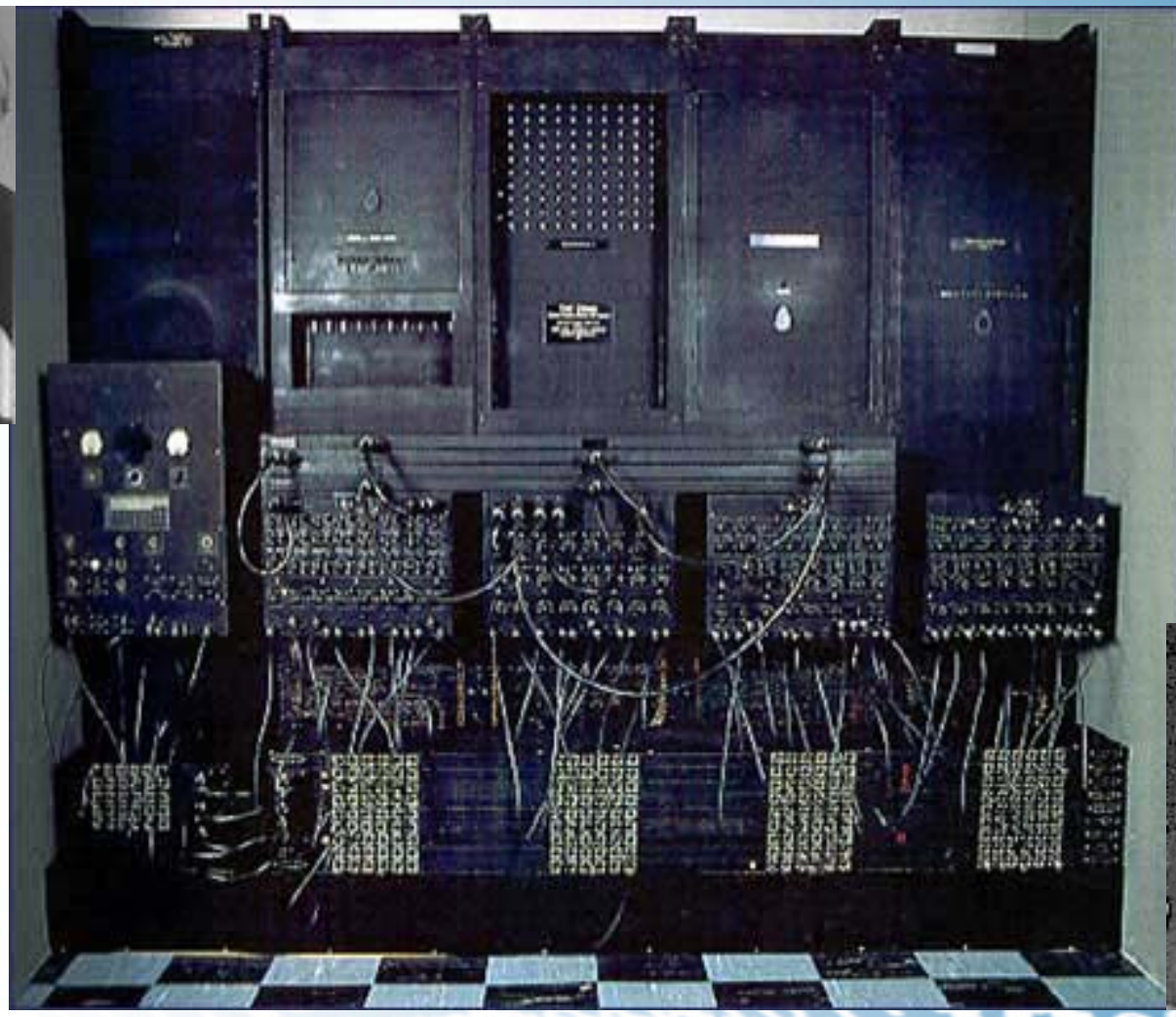


- **Memory stores programs and data**
  - Programs are data
  - Data are programs
- **Control unit**
  - Sequentially controls arithmetic unit
- **Arithmetic unit does the math**
- **I/O to outside world**

Key point: still the basic design building block of modern computers

1950 ENIAC: Many Firsts

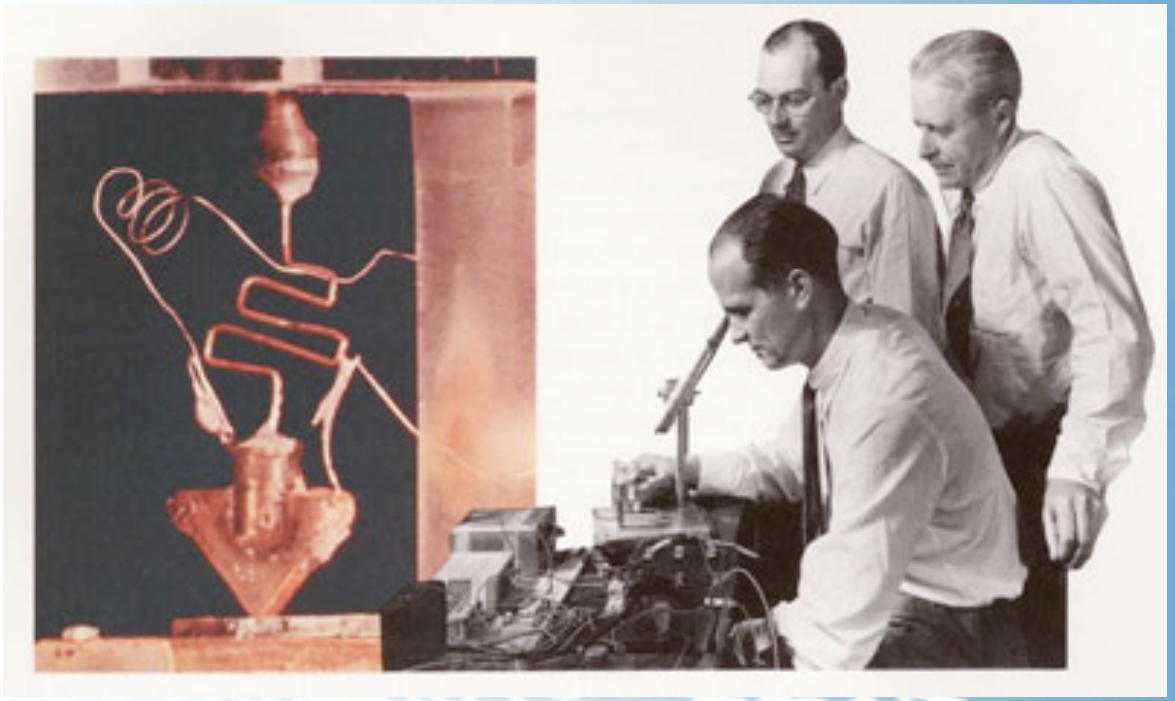
Mauchly & Eckert



5000 Operations per second!

Von Neumann
Charney
Fjörtoft

Game Changer: Invention of the Transistor - 1947

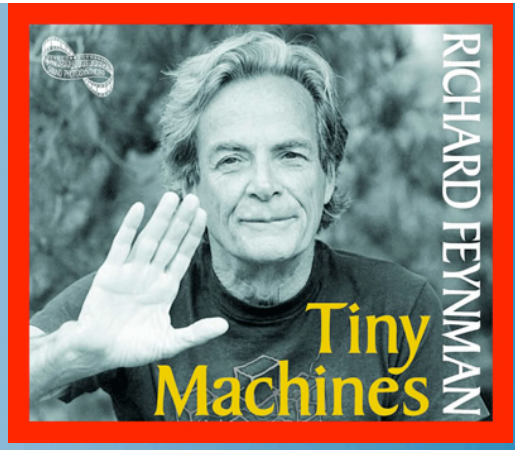


William Shockley (1910-1989), John Bardeen (1908-1991),
Walter Brattain (1902-1987)

CISL Computational & Information Systems Laboratory

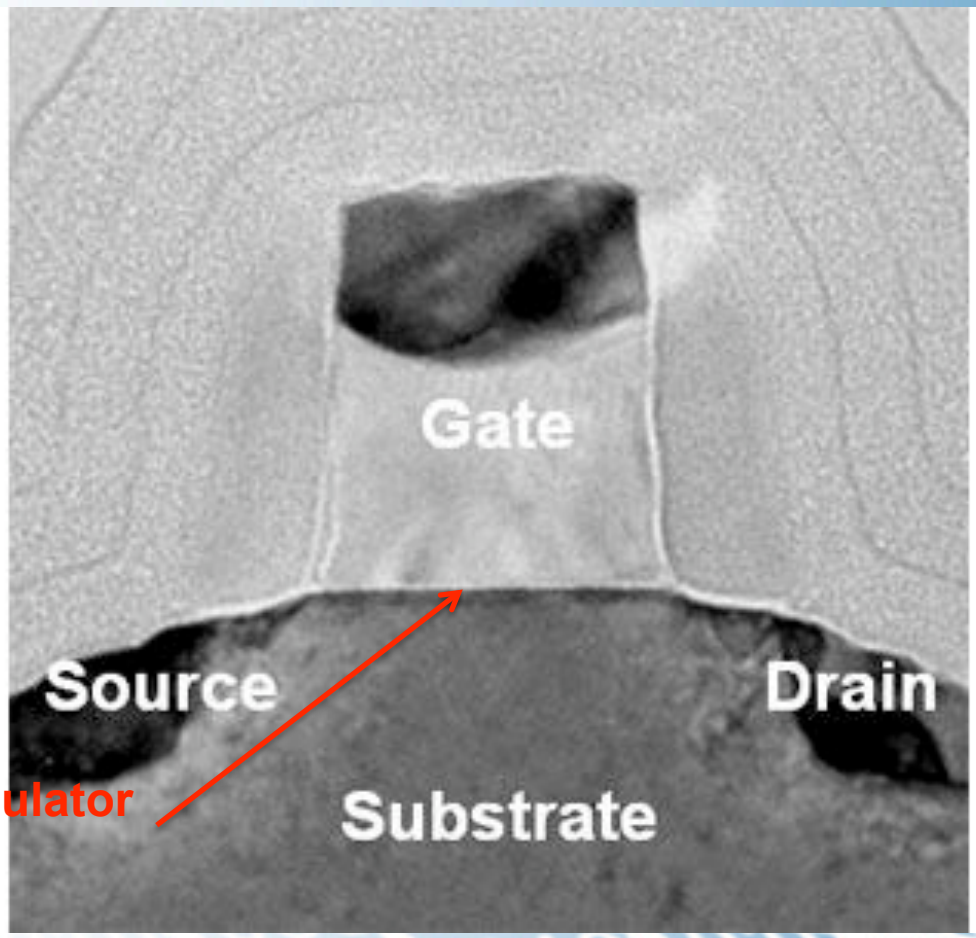
NCAR

Richard Feynman in 1960



- "… I do know that computing machines are very large; they fill rooms. Why can't we make them very small, make them of little wires, little elements – and by little, I mean *little*. *For instance, the wires should be 10 or 100 atoms in diameter, and the circuits should be a few thousand angstroms across."*
- "… all of the information that man has carefully accumulated in all the books in the world can be written … in a cube of material one two-hundredth of an inch wide – which is the barest piece of dust that can be made out by the human eye. *So there is plenty of room at the bottom!"*

Voila! Modern Transistor: ~200 atoms across

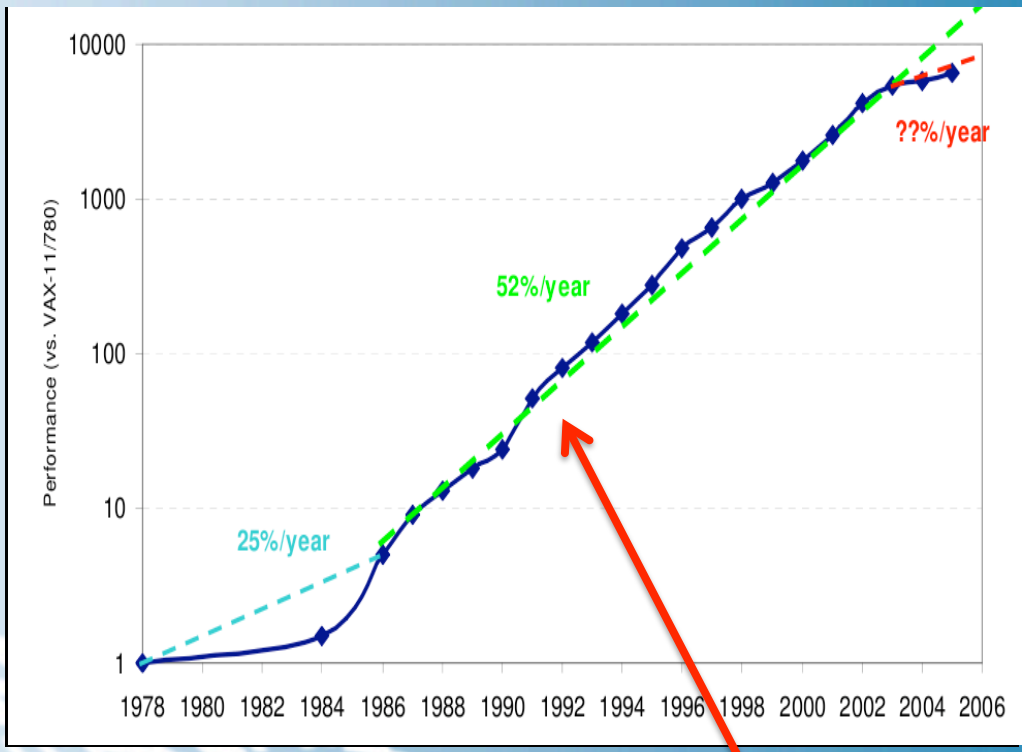


1.2 nm thick insulator

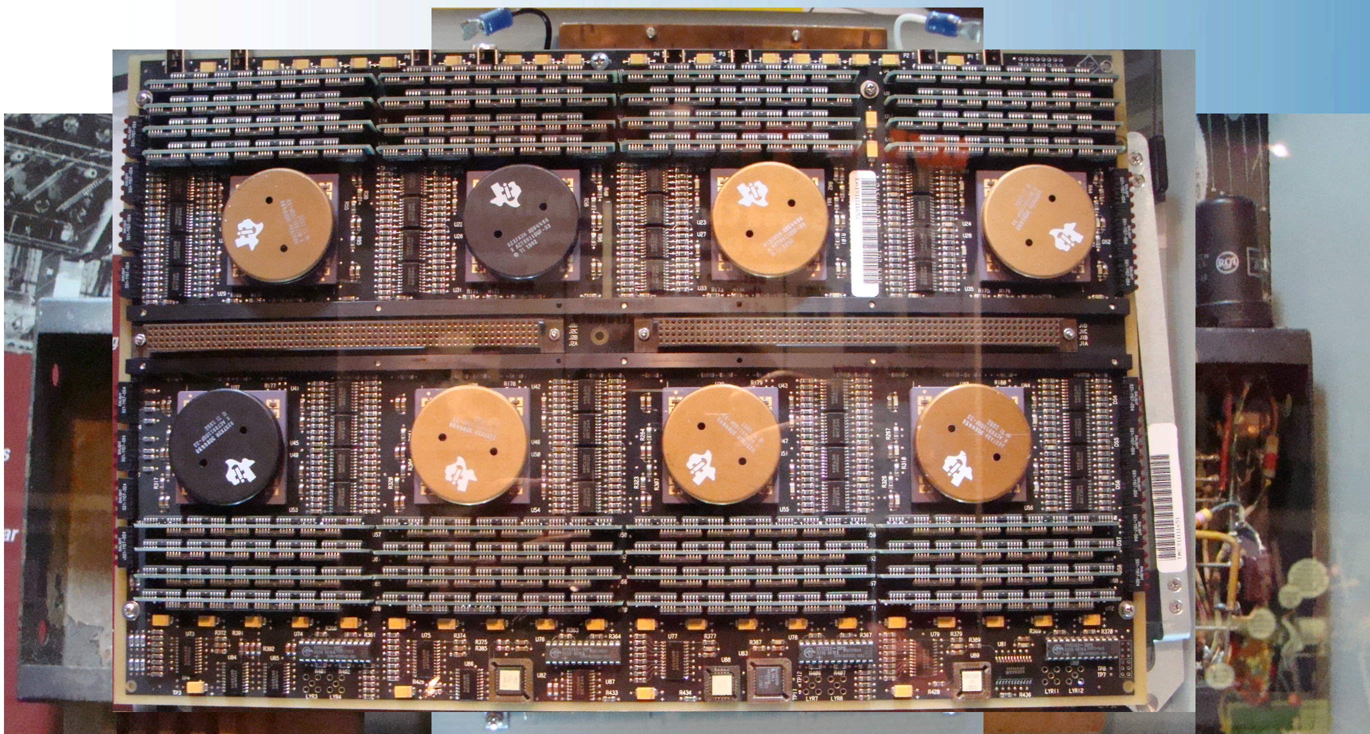
Realizing Feynman's dream!

How did this happen? Moore's Law (1965)

- Not a physical law. More of a self-fulfilling prophesy.
- Moore's Law Says: "Number of transistors doubles every 24 months"
- Not "performance doubles every 18 months." due to House
- Something went wrong with House's Law in the middle of last decade.

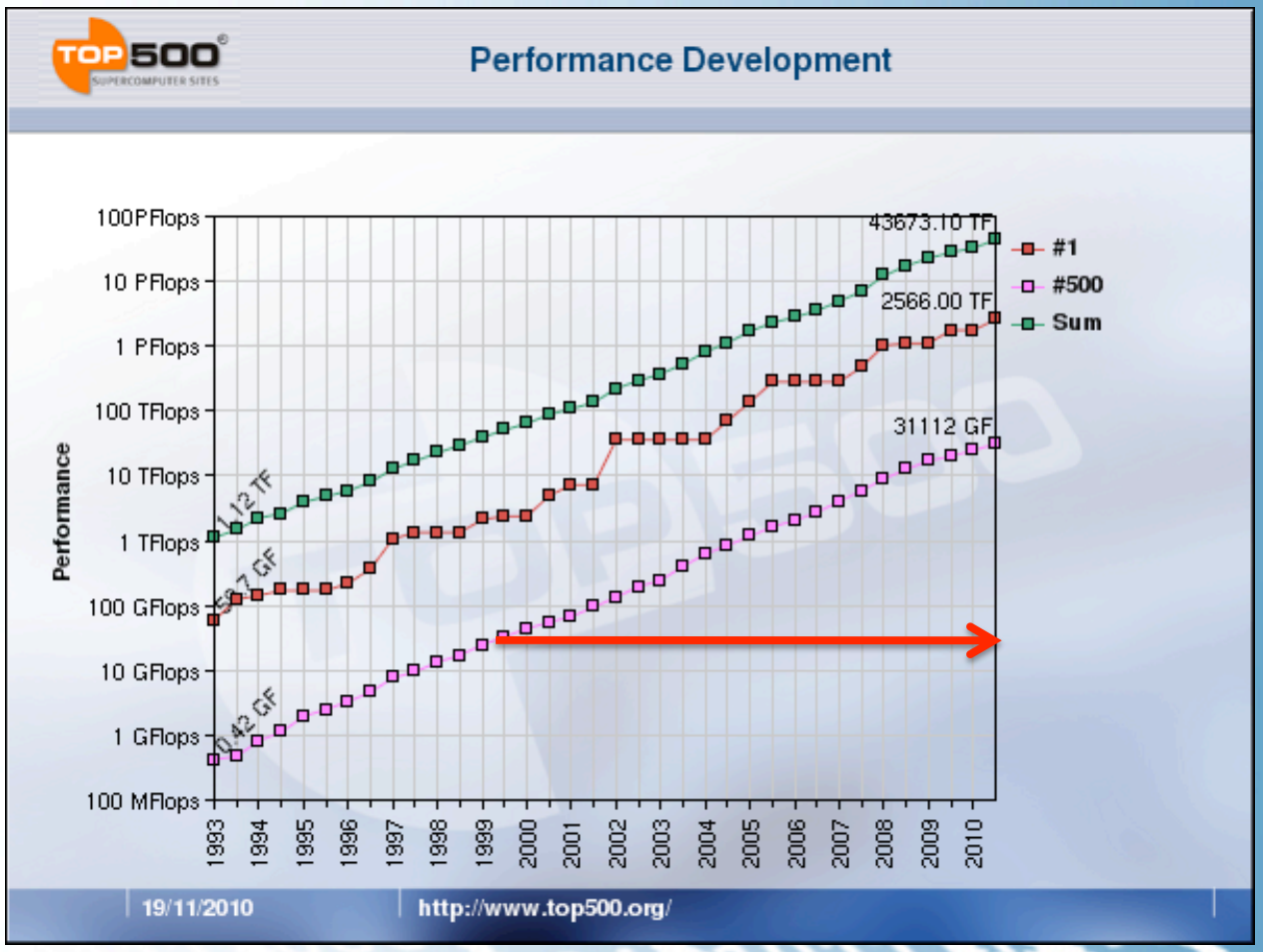


Rapid progress of supercomputers…



Computational & Information Systems Laboratory

The exponential increase in the computational power of supercomputers.



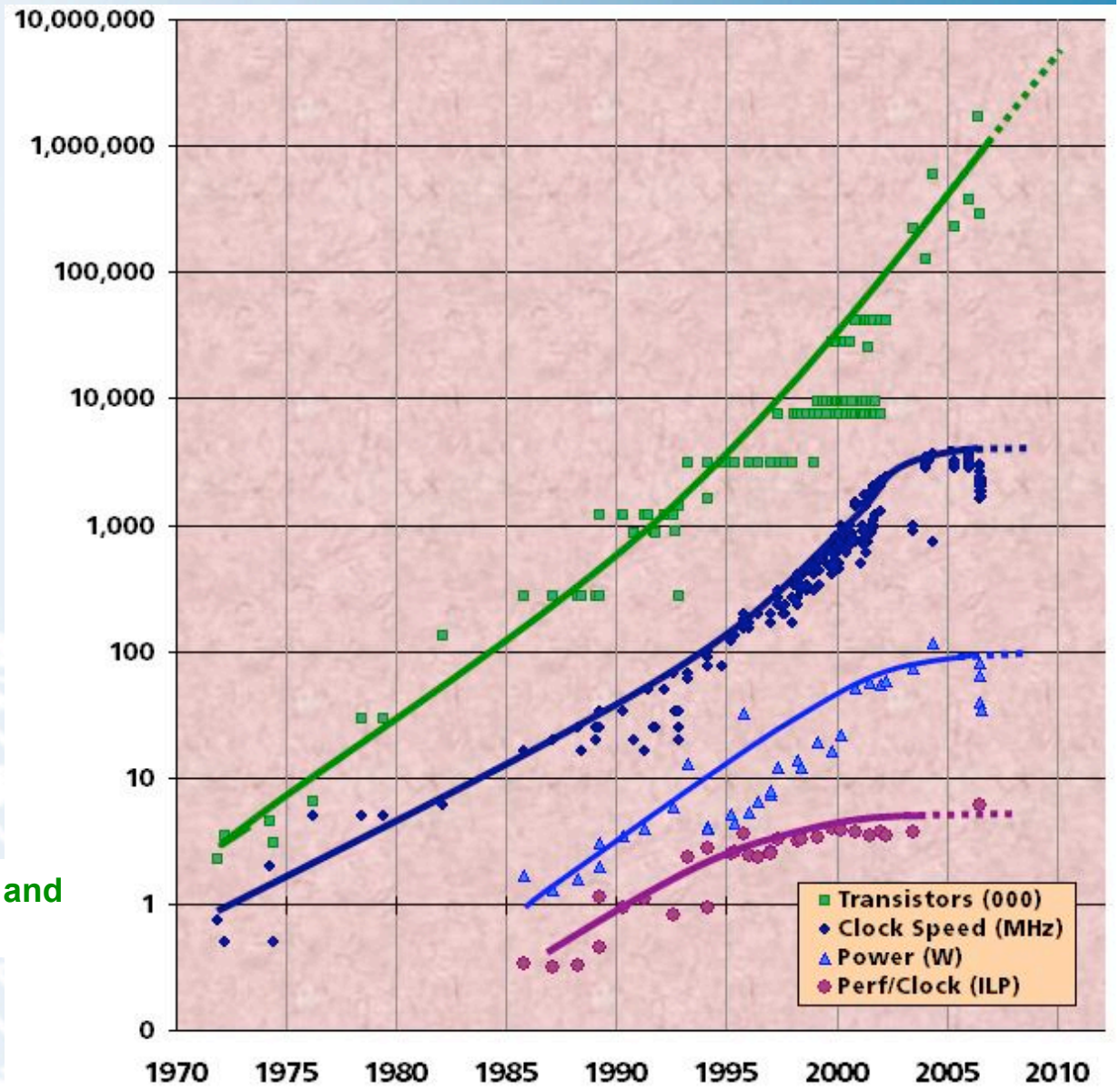
… This laptop would have been on the top 500 list in 1999

What's going on behind the screen?

The Wizard of Oz

Clock Speed Peaked in 2006

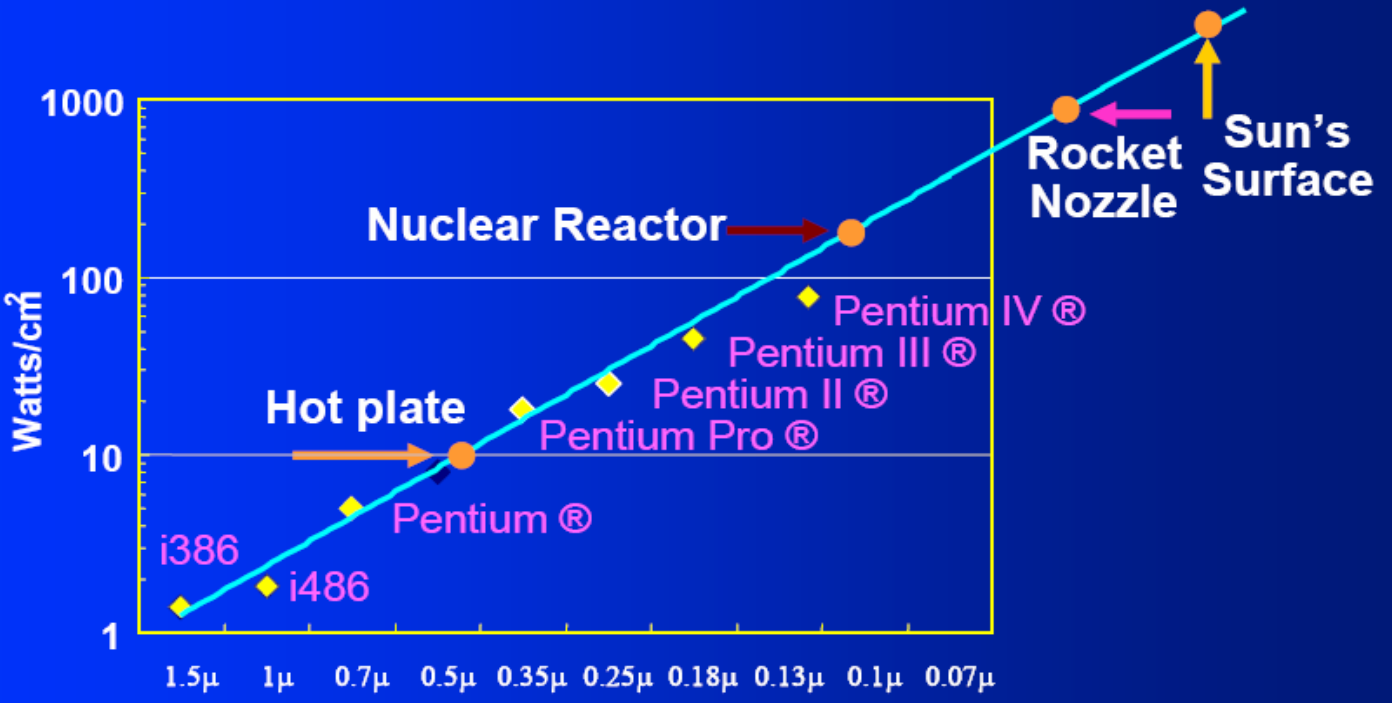
- Chip density is continuing increase ~2x every 2 years
  - Clock speed is not
  - Hit the power wall
  - Hit the memory wall
- There is little or no additional hidden parallelism (ILP)
- Parallelism must be exploited to improve performance



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

Problems behind the curtain:
Clock frequency limited by heat dissipation

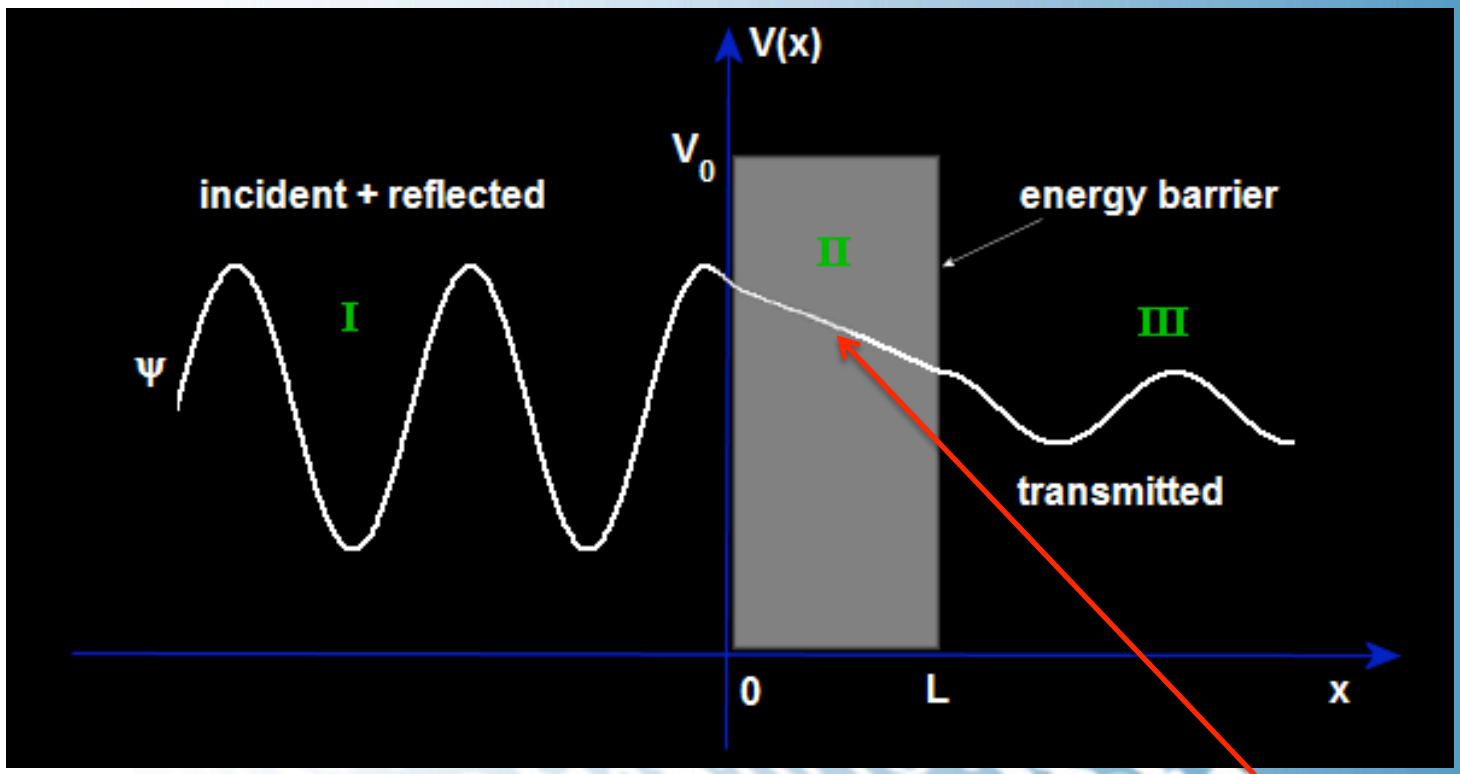
Relentless rise of power density



- 80% increase in power density/generation
- Voltage scales by ~0.8
- 225% increase in current consumption/unit area !

Source: Shekhar Borkar. Intel

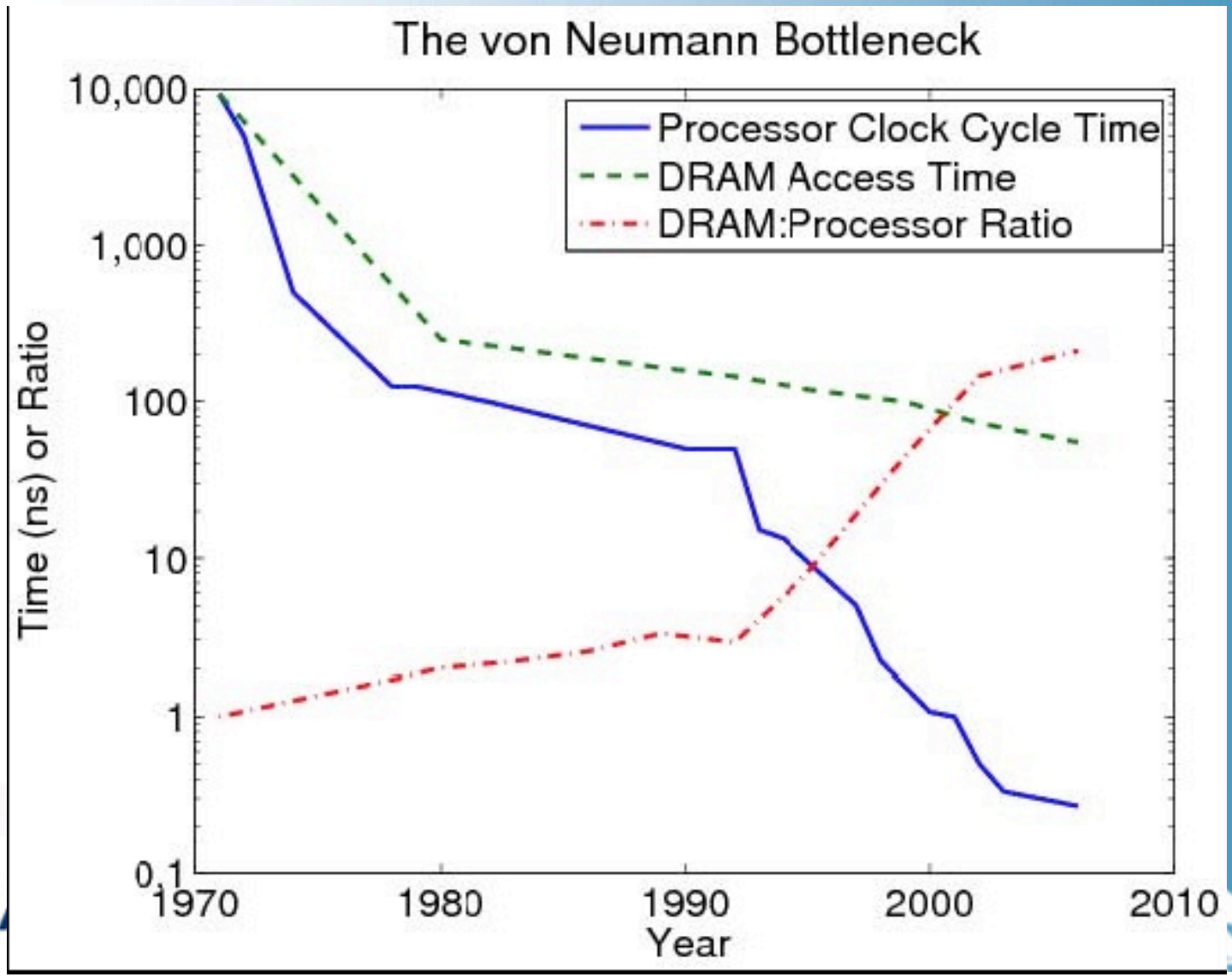
Problems at the atomic scale: *leakage currents due to quantum mechanical tunneling*



tunneling through a barrier

Exponential decay

More problems: the Memory Wall – the gap between processor and memory speed.



What is Cache?

- Cache is a temporary place to keep a copy of information for faster retrieval on subsequent requests
- Cache – fast small memory nearer to the processor than RAM.
- Memory hierarchy:

  Registers –> L1 cache -> L2 cache -> RAM -> Disk->Tape

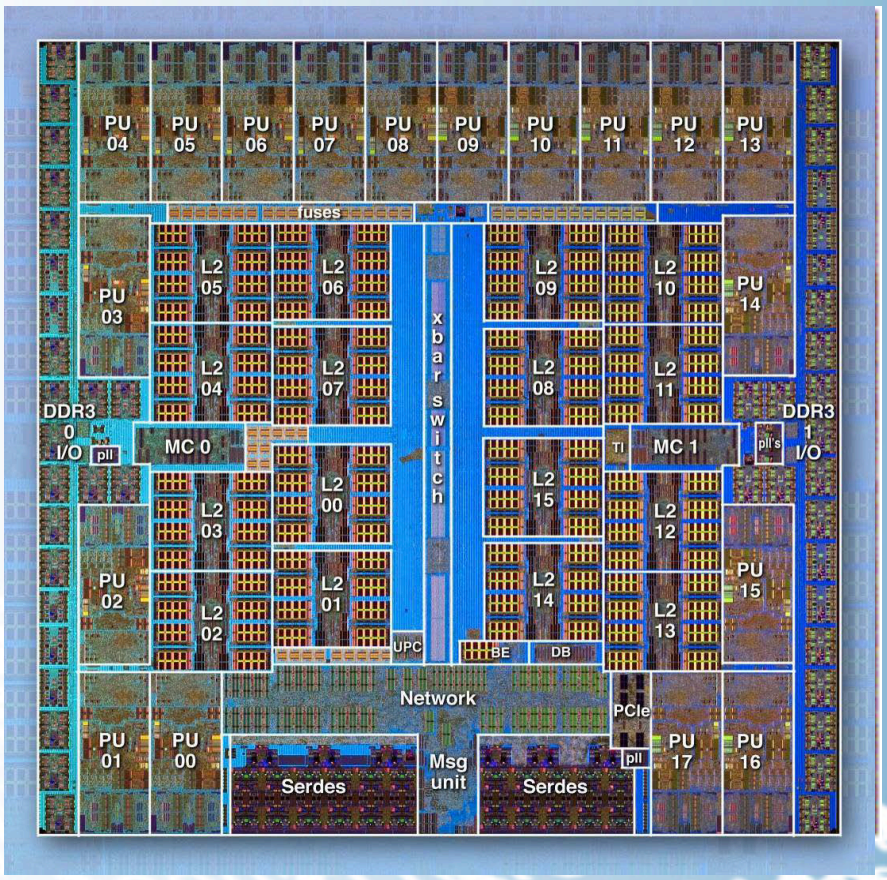
Why is there Cache?

- Memory speed at 7% per year while processor speed growth at ~50% per year
- Retrieval from main memory now takes several hundred clock cycles
- Retrieval from L1 cache takes the order of one clock cycle and from L2 cache takes the order of 10 clock cycles.
- Cache hierarchy (L1, L2) has emerged as memory latency has increased.

More Cache-related Jargon

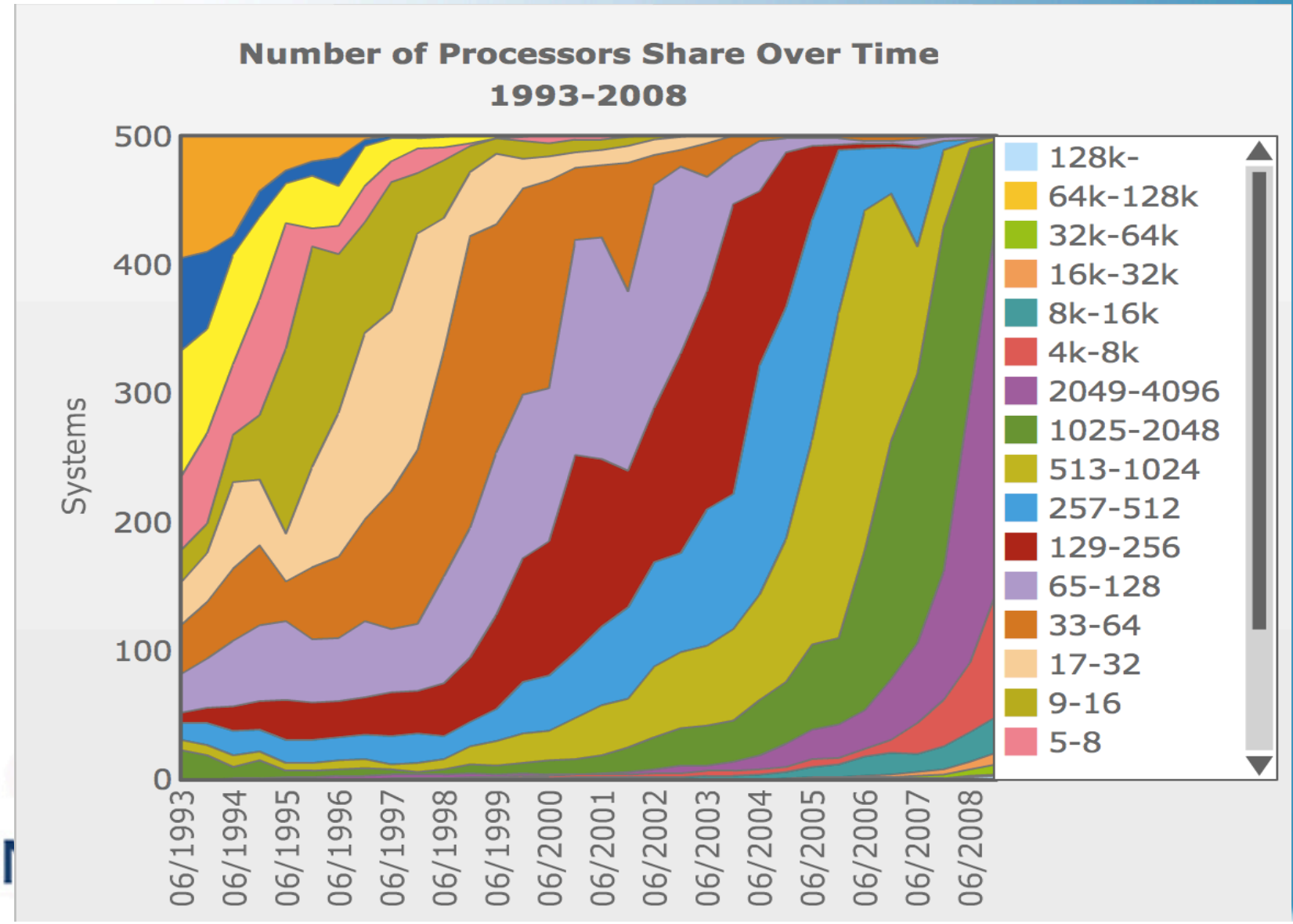
- Cache 'hit' means data is found in cache.
- Cache 'miss' means data is not found there.
- Prefetch used to avoid cache misses at the start of the execution of the program.
- Cache lines are the smallest unit of memory that can be transferred between memory and cache.
- The use of cache lines reduces latency overheads in case of a cache miss for contiguous data accesses.
- Cache coherency – Maintains correctness of data. Important for parallel computing on shared memory.
- Cache reuse (lots of hits) is good – reduces average latency to memory.

Moore's Law -> More's Law: Speed-up through increasing parallelism



How long can we increase the number of cores per chip?

Core Parallelism Doubling Every 18 Months



Computational & Information Systems Laboratory

We need to think in parallel!

Bees do it…

Scaling: definitions

- **Scaling studies involve changing the degree of parallelism. Will we change the problem also?**
- **Strong scaling**
  - Fixed problem size
  - Measures of scientific capability (e.g. years/day)
- **Weak scaling**
  - Problem size grows with additional resources
  - Demonstrates scalable system/algorithm
- **Speed up = $T_s/T_p(n)$**
- **Efficiency = $T_s/(n*T_p(n))$**

Amdahl's Law (1967)

- Fraction F of execution time perfectly parallelizable, i.e.
- No Overhead for
  - Scheduling
  - Communication
  - Synchronization, etc.
- Fraction 1 – F Completely Serial
- Then:

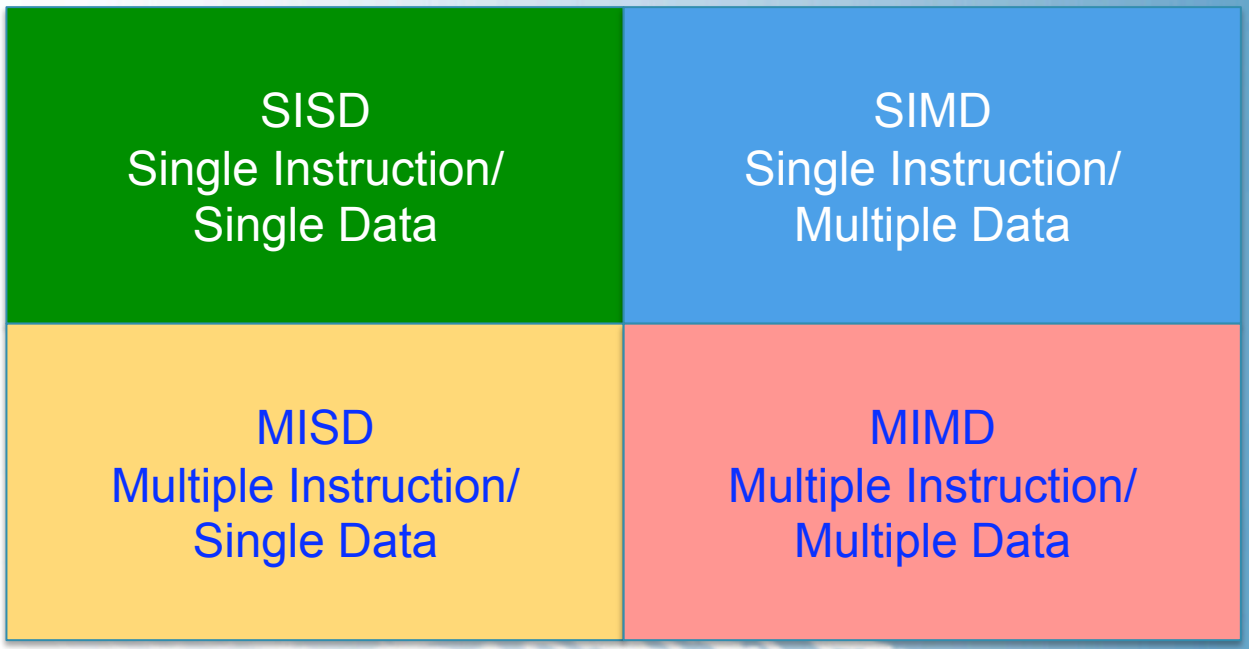
$$\text{Amdahl's Speedup} = \frac{1}{\frac{1 - F}{1} + \frac{F}{N}}$$

Amdahl's Law (cont)

- **The limit as N->∞ says you will never speed up more than: 1/(1-F)**
  - So, if you have 1% serial code: max is 100X.
  - And if you want 10,000X speedup you can have only .01% serial code.
- **You can get 50% of max speedup with**
  - O(1/(1-F)) processors
  - It takes 4x more processors to get 80%
- **Basically everything must be parallelized, so we had better get good at it!**

We need to understand computer architectures…

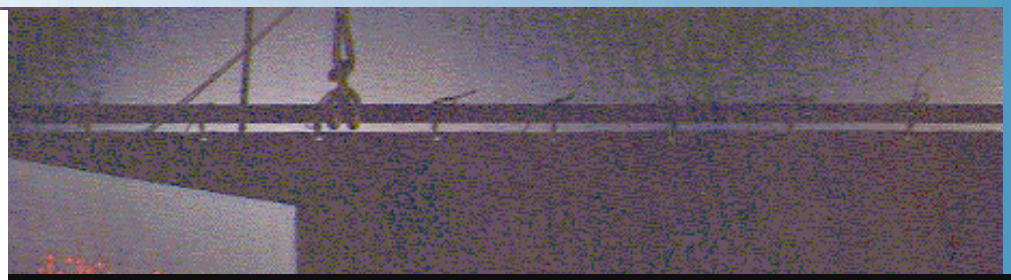
Flynn's Taxonomy (1966)

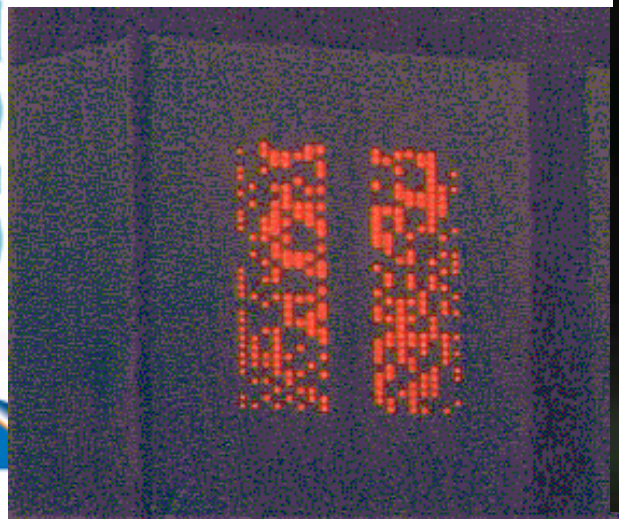


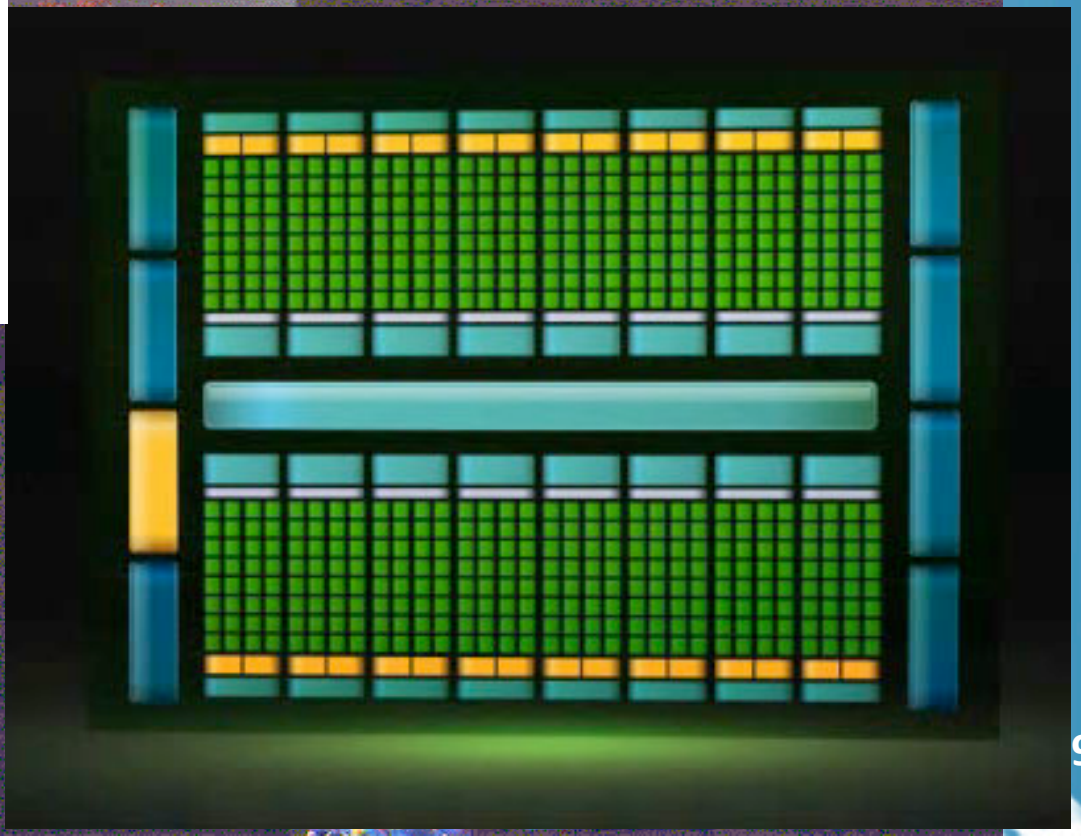
SIMD Computers

- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for problems characterized by a **high degree of regularity**, such as **graphics/ image processing.**

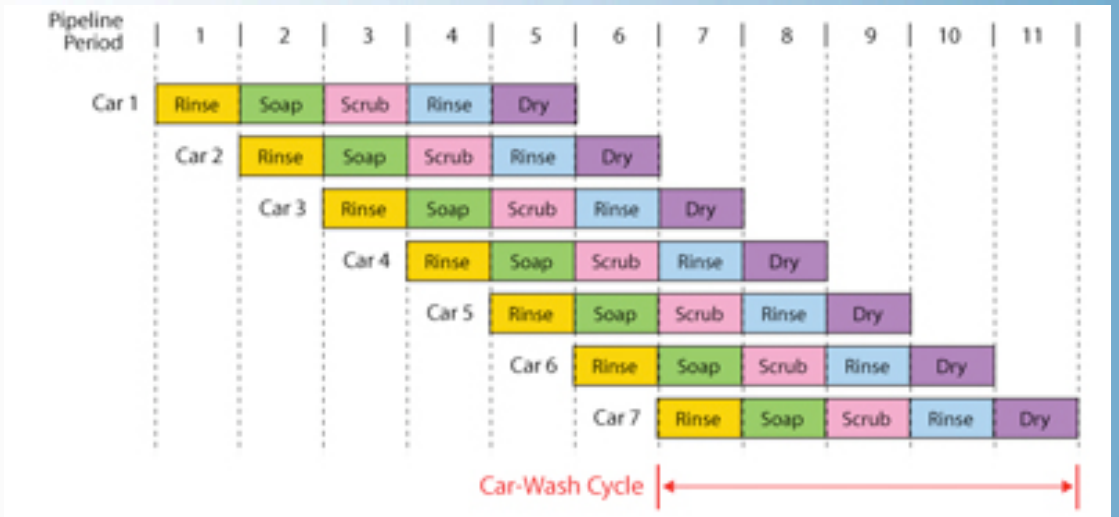
SIMD: Processor Arrays







SIMD: Vector Pipeline

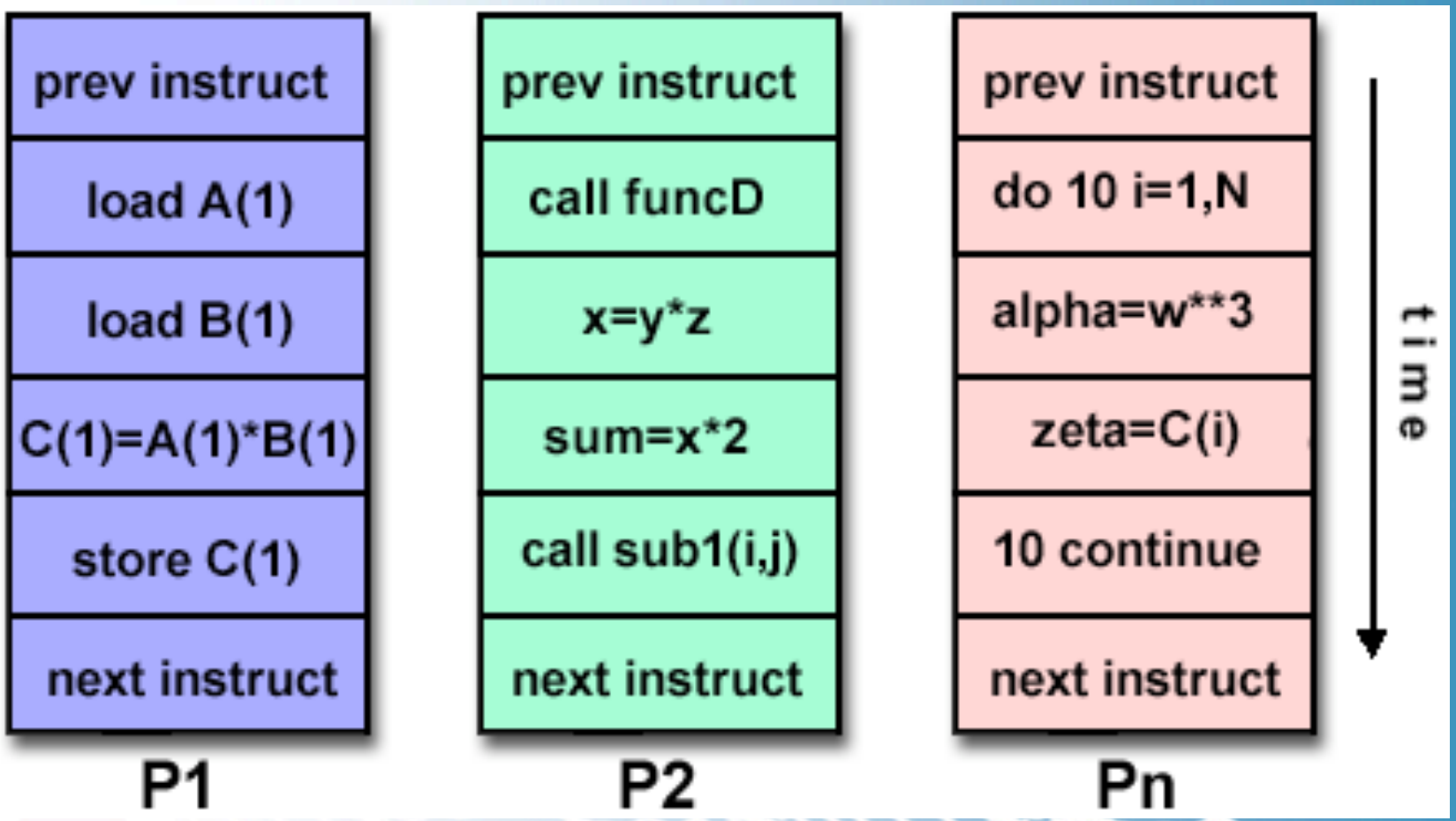


Vector Processors

Iliac IV 1971
1 GFLOPS

Cray YMP – NCAR 1989
2.66 GFLOPS

MIMD Computing



MIMD Systems @ NCAR

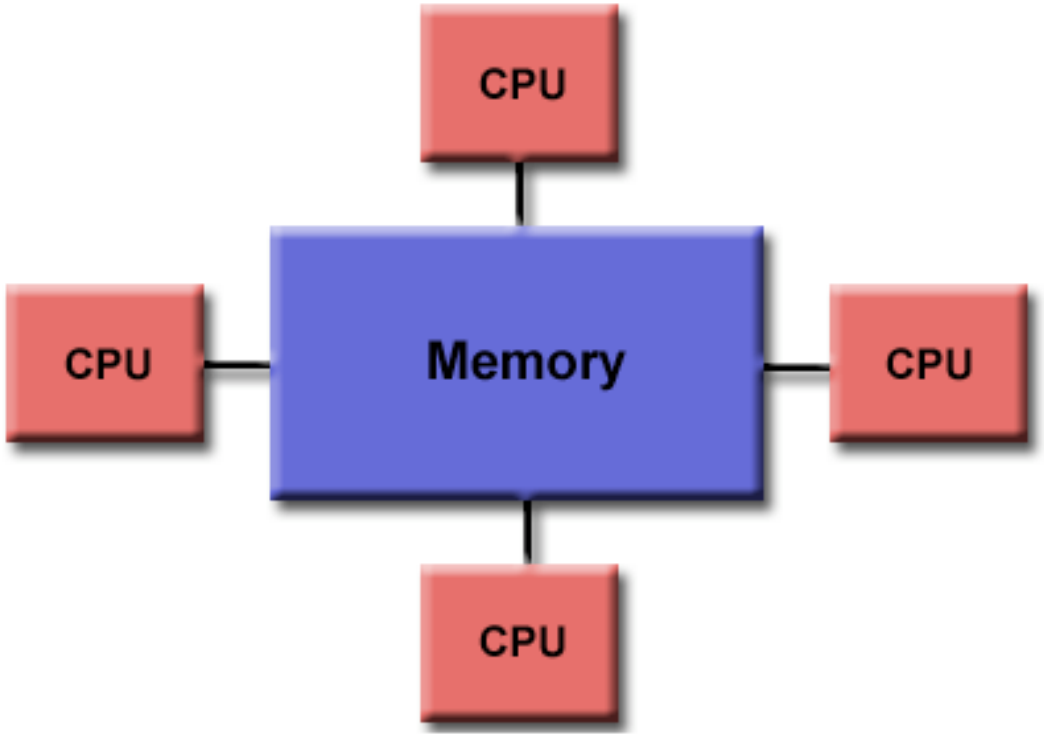
IBM SP – 2000
906 GFLOPS

Cray T3D – 1994
9.6 GFLOPS

NCAR

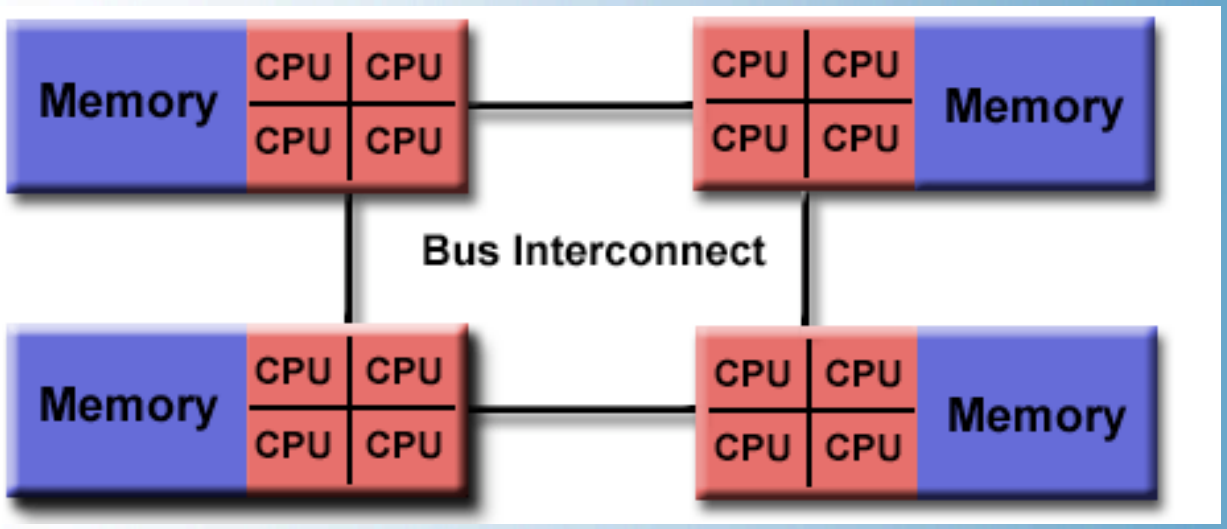
IBM PWR-6 2008
77 TFLOPS

Parallel Architectures: Shared Memory uniform memory access (UMA)



- Symmetric Multi-Processor
- Global address space
- Uniform access times
- Hardware cache coherence

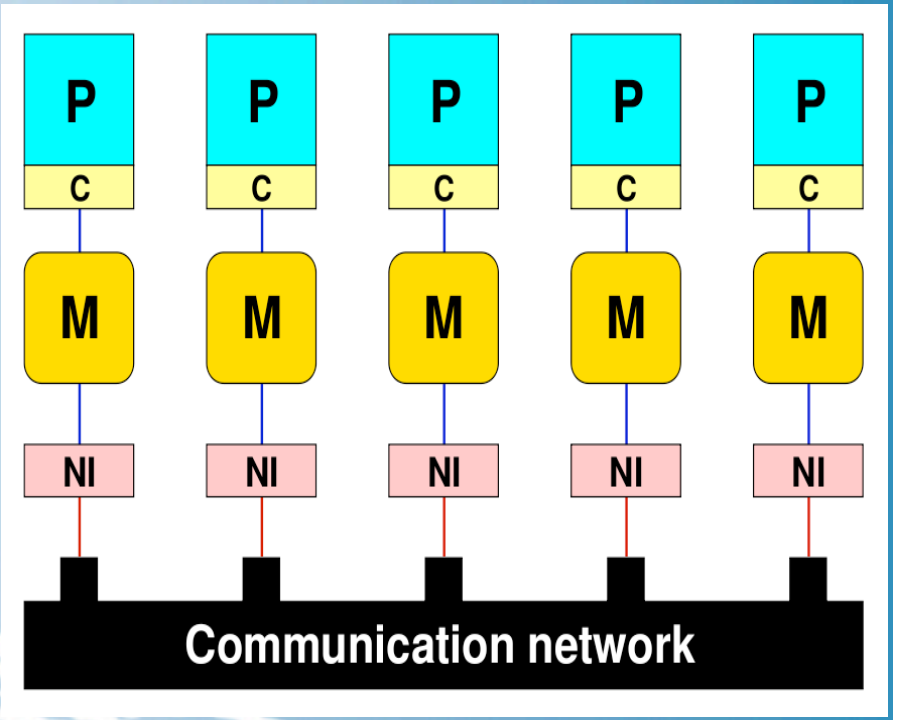
Parallel Architectures: Non-Uniform Memory Access



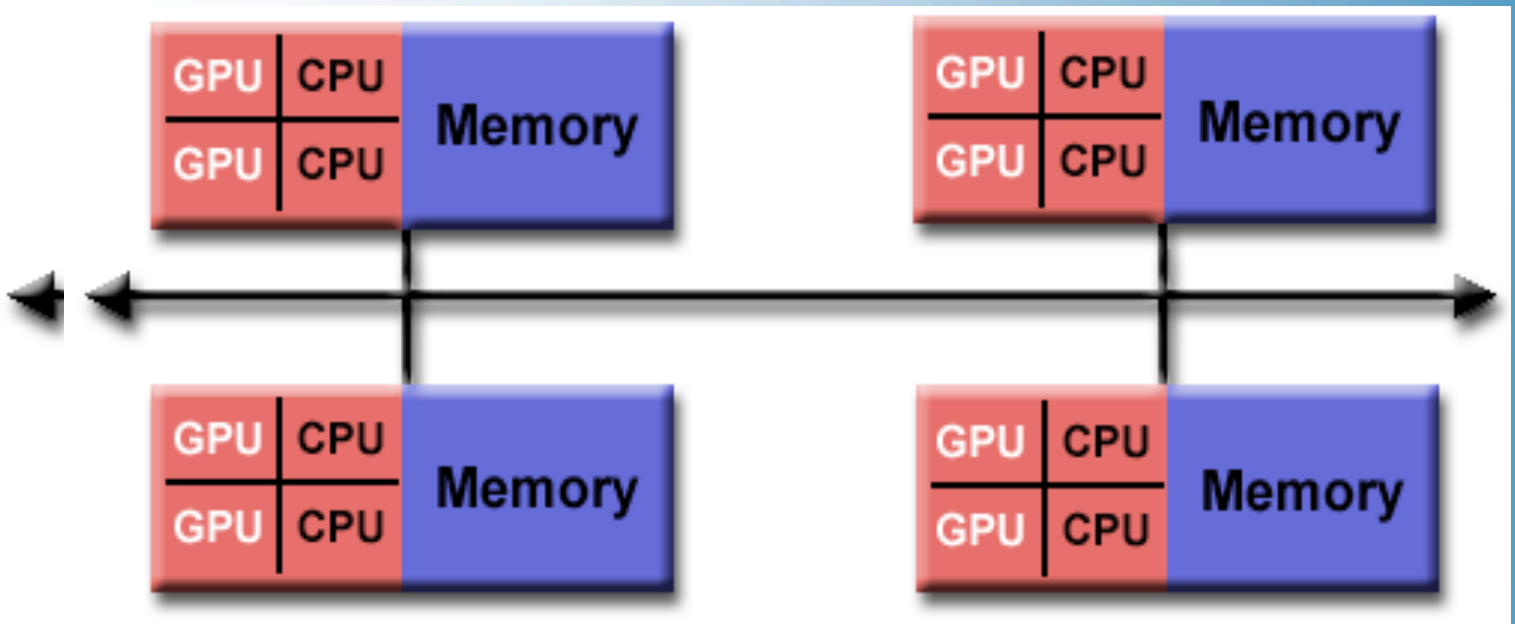
- Made by connecting up multiple SMPs
- One SMP can directly access memory of another SMP
- Processors have unequal access time to all memories
- Memory access across bus interconnect slower

Distributed Memory—Message Passing Architectures

- Each processor P (with its own local cache C) is connected to exclusive local memory, i.e. no other CPU has direct access to it.
- Each node comprises at least one network interface (NI) that mediates the connection to a communication network.
- On each CPU runs a serial process that can communicate with other processes on other CPUs by means of the network.



Parallel Architectures: Hybrid Distributed/Shared Memory Systems

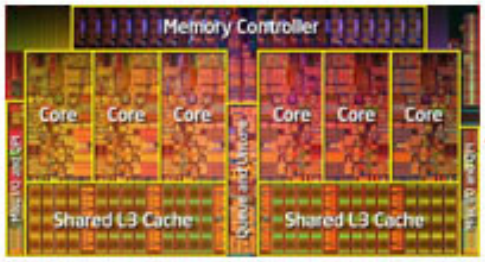


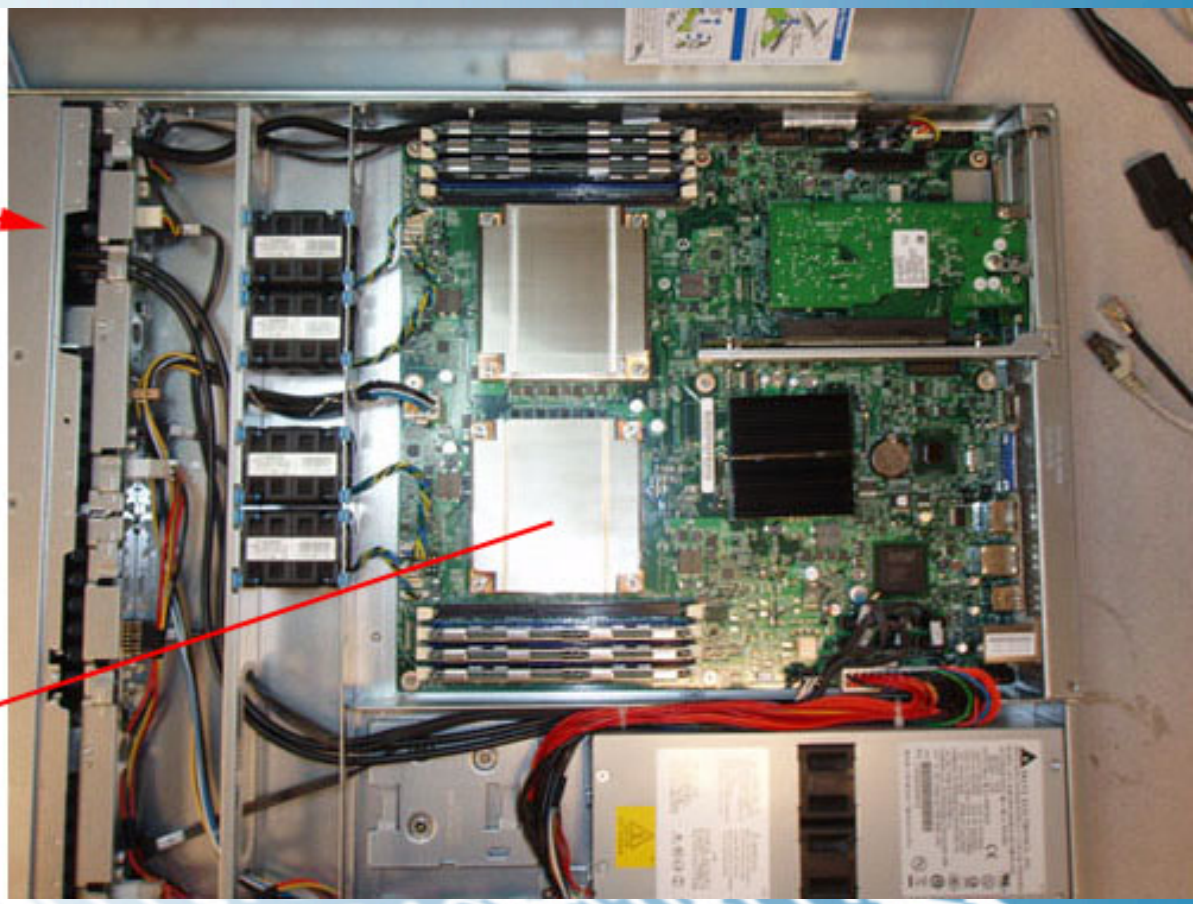
Anatomy of a Parallel Computer: nodes, processors, sockets, cores and all that

Supercomputer - each blue light is a node

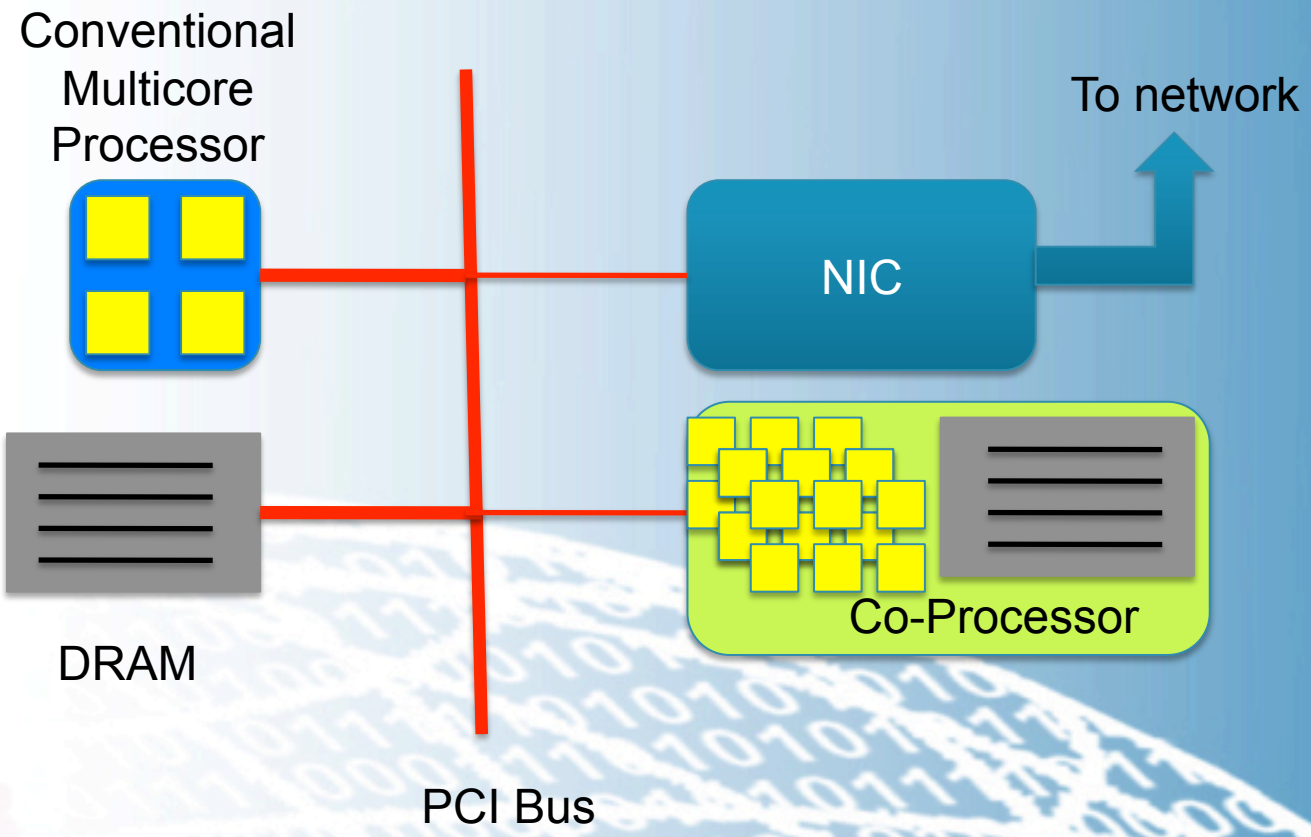
Node - standalone Von Neumann computer

CPU / Processor / Socket - each has multiple cores / processors.

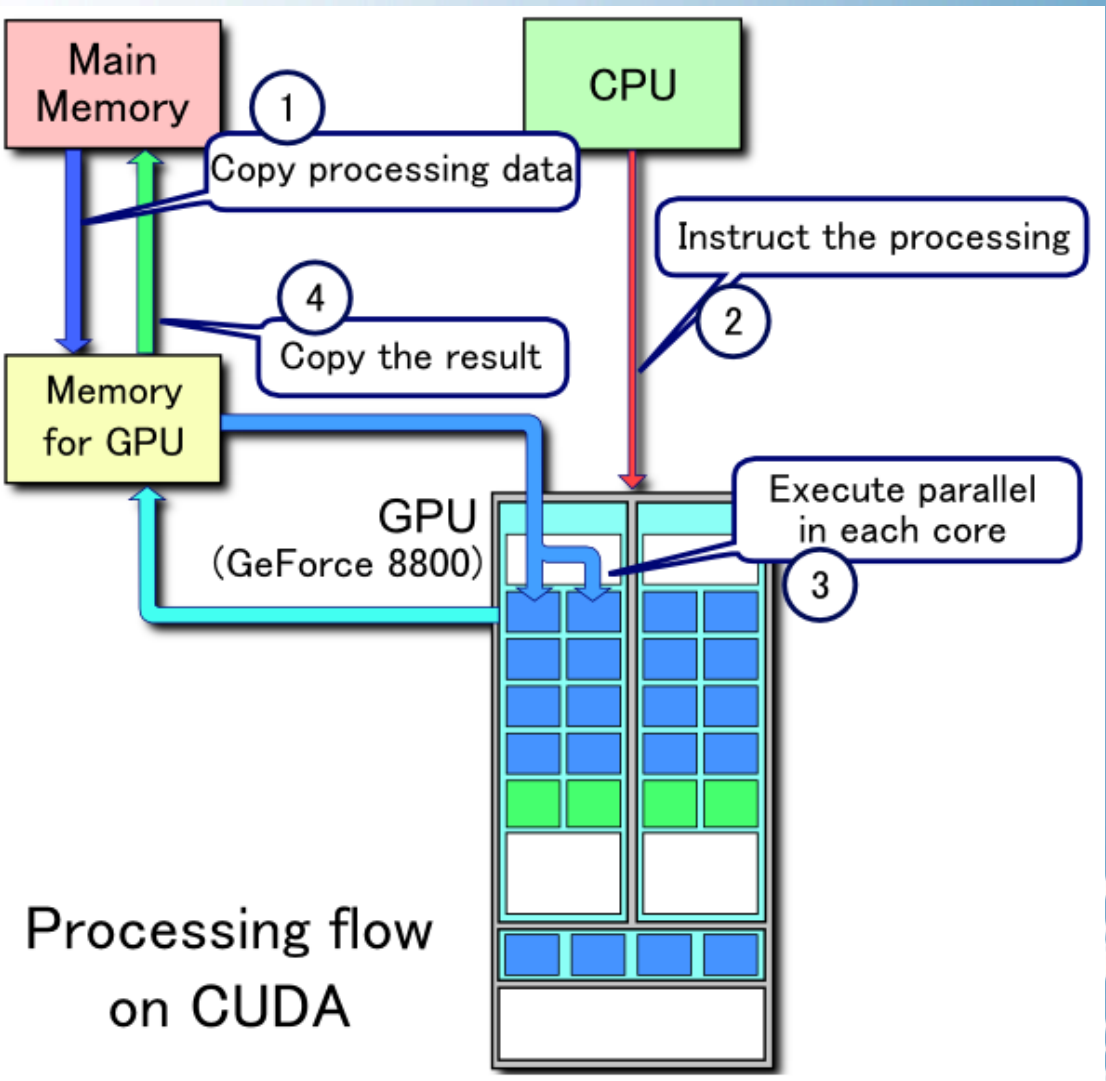




Coprocessor Offload: The heterogeneous node architecture



How to talk to coprocessors



Processing flow on CUDA

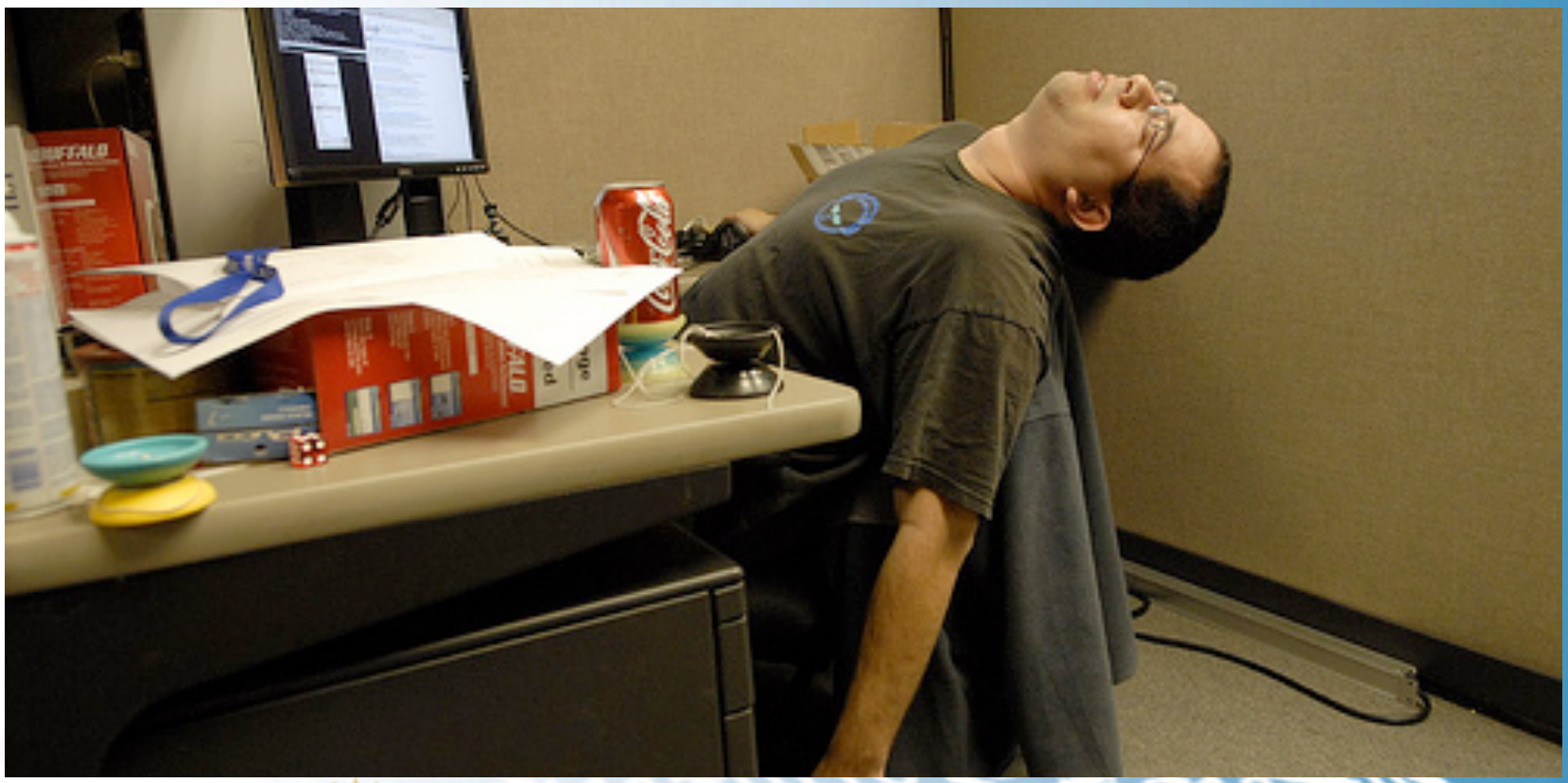
Computational Intensity (CI)

- Compute Intensity:

  CI = Total Operations/(Input + Output data)

- Or, GFLOPS = CI*Bandwidth
- Bandwidth expensive, flops cheap
- The higher the CI, the better chance of offloading to a coprocessor.

Understanding Parallel Programming Paradigms…



Some Parallel Programming Models:

- Threads
- Message Passing
- Hybrid (Threads + Message Passing)
- CUDA (SIMD programming)
- Need to master all of these to be successful!

Programming Models: Threads

- A library of subroutines that are called from within parallel source code (e.g. pthreads)
- A set of compiler directives imbedded in either serial or parallel source code (OpenMP)
- Language extensions (e.g. Cilk)

Message Passing Interface (MPI)— A distributed memory parallel programming API

- **All variables are local! No concept of shared memory**
- **Each process is numbered, called its rank.**
- **The program is written in a sequential language (FORTRAN)**
- **Data exchange between processes through library calls**
- **MPI System requires information about**
  - Which processor is sending/receiving the message.
  - Data locations on the sending/receiving processors.
  - What kind of data and how much is being sent/received.
- **Works well with the single program multiple data (SPMD) approach**

CUDA: Devices and Threads

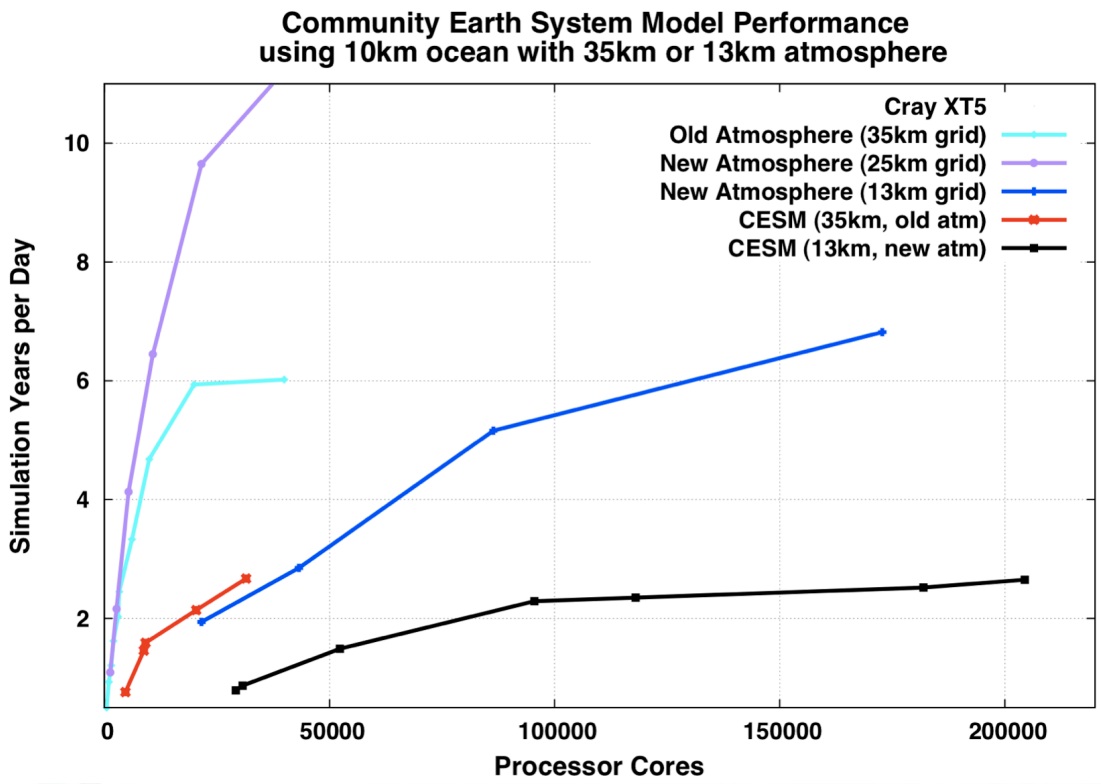
- **A compute device**
  - Is a coprocessor to the CPU or host
  - Has its own DRAM (device memory)
  - Runs many threads in parallel
  - Is typi cally a GPU but can also be another type of parallel processing device
- **Data-parallel portions of an applicatio n are expressed as device kernels which run on many threads**
- **Differences between GPU and CPU threads**
  - GPU threads are extremely lightweight
    - Very little creation overhead
  - GPU needs 1000s of threads for full efficiency
    - Multi-core CPU needs only a few

Important Ingredients (and problems) for Earth System Modeling

- **Locality and grid uniformity**
  - **Key to achieving scalable applications**
  - Local refinement and global solvers are problematic
- **Flexible expression of parallelism**
  - E.g. CESM components use **hybrid MPI and OpenMP**
- **Load balancing strategies**
  - Either static (ocean model) or dynamic (e.g. AMR)
- **Small memory footprint per thread**
  - Memory scalability of all components - parallel I/O
  - **Memory/core -> 0 on future systems**

CESM Computational Performance

(courtesy of Pat Worley)



- For 35Km CAM / 10 km POP/ CICE, CESM - performance is constrained by CAM, CICE and POP
- For 13km CAM / 10KM POP/ CICE, CESM performance is not constrained by CAM
- Spectral element-based atmospheric dynamics permits scalable CESM performance at high resolution.

Thanks! Questions?