Introduction to High Performance Computing

M. D. Jones, Ph.D.

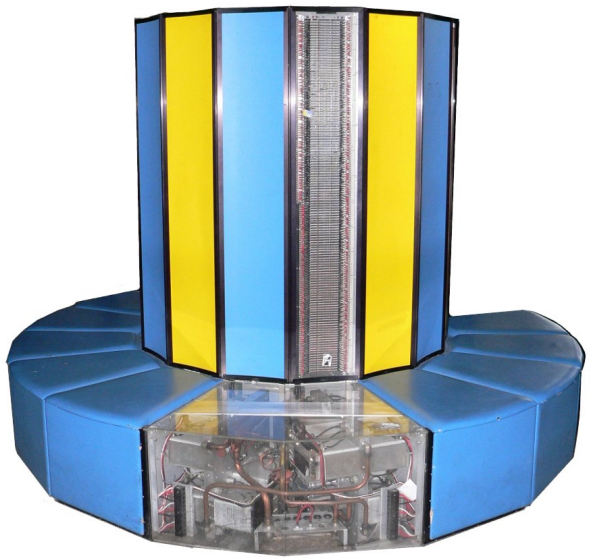
Center for Computational Research
University at Buffalo
State University of New York

HPC-I 2013

HPC

- HPC = High Performance Computing
- and by “High Performance” we mean what?
- Unfortunately, HPC is a slippery term - let’s construct a more careful definition ...

High Performance is Quite Relative



Consider the (rather unfair) comparison ...

- The Cray-1 Supercomputer, circa 1976 (first commercially successful vector machine)
  - Weight - about 2400 kg, Cost - roughly 8M$ (US)
  - Performance - about 160 million floating point operations per second (MFlop/s)

- A modern day PC, circa 2013 (say running a 4th generation quad-core 3 GHz Intel i7 CPU)
  - Weight - roughly 5 kg, Cost - 1000$ (US)
  - Performance - about 192 billion floating points operations per second (192 GFlop/s)
- So that is roughly an increase of 1200 in peak performance, and a whopping reduction of 8000 (not accounting for inflation, which would be about another factor of 4 or so) in price.
- We will come back to this notion of “peak performance” later ...

Attributes of HPC

Something that 'everyone knows' what it means, so somehow manages to escape concrete definition. Most will likely agree with the following key attributes:

- Problems in HPC require "significant" computational resources (where significant, in this case, can be taken to be more resources than are generally available to the average person)
- Data sets require large (in the same sense as "significant" above) amounts of storage and/or processing
- Needs to collectively operate on a (possibly distributed) network

Operational Definition of HPC

Definition

(HPC): HPC requires substantially (more than an order of magnitude) more computational resources than are available on current workstations, and typically require concurrent (parallel) computation

Supercomputers/Supercomputing

The term **"Supercomputing"** is usually taken to be the top-end of HPC:

- Massive computing at the highest possible speeds
- Usually reserved for the very fastest (or elite) HPC systems
- An old definition - 'any computer costing more than ten million dollars', which is now quite obsolete
- The Top500 list has been used to rank such systems since 1993, but we will come back to that ...

Peak Performance

Definition

(Peak Performance): The theoretical maximum performance (usually measured in terms of 64-bit (double precision, on most architectures) floating point operations per second, or Flop/s) achievable by a computing system.

- Typically one simply adds the floating point capabilities in the most simplistic way
- MFlop/s, GFlops/s, TFlop/s, and coming soon to a facility near you, PFlop/s (in the usual metric usage).

Peak Performance Example

Example

Intel has transitioned from the **netburst** core architecture (2003-2006) to the **core/core2/i7** - the netburst processors could do a maximum of 2 floating point operations per clock tick, so a 3GHz netburst core could do 6 GFlop/s as its theoretical peak performance (TPP). The first generation i7 processors doubled the floating point capacity (4 Flop/s per clock tick) and the second and third generation quadrupled (8 Flop/s per clock tick), so each core of a 3GHz generation 3 (“Ivy Bridge”) i7 would have a TPP of 48 GFlop/s. The fourth generation ("Haswell") is supposed to again double the double-precision floating point capacity to 16 floating point operations per clock tick.

Significance of TPP

So what about the theoretical peak performance (TPP)?

- Guaranteed "not-to-exceed" limit on what you can hope to achieve
  - In fact, very few applications exceed a few percent
  - Careful tuning might get you to 10-20%, unless you have an algorithm dominated entirely by easily predictable data patterns dominated by simple computation
- Gives you a relatively robust way to compare very different platforms
- Behold, TPP as the computational equivalent of Mt. Everest:
  - Bandwidth of memory to cache/processor is limited
  - Conflicts in cache/memory access
  - Communication time dominates computation time
  - Like Mt. Everest, you can kill yourself trying to achieve TPP ...

Experimental/Special Purpose

1939 Atanasoff & Berry build prototype electronic/digital computer at Iowa State

1941 Conrad Zuse completes Z3, first functional programmable electromechanical digital computer

1943 Bletchley Park operates Colossus, computer based on vacuum tubes, by Turing, Flowers, and Newman

1946 ENIAC, Eckert and Mauchly, at the University of Pennsylvania

1951 UNIVAC I (also designed by Eckert and Mauchly), produced by Remington Rand, delivered to US Census Bureau

1952 ILLIAC I (based on Eckert, Mauchly, and von Neumann design), first electronic computer built and housed at a University

The Cray Years

1962 Control Data Corp. introduces the first commercially successful supercomputer, the CDC 6600, designed by Seymour Cray. TPP of 9 MFlop/s

1967 Texas Instruments' Advanced Scientific Computer, similar to CDC 6600, includes vector processing instructions

1968 CDC 6800, Cray's redesign of 6600, remains fastest supercomputer until mid 1970s. TPP of about 40MFlop/s.

1976 Cray Research Inc.'s Cray-I starts vector revolution. TPP of about 250MFlop/s.

1982 Cray X-MP, Cray's first multiprocessor computer, original 2 processor design had a TPP of 400MFlop/s, included shared memory access between processors.

Clusters Take Over

1993 Cray introduces the T3D - an MPP (Massively Parallel Processing) based on 32-2048 DEC Alpha (21064 RISC, 150MHz) processors and a proprietary 3D torus interconnect

1997 ASCI Red at Sandia delivers first TFlop/s (on the Linpack benchmark) using Intel Pentium Pro processors and a custom interconnect.

2002 NEC's Earth Simulator is a cluster of 640 vector supercomputers, delivers 35.61 TFlop/s on the Linpack benchmark.

2005 IBM's Blue Gene systems regain top rankings (again, according to Linpack) using massive numbers of embedded processors and communication sub-systems (more later), each running a stripped down Linux-based operating system.

Clusters Take Over (cont'd)

2008 IBM deploys a hybrid system of Opteron nodes coupled with Cell processors interconnected via Infiniband, achieves first sustained PFlop/s on top500 ...

2010 Tianhe-1A at the National Supercomputing Center in Tianjin, China, mix of 14,336 Intel Xeon X5670 processors (86,016 cores) and 7168 Nvidia Tesla M2050 general purpose GPUs, custom (Arch) interconnect, 2.566 PFlop/s

2011 K computer, at RIKEN in Kobe, Japan, 68544 2.0GHz 8-core Sparc64 VIIIfx processors (548,352 cores), custom (Tofu) interconnect, 8.162 PFlop/s

2012 Sequoia, IBM Blue Gene Q at LLNL, 1,572,864 1.6GHz PowerPC A2 cores, custom BG interconnect, 16.32 PFlop/s

Off-The-Shelf Supercomputers

The last decade has seen so-called **COTS** (Commodity, Off-The-Shelf Systems) replace, well, almost everything else, at the heart of HPC/Supercomputing.

- Starting in 1993, Donald Becker and Thomas Sterling (at the Center of Excellence in Space Data and Information Sciences, or CESDIS) at NASA/Goddard proposed and prototyped a COTS system based on 16 Intel DX4 processors (100MHz) and channel bonded 10Mbit/s Ethernet. This project was named after the epic Norse hero, **Beowulf**.
- HPC then underwent a remarkable transformation.

Beowulf Slays Grendel

In this case, Grendel consisted of the cadre of specialized manufacturers that supplied HPC hardware. Following is an (abbreviated, I am quite sure) list of vendors that either went completely out of business, or were consumed by competitors:

- Control Data Corp.
- Cray (eaten by SGI, then TERA, now reborn)
- Convex
- DEC (Digital Equipment Corp., eaten by Compaq, in turn by HP)
- Kendall Square Research
- Maspar
- Meiko
- SGI (recently twice deceased)
- Sun (recently absorbed by Oracle)
- Thinking Machines Corp.

And the Sword ...

The tool by which these companies were driven out of business, of course, is the commodity CPU of Intel (and its primary competitor, AMD).

- Commodity CPUs are as cheap as dirt (well, ok, sand) compared to their proprietary foes (be they RISC or vector).
- Commodity networking has taken flight as well, making it also rather hard to compete on the interconnect front.
- The end result of these “clusters” of low-cost systems is that the per-CPU cost can be one (if not two) orders of magnitude less than their proprietary counterparts.

The Top500 List

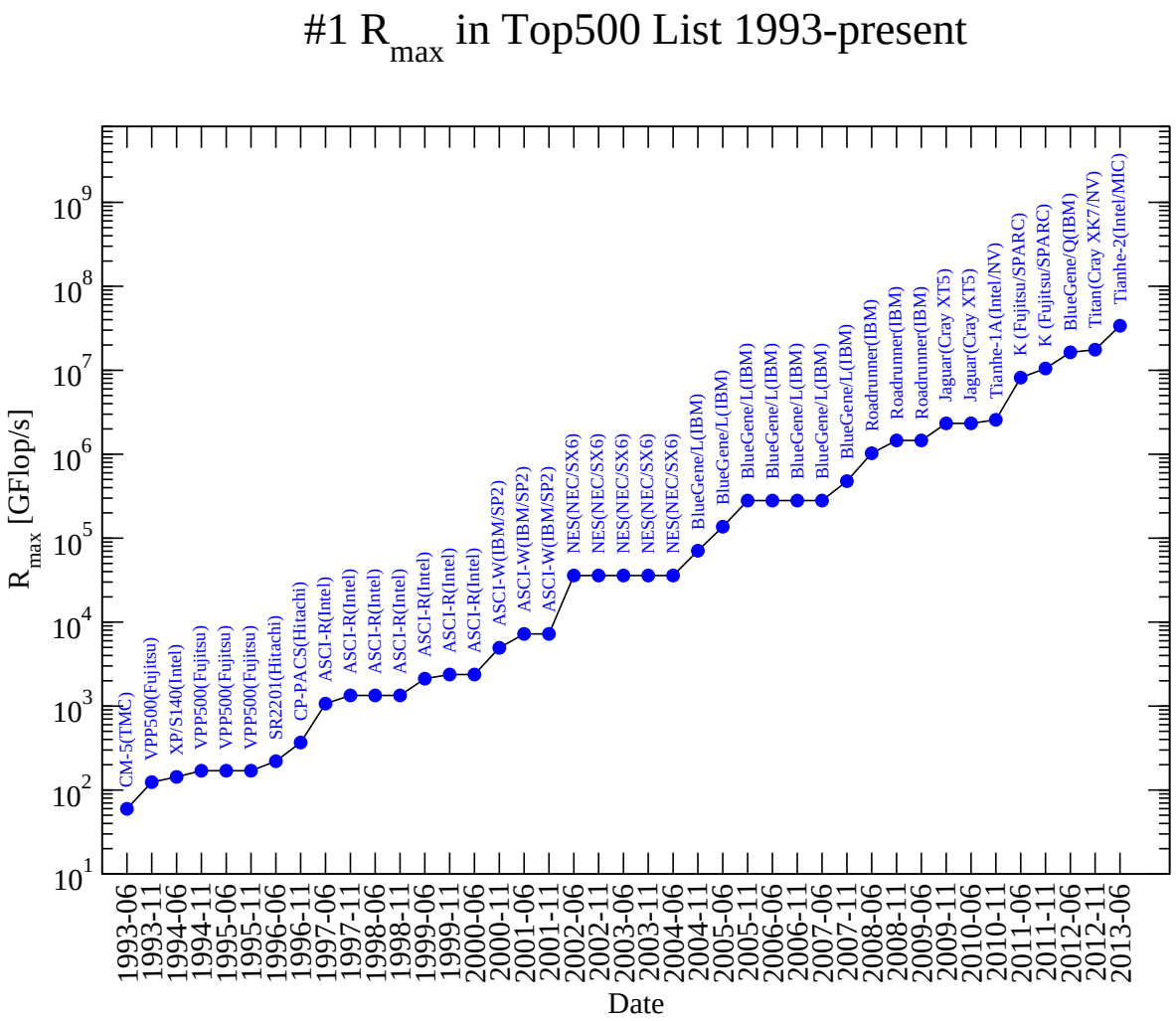
One measure of HPC/supercomputer performance is the **LINPACK** benchmark, which has been used since 1993 to rank the world's most powerful supercomputers.

`www.top500.org`

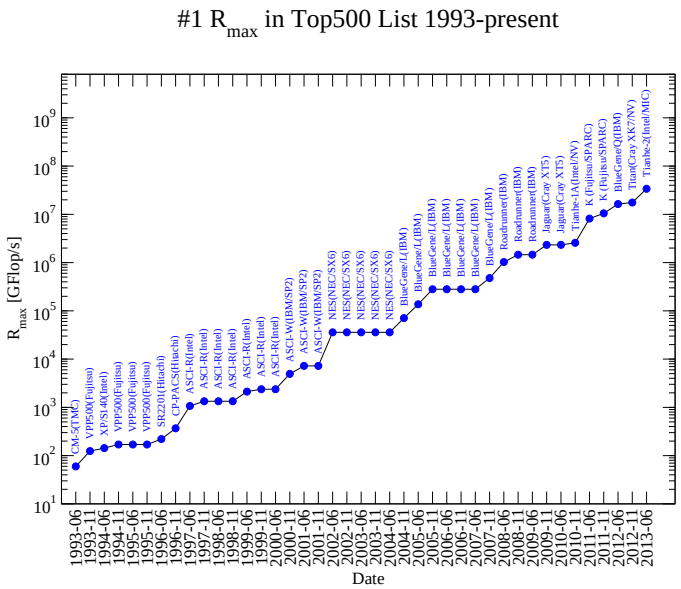
The LINPACK benchmark was introduced by Jack Dongarra:

- solves a dense system of linear equations, $\mathbf{Ax} = \mathbf{b}$
- for different problem sizes N, get maximum achieved performance R_{max} for the problem size N_{max}
- must conform to the standard operation count for LU factorization with partial pivoting (forbids fast matrix multiply method like "Strassen's Method"), i.e. $2/3N^3 + \mathcal{O}(N^2)$
- a version is available for distributed memory machines (also known as the "high-performance Linpack benchmark")

R_{max} in the Top500

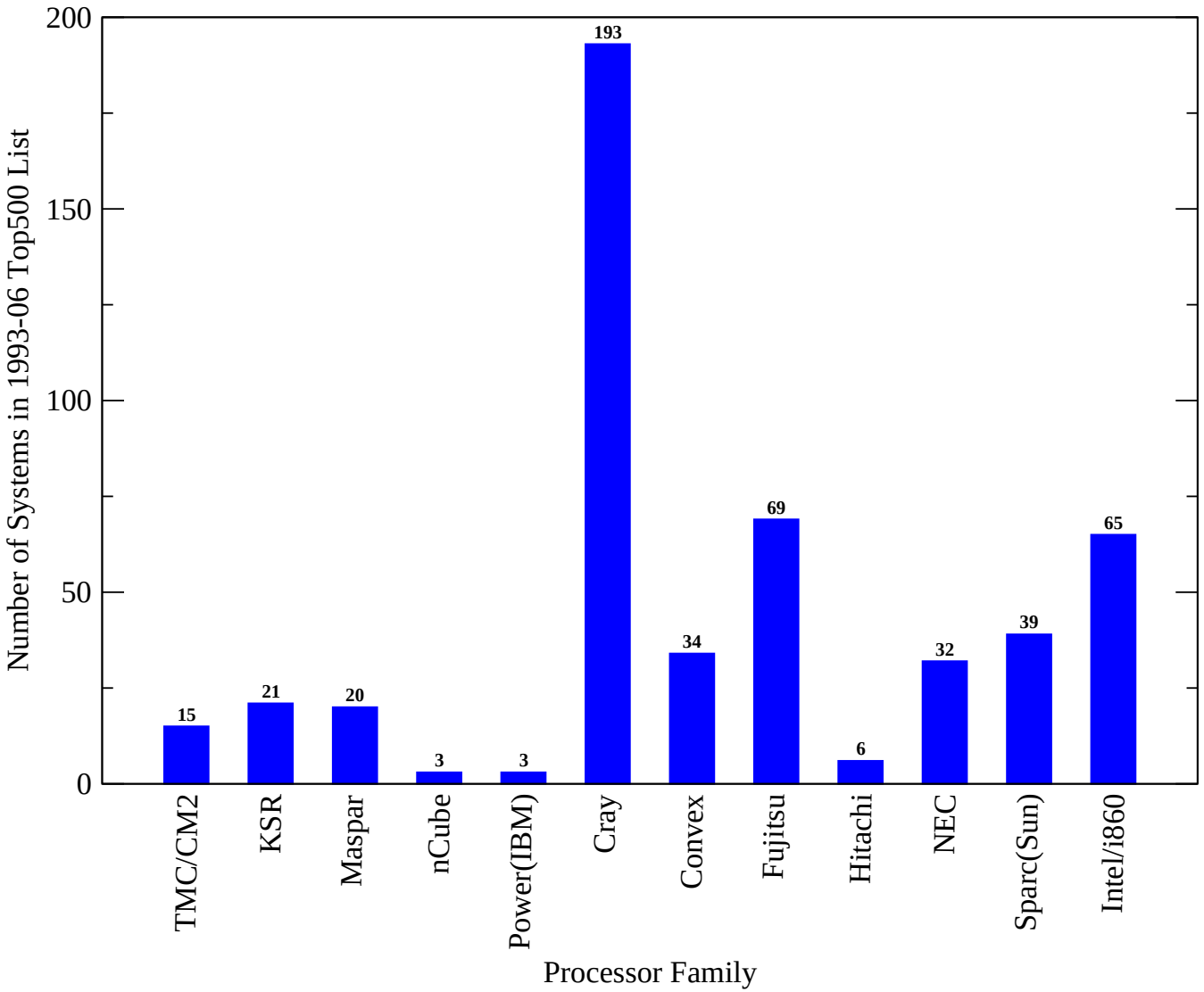


Trends in R_{max}

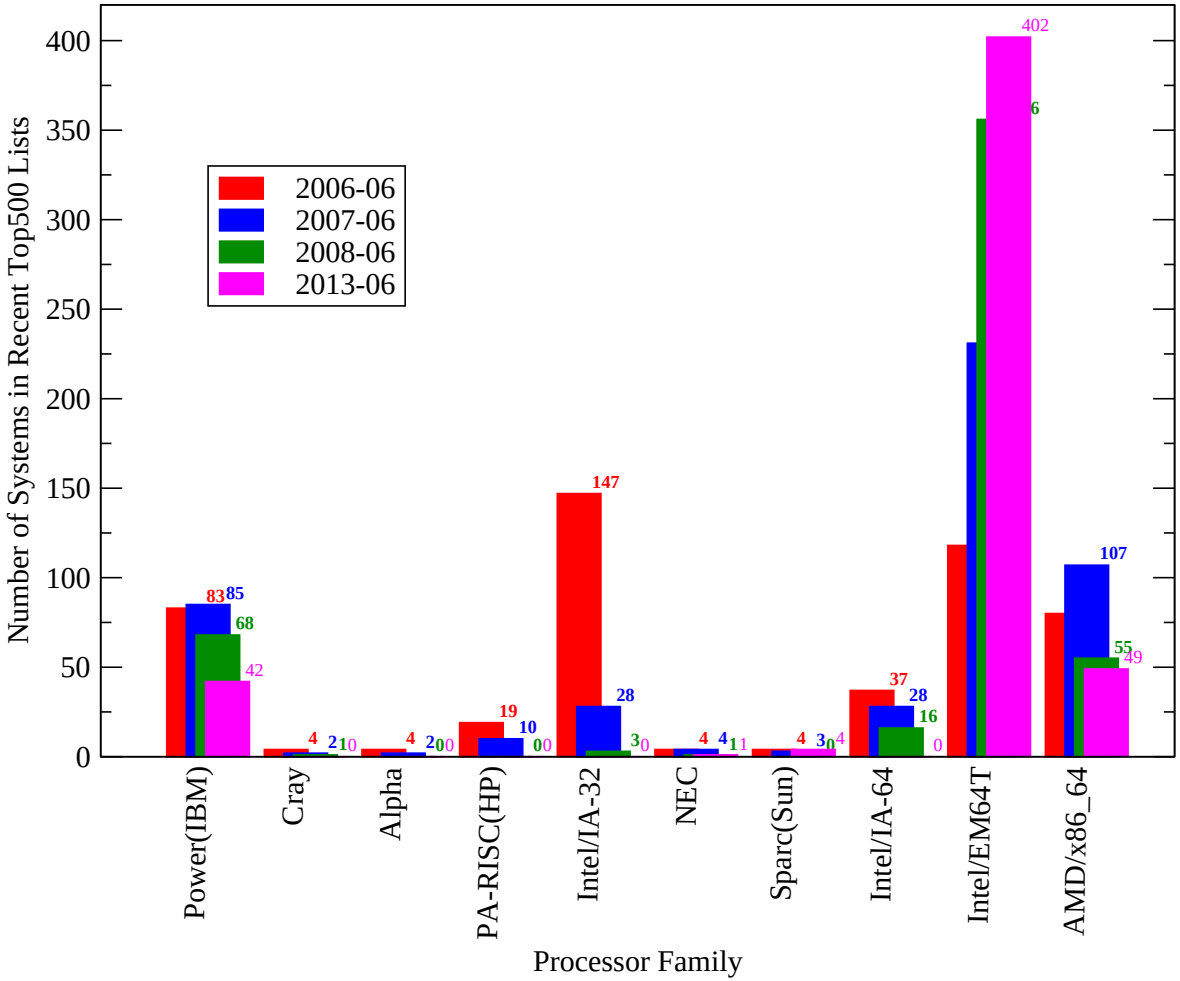


- One can certainly argue with using this single benchmark (LINPACK) as **the** supercomputer ranking, but it is quite instructive as to historical trends in HPC.
- Note the growth in R_{max}
- What about the trends in processor architecture?

Trends in Processor Family



Trends in Processor Family



Current Processor Trends

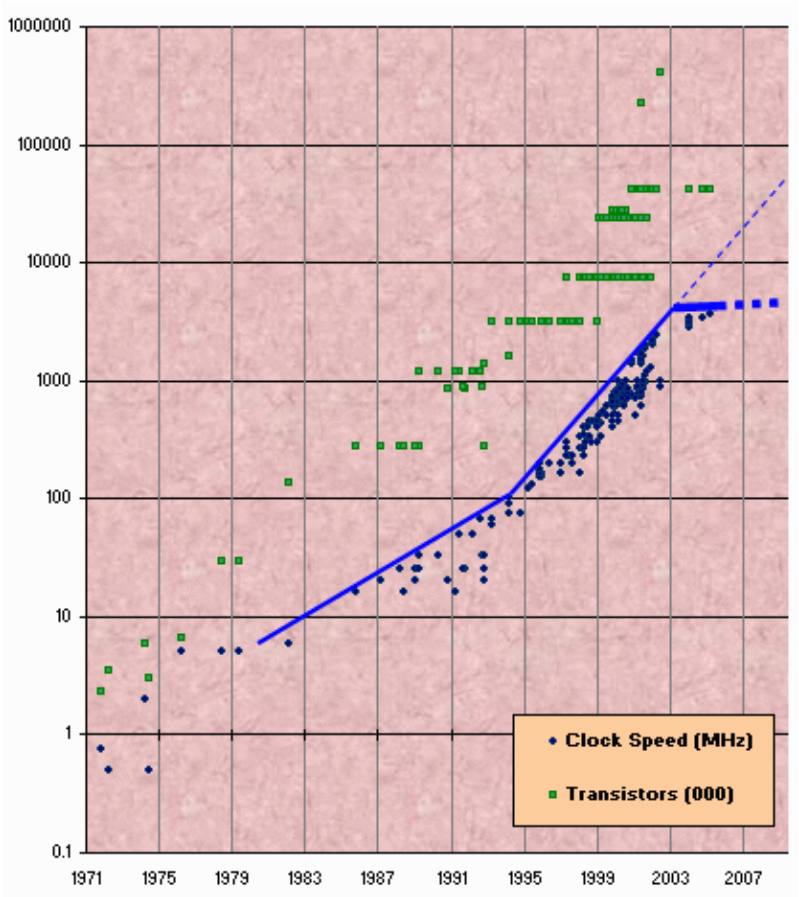
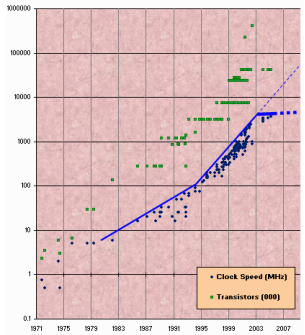


Figure: Intel processors according to clock frequency and transistor count. (H. Sutter, Dr. Dobb's Journal **30**(3), March 2005).



- Processor frequency has reached a plateau due to power and design requirements
- Moore's "Law" of transistor density doubling every 18-24 months is still in effect, however
- More transistor logic is being devoted to innovations in multi-core processors
- Will become significant software issue - how do you harness the computational power in all of those cores?

The Power Envelope

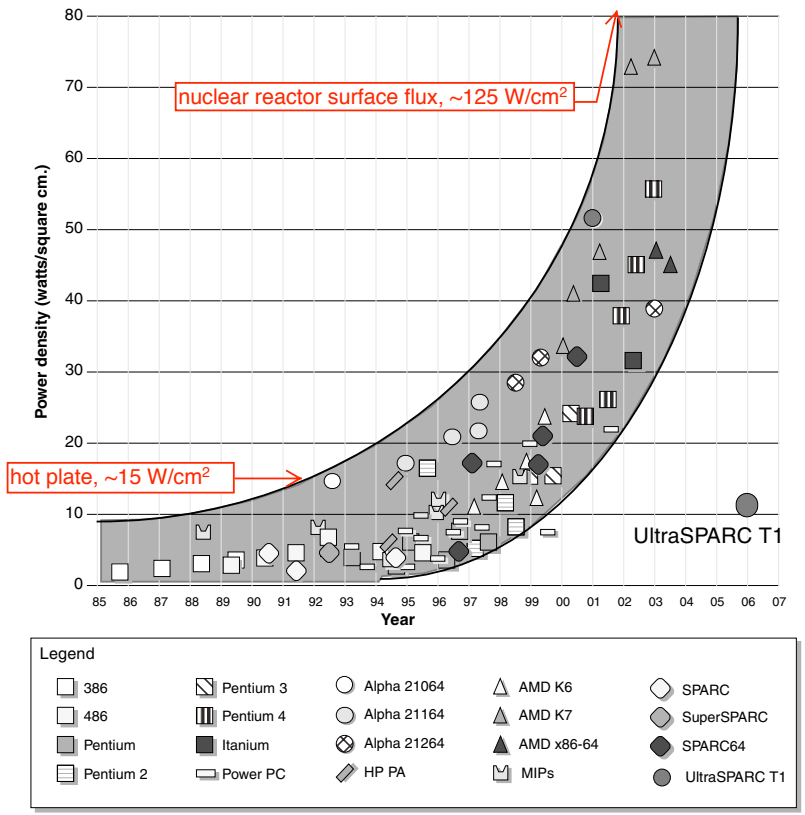
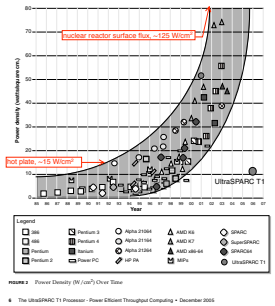


FIGURE 2 Power Density (W/cm^2) Over Time

6 The UltraSPARC T1 Processor - Power Efficient Throughput Computing • December 2005

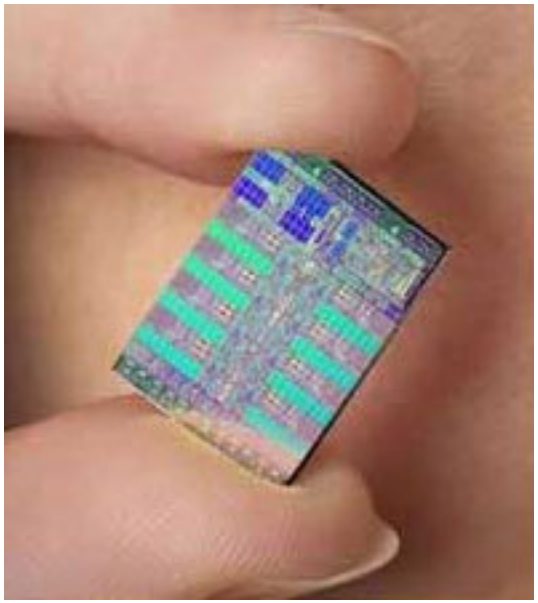
Figure: Processor power expenditures, courtesy Sun Microsystems.

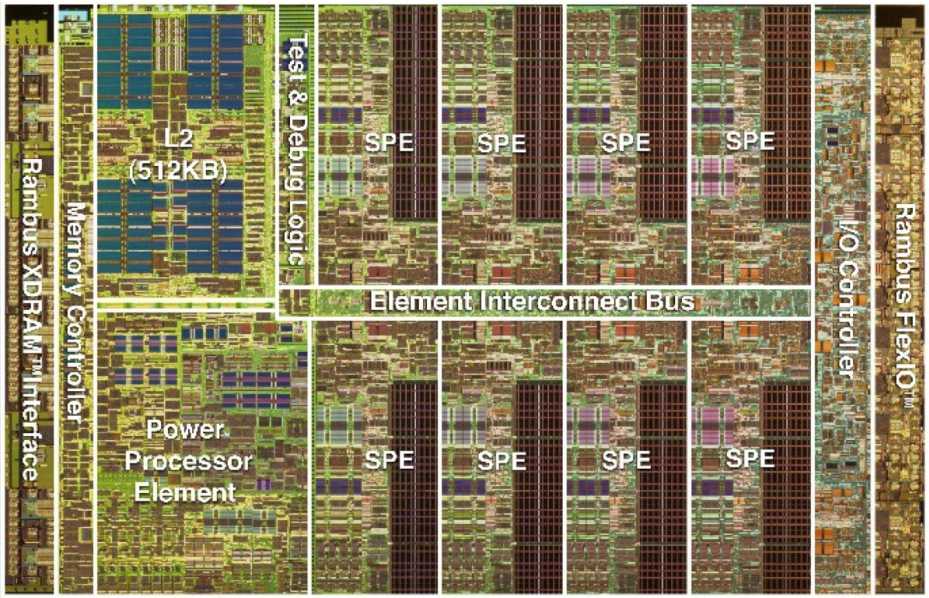
The Power Envelope



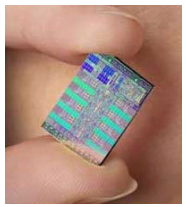
- Around 2003, Intel & AMD both reached close to the limit of air cooling for their processors - why?
- Power in a CPU is proportional to $V^2 fC$:
  - Increasing frequency, f, also requires increasing voltage, V - highly nonlinear
  - Increasing core count will increase capacitance, C, but only linearly
- Behold, the multi-core era is born, but by necessity rather than choice ...

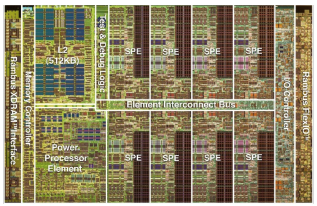
Future Speculation





- Hot trend in HPC - special-purpose processors. Recognize this one?





- Cell single-chip multi-processor (IBM/Sony/Toshiba), debuted in 2005/2006
- 3.2GHz PowerPC RISC as Power Processing Element (PPE)
- 8 Synergistic Processing Elements (SPEs) act as vector processing elements (hardware accelerators)
- Low power, high performance on specialized applications (230 GFlop/s, in single-precision)
- #1 on top500 list in 2008-2009 was a mix of Opterons ($\sim$7000) and Cells ($\sim$13000)

Current Multicore Offerings

Short table of current multi-core processors ...

Manufacturer	Arch	Cores
Sun/Oracle "Niagara" T2	SPARC64	16 (1.4 GHz)
IBM POWER7	POWER	4-8 (3-4.5 GHz)
IBM PowerPC A2	POWER	18 (1.6 GHz)
Intel "Nehalem EP/Westmere"	x86_64	4/6 (1.86-3.2 GHz)
Intel "Nehalem EX/Beckton"	x86_64	4/6/8 (1.86-2.66 GHz)
Intel "Sandy Bridge EP/E5"	x86_64	4/6/8 (1.8-3.6 GHz)
AMD "Shanghai"	x86_64	4 (2.1-3.1 GHz)
AMD "Istanbul"	x86_64	6 (2.1-2.8 GHz)
AMD "Magny-Cours"	x86_64	8/12 (1.8-2.4 GHz)
IBM/Sony/Toshiba Cell	PPC+8xSPE	1+8 (3.2 GHz)
SciCortex	MIPS64	6 (500 MHz)
Nvidia "Tesla" C2050	–	448 (1.15 GHz)

All is Parallel

Well, at least pretty much all of the **hardware** is parallel:

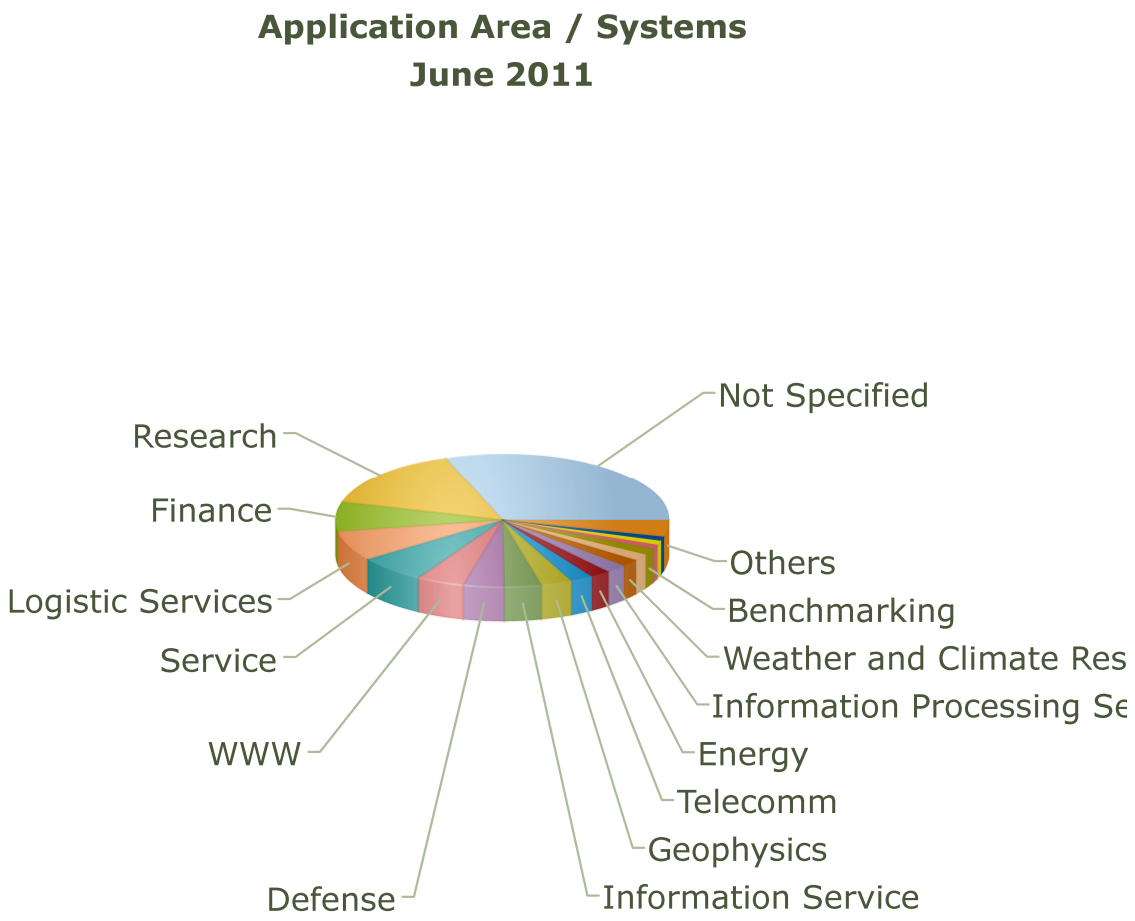
- Right now is it is difficult to buy a computer that has only a single processor - even laptops have multiple cores.
- GPUs have a great many processing elements (Cell has 9, NVIDIA and ATI offerings have hundreds).
- This increasing processor core count trend is going to continue for a while.

What about the **software** that can actually concurrently use all of these hardware resources?

- Software is lagging behind the hardware.
- Some specialized parallel libraries for multicore systems.
- Some APIs for harnessing multiple cores (OpenMP) and multiple machines (MPI).
- Generally the software picture is one of tediously mapping applications to new architectures and machines.

Who Uses HPC?

Users of HPC/parallel computation, as reflected in a recent top500 list:



Why Use Parallel Computation?

Note that, in the modern era, inevitably HPC = Parallel (or concurrent) Computation. The driving forces behind this are pretty simple - the desire is:

- Solve my problem **faster**, i.e. I want the answer **now** (and who doesn't want that?)
- I want to solve a **bigger** problem than I (or anyone else, for that matter) have ever before been able to tackle, and do so in a reasonable (generally reasonable = within a graduate student's time to graduate!)

A Concrete Example

Well, more of a **discrete** example, actually. Let's consider the **gravitational *N*-body problem**.

Example

Using classical gravitation, we have a very simple (but long-ranged) force/potential. For each of N bodies, the resulting force is computed from the other $N-1$ bodies, thereby requiring N^2 force calculations per step. If a galaxy consists of approximately 10^{12} such bodies, and even the best algorithm for computing requires $N \log_2 N$ calculations, that means $\simeq 10^{12} \ln(10^{12}) / \ln(2)$ calculations. If each calculation takes $\simeq 1 \mu\,\mathrm{sec}$, that is $\sim 40 \times 10^6$ seconds *per step*. That is about 1.3 **CPU-years** per step. Ouch!

A "Size" Example

On the other side, suppose that we need to diagonalize (or invert) a dense matrix being used in, say, an eigenproblem derived from some engineering or multiphysics problem.

Example

In this problem, we want to increase the resolution to capture the essential underlying behavior of the physical process being modeled. So we determine that we need a matrix on the order of, say, 400000 elements. Simply to store this matrix, in 64-bit representation, requires $\simeq 1.28 \times 10^{12}$ Bytes of memory, or $\sim$ 1200 GBytes. We could fit this onto a cluster with say, 10^3 nodes, each having 4 GBytes of memory, by **distributing** the matrix across the individual memories of each cluster node.

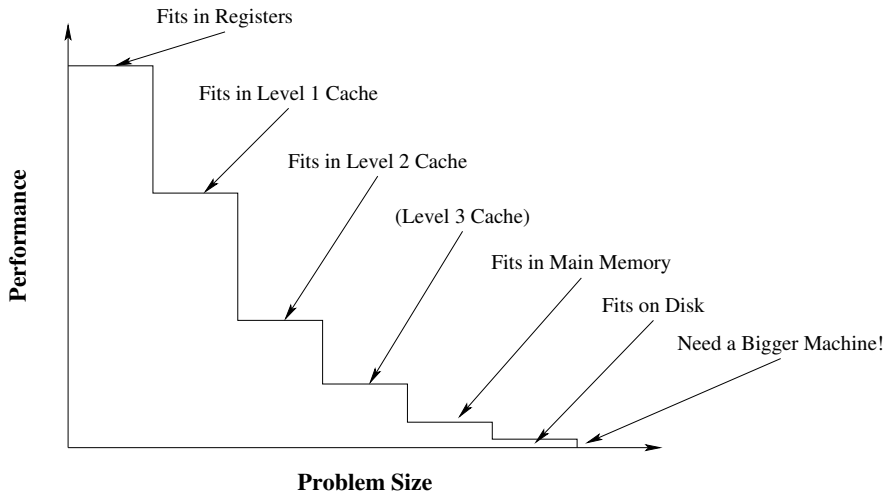
So ...

So the lessons to take from the preceding examples should be clear:

- It is the nature of the research (and commercial) enterprise to always be trying to accomplish **larger** tasks with ever increasing speed, or **efficiency**. For that matter it is human nature.
- These examples, while specific, apply quite generally - do you see the connection between the first example and general molecular modeling? The second example and finite element (or finite difference) approaches to differential equations? How about web searching?

Dictatorship of Data Storage

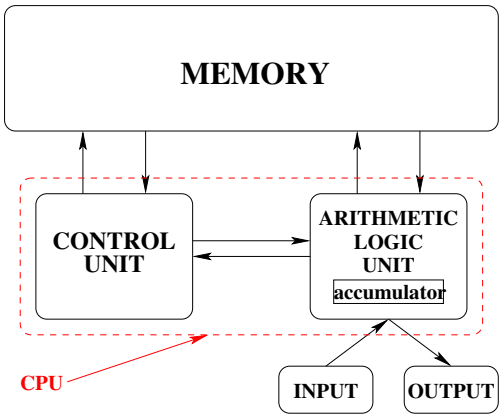
Very strong limitations are imposed by the hierarchy of data/memory storage, schematically:



- Note that the performance scale shown is likely logarithmic.
- Ever-decreasing speed of memory as you get further from the processor(s) is inevitable, often called the "**memory wall**," which we will discuss later in more detail.

What is Old is New Again ...

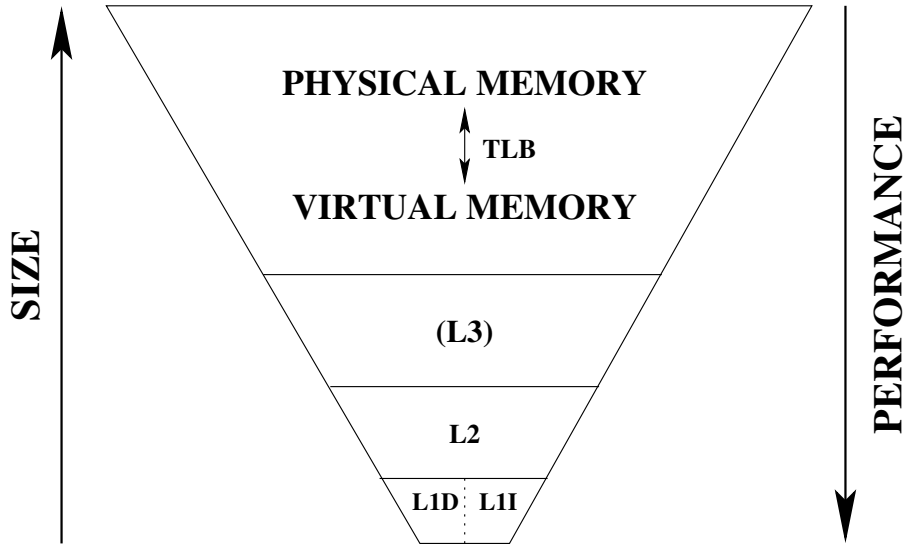
Interestingly enough, computer processors have not changed very much - consider the following picture:



- Look familiar? Pretty much every modern processor follows this design ...
- John von Neumann, the famous polymath, described this concept of the **stored-program** computer in **1946**!
- It follows directly from Alan Turing's concept of the Turing machine from 1936

Cache & Memory Hierarchy

Almost all modern processors try to improve performance by keeping the most oft-used data close to the CPU(s), using a **hierarchy of caches**:



Note in particular that as the caches increase in size, they decrease in performace (bandwidth and latency).

CPU/Memory Imbalance

It has been noted that CPU performance has been increasing at roughly a rate of 80% per year (aside: note Moore's "observation" was for transistor counts), but **memory** performance has lagged behind considerably - perhaps 7% per year[1].

- Has led to a significant imbalance between CPU and memory performance
- Explains the preceding cache hierarchy figure pretty well!
- Also explains why getting even 10% of the theoretical peak performance out of a CPU can be a considerable achievement!

[1] see, for example, Sally A. McKee, "Reflections on the Memory Wall," CF04: *Proceedings of the 1st Conference on Computing Frontiers (Ischia, Italy, 2004), p. 162.*

CPU/DRAM Performance Gap

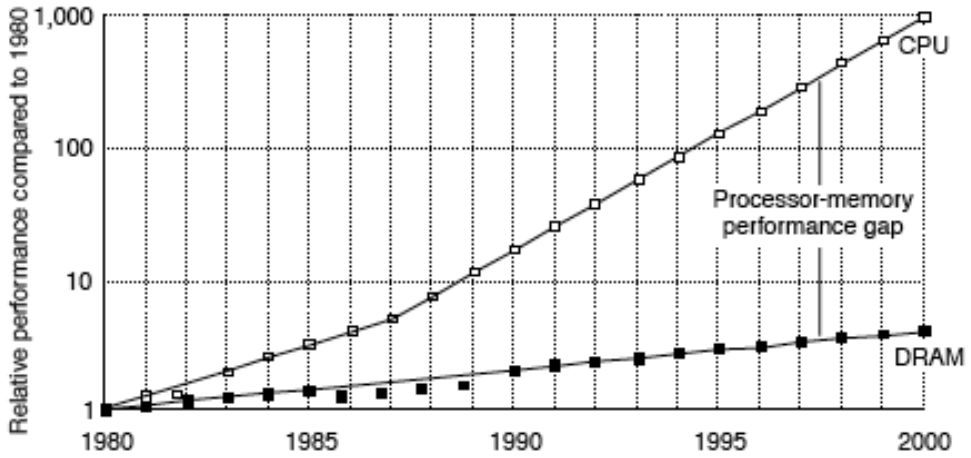


Figure: D. Patterson *et. al.*, "A case for intelligent RAM," IEEE Micro **17**, 34 (1997).

Machine Balance

The notion of **machine balance** (or work/memory ratio) attempts to quantify, in at least a simple way, this distinction between CPU and memory,

$$\rho_{WM} = \frac{\text{number floating-point operations}}{\text{number of memory references}}.$$

- Larger ρ_{WM} is better given the relative slow speed of memory.
- If the maximum bandwidth to memory is bw_M, then a given algorithm has a maximum performance of $\rho_{WM} \times \text{bw}_M$.
- The `STREAM` benchmark was developed to establish a baseline for machine balance (J. D. McCalpin, "Memory Bandwidth and Machine Balance in Current High Performance Computers", IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, December 1995).

Behold the Power of ... Google

Google is a fine example of the power of parallel computation. Did you already know:

- Google uses very large Linux clusters as their platform for their search engine?
  - Exactly how many is closely guarded, but experts estimate from 100,000 to 300,000 servers.
  - Modifications to the Linux kernel to allow for customization - fault tolerance, redundancy, etc.
- A customized cluster filesystem (the **googleFS**) to accelerate their search and indexing algorithms, and the scalable programming model (**MapReduce**) to go with it

`research.google.com`

Scaling Concepts

By “scaling” we typically mean the relative performance of a parallel vs. serial implementation:

Definition (Scaling): *Speedup Factor, $S(p)$ is given by*

$$S(p) = \frac{Sequential\ execution\ time\ (using\ optimal\ implementation)}{Parallel\ execution\ time\ using\ \mathbf{p}\ processors}$$

so, in the ideal case, $S(p) = p$.

Parallel Efficiency

Using τ_S as the (best) sequential execution time, we note that

$$\begin{aligned} S(p) &= \frac{\tau_S}{\tau_p(p)}, \\ &\leq p, \end{aligned}$$

for a lower bound, and the parallel efficiency is given by

$$\begin{aligned} \mathcal{E}(p) &= \frac{S(p)}{p}, \\ &= \frac{\tau_S}{p \times \tau_p}. \end{aligned}$$

Inherent Limitations in Parallel Speedup

Limitations on the maximum speedup:

- Fraction of the code, f, can not be made to execute in parallel
- Parallel overhead (communication, duplication costs)
- Using this *serial fraction*, f, we can note that

$$\tau_p \geq f\tau_S + (1 - f) * \tau_S/p,$$

Amdahl's Law

This simplification for τ_p leads directly to *Amdahl's Law*,
Definition (Amdahl's Law):

$$S(p) \leq \frac{\tau_S}{f\tau_S + (1-f) * \tau_S/p},$$
$$\leq \frac{p}{1 + f(p-1)}.$$

G. M. Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," AFIPS Conference Proceedings **30** (AFIPS Press, Atlantic City, NJ) 483-485, 1967.

Implications of Amdahl's Law

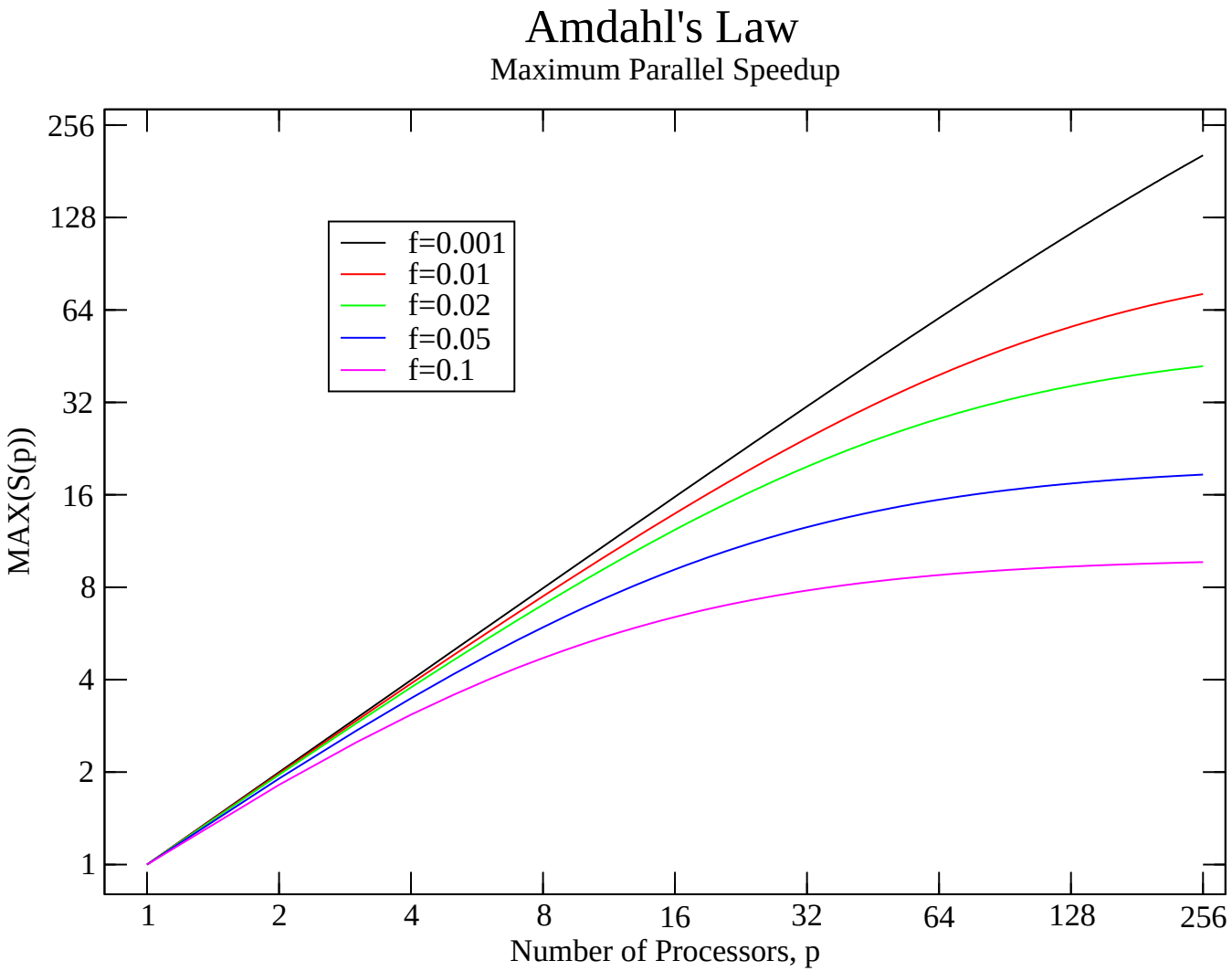
The implications of Amdahl's law are pretty straightforward:

- The limit as $p \to \infty$,

$$\lim_{p \to \infty} S(p) \leq \frac{1}{f}.$$

- If the serial fraction is 5%, the maximum parallel speedup is only 20.

Implications of Amdahl's Law (cont'd)



A Practical Example

Let $\tau_p = \tau_{\text{comm}} + \tau_{\text{comp}}$, where τ_{comm} is the time spent in communication between parallel processes (unavoidable overhead) and τ_{comp} is the time spent in (parallelizable) computation.

$$\tau_1 \leq p\tau_{\text{comp}},$$

and

$$\begin{aligned} S(p) &= \frac{\tau_1}{\tau_p}, \\ &\leq \frac{p\tau_{\text{comp}}}{\tau_{\text{comm}} + \tau_{\text{comp}}}, \\ &\leq p\left(1 + \tau_{\text{comm}}/\tau_{\text{comp}}\right)^{-1} \end{aligned}$$

The point here is that it is critical to minimize communication time to time spent doing computation (recurring theme).

Defeating Amdahl's Law

There are ways to work around the serious implications of Amdahl's law:

- We assumed that the problem size was *fixed*, which is (very) often not the case.
- Now consider a case where the problem size is allowed to vary
- Assume that now the problem size is fixed such that τ_p is held constant

Gustafson's Law

Now let τ_p be held constant as p is increased,

$$\begin{aligned} S_s(p) &= \tau_S/\tau_p, \\ &= (f\tau_S + (1-f)\tau_S)/\tau_p, \\ &= (f\tau_S + (1-f)\tau_S)/(f\tau_S + (1-f)\tau_S/p), \\ &= p/(1-(1-p)f), \\ &\simeq p + p(1-p)f \ldots . \end{aligned}$$

Another way of looking at this is that the serial fraction becomes negligible as the problem size is scaled. Actually that is a pretty good definition of a scalable code ...

J. L. Gustafson, "Reevaluating Amdahl's Law," Comm. ACM **31**(5), 532-533 (1988).

Scalability

Definition

(Scalable): An algorithm is scalable if there is a minimal nonzero efficiency as $p \to \infty$ and the problem size is allowed to take on any value

I like this (equivalent) one better:

Definition

(Scalable): For a scalable code the sequential fraction becomes negligible as the problem size (and the number of processors) grows

Flynn's Taxonomy

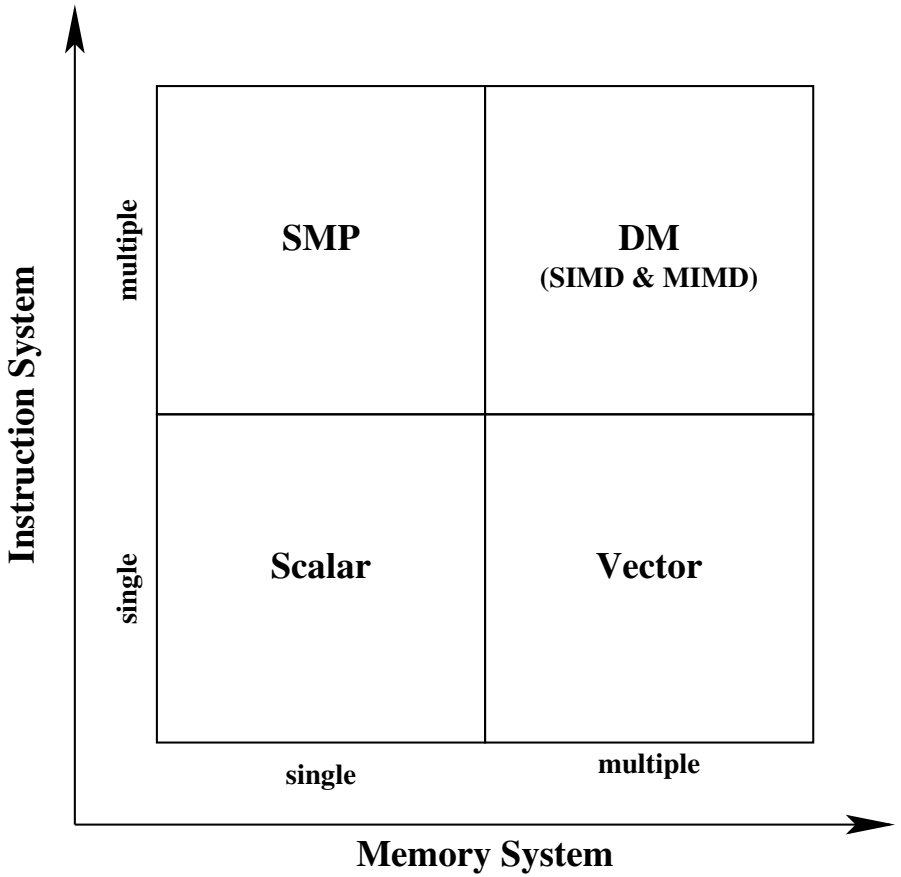
Based on the flow of information (both data and instructions) in a computer:

- **SISD** Single Instruction, Single Data
  quite old-fashioned
- **SIMD** Single Instruction, Multiple Data
  data parallel
- **MISD** Multiple Instruction, Single Data
  never implemented
- **MIMD** Multiple Instruction, Multiple Data
  task parallel

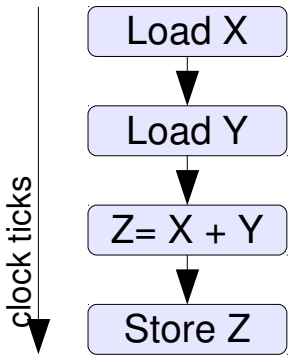
Many taxonomies are possible ...

A More Intuitive "Parallel" Taxonomy

A somewhat more intuitive illustration of parallel taxonomy:



SISD

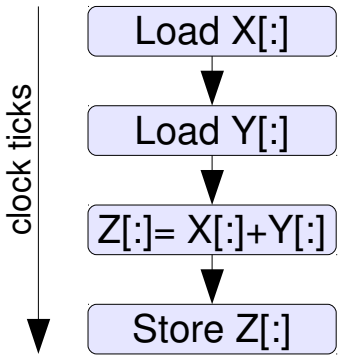


SISD

- sequential, deterministic

- Sequential (non-parallel) instruction flow
- Predictable and deterministic

SIMD

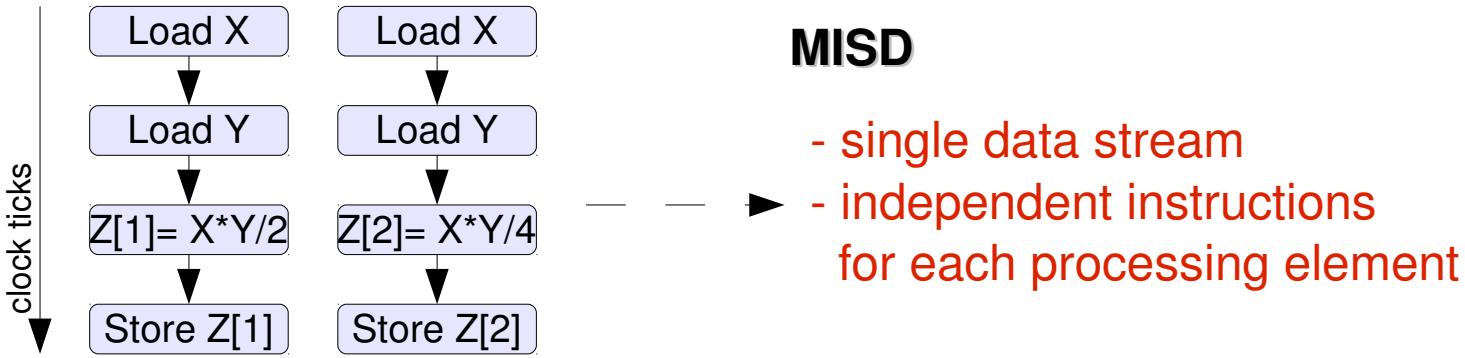


SIMD

- type of parallel computation
- may be implemented as vector pipeline and/or processor arrays

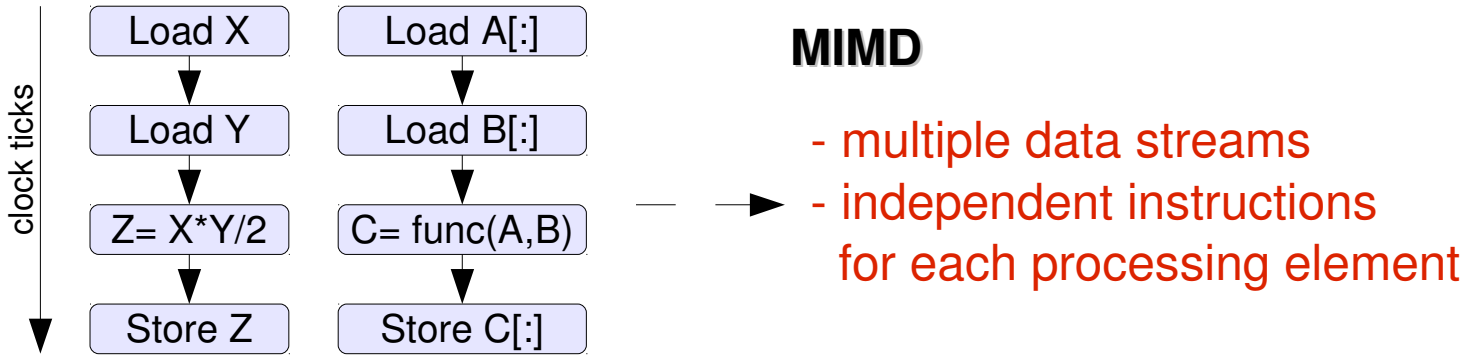
- Single (typically assembly) instruction operates on different data in a given clock tick.
- Requires predicatble data access patterns - "vectors" that are contiguous in memory.
- Almost all modern processors use some form - for Intel, SSE (Streaming SIMD Extensions), AVX (Advanced Vector Extensions)

MISD



- Multiple instructions applied independently on same data.
- “Theoretical” rather than practical architecture - few implementations

MIMD



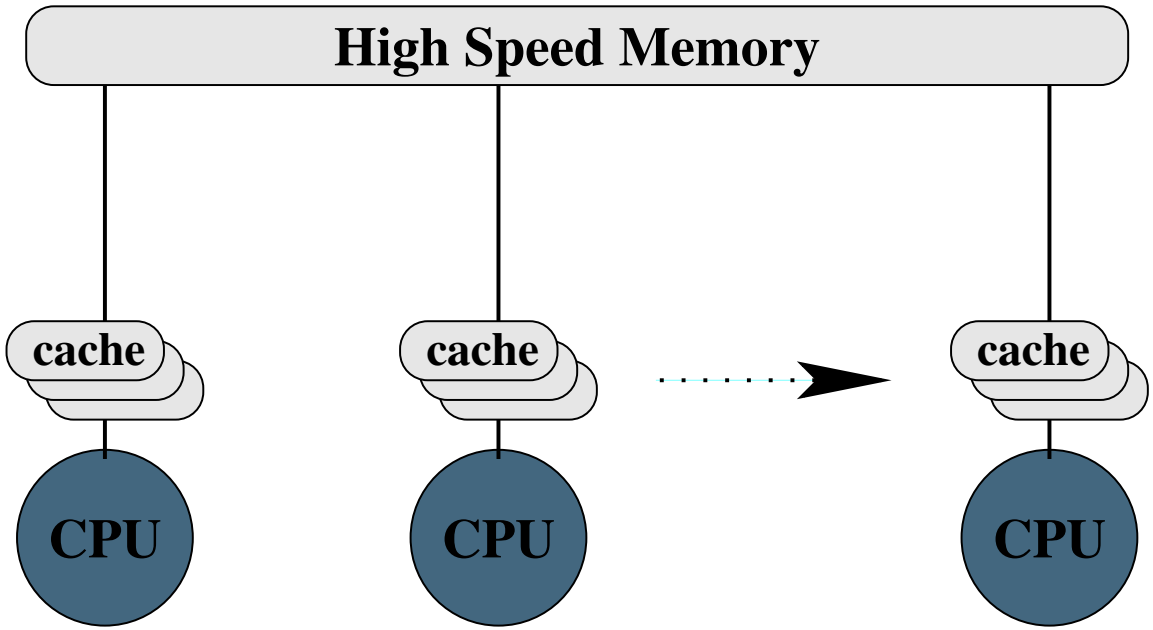
- Multiple instructions applied independently on different data.
- Very flexible - can be asynchronous, non-deterministic
- Most modern supercomputers follow MIMD design, albeit with SIMD components/sub-systems

Data/Task Parallel

Where is the parallelism in your code?

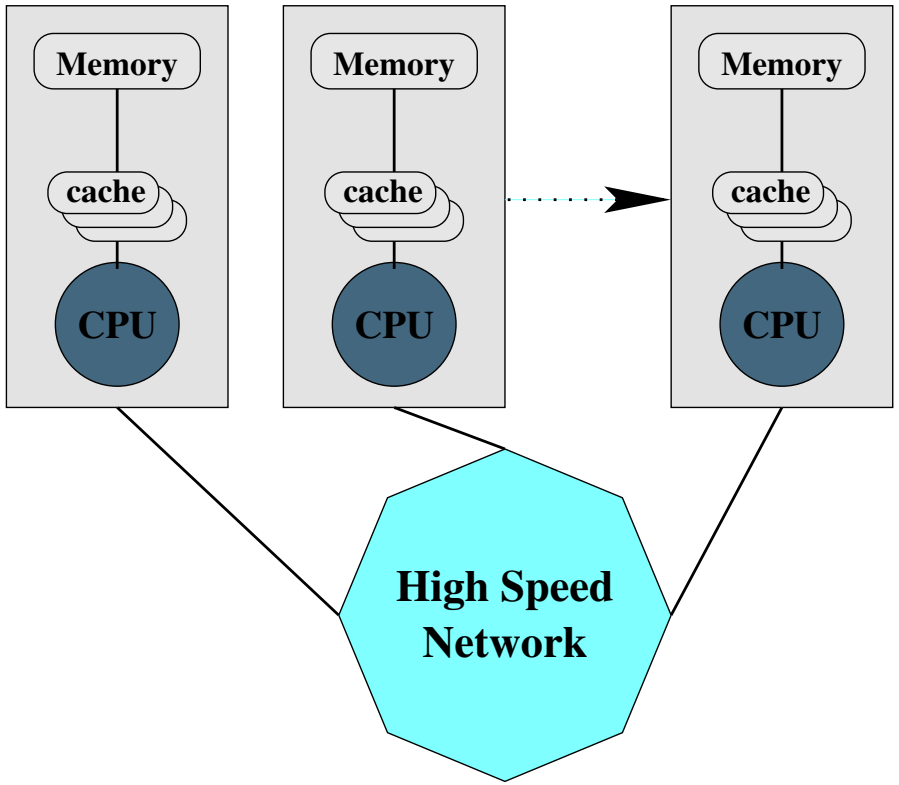
- “data parallel” - the code has data that can be distributed over and acted upon by multiple processing units. (think OpenMP and HPF)
- “task parallel” - the tasks performed by the code can be distributed over multiple processors, often with some need for inter-process communication. (think MPI, PVM, etc.)

SMP



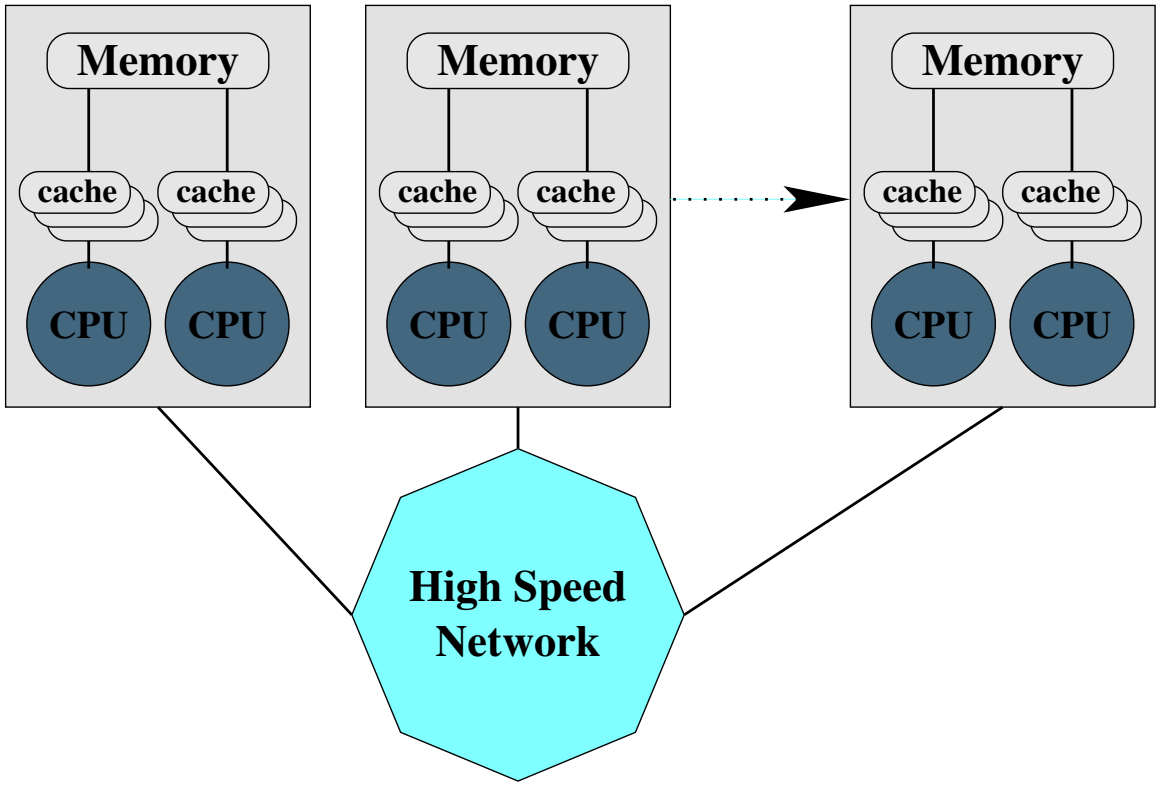
- Each processor has equal access to pool of *shared* global memory (contrast with non-uniform, or NUMA)
- Cache *coherency* is a serious issue
- Quickly limits scalability due to bottlenecks
- Examples: Sun Enterprise Series,Intel/AMD x86/x86_64 bus-based SMPs (small), IBM POWER Nodes

Distributed Memory



- Single processors with a hierarchy of memory interconnected through an external network
- Examples: rather uncommon now - older clusters of uniprocessor nodes.

DSM



- Some processors have truly shared memory, rest accessed in a non-uniform (NUMA, or non-uniform memory access) way through network
- Examples: clusters of SMPs, SGI Origin/Altix Series (OS makes it look like SM).

Data Parallel

- Also known as fine-grained parallel.
- Same set of instructions runs simultaneously on different pieces of data.
- Typical example - parallel execution of **DO/for** loops in FORTRAN/C codes.
- Examples of directive-based data-parallel APIs: **HPF**, **OpenMP**.

Task Parallel

- Also known as coarse-grained parallel.
- Multiple code segments are run concurrently on different processors.
- Generally only works well when the code fragments are independent (otherwise the communication cost may be prohibitive).
- Typical example - **subroutine** calls in FORTRAN codes.
- Examples of message-passing task parallel APIs: **PVM**, **MPI**.

SIMD

Processors that incorporate SIMD vector-like features are widespread:

- packs N (often a power of 2) operations into a single instruction
- included in processors from Intel and AMD (streaming SIMD extensions, or SSE)
- allow the processor to speed calculation by extensive prefetching of multiple pieces of data which will be subjected to the same set of operations.

SPMD vs. MPMD

All the architectures that we will consider follow the MIMD design model, and we can denote our general approach as **M**ultiple **P**rogram, **M**ultiple **D**ata. A typical example consists of two programs in a master-worker arrangement. SPMD is another such structure, in which the same code is executed by all processes, but on different data (e.g. in the master-worker example, the code would have sections for the master, and sections for the workers).

Vector Processing

A vector processor is one in which entire vectors can be added together with a single (vector) instruction, rather than simply two elements in the case of scalar processors. This approach can be quite advantageous:

- Vector processors may have to fetch and decode far fewer instructions (less overhead)
- More (contiguous in memory) data can be moved simultaneously
- Keeps processor fed with data rather than waiting on memory operations

Typical Cluster Compute Nodes

HPC clusters can be built out of just about any type of system, depending on your budget. Some common (and a few less common) options:

- Commodity PCs (workstations) - typically networked together using Ethernet
- Commodity Servers (2-12 processor cores/server) - often with (advanced) third party networks in addition to, or in place of, Ethernet

Example

Example: CCR's **U2** has 256 nodes with 8x 2.26 GHz Nehalem cores/server, 372 nodes with 12x 2.40GHz Westmere cores/server and 20 "fat" nodes with 32 cores/server (some Intel, some AMD), all with a QDR Infiniband interconnect for applications and gigabit Ethernet for storage/systems operation.

Typical Cluster Compute Nodes (cont'd)

- "Constellations" of High-End Servers ($\geq$ 16 processors/server)

  *NCAR's **Bluefire** system of 128 32-processor IBM p575 servers (Power 6+) with Infiniband interconnect*

  *NASA's **Columbia** system of 20 512-processor SGI Altix 3700s (Itanium2) with Infiniband interconnect*

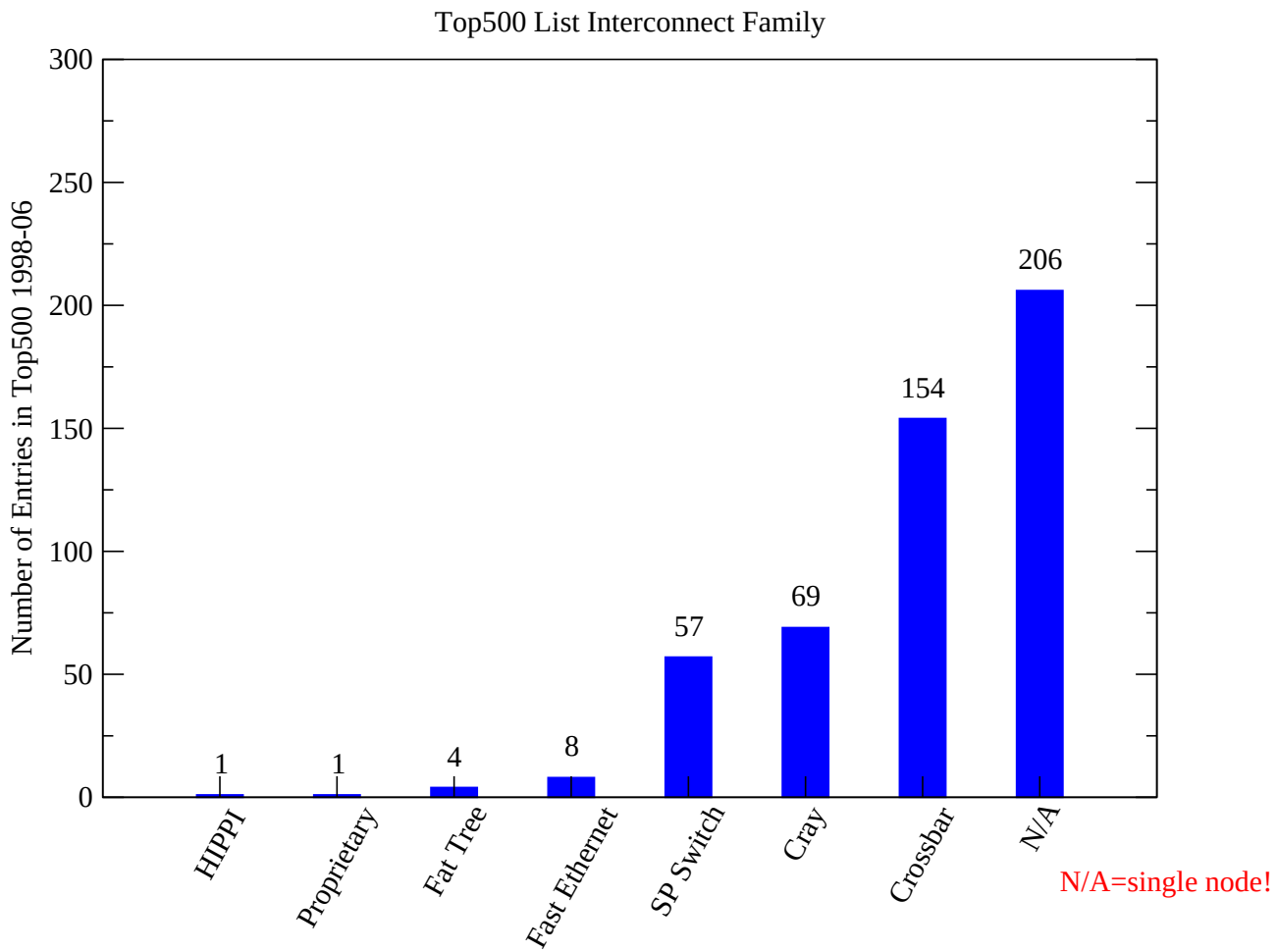
Typical Cluster Networks

By far the most common interconnect for HPC is some form of Ethernet (TCP/IP), but there are other, more highly performing, options:

- **Myrinet**, `www.myrinet.com`
  Myrinet is now available at 10Gbit/s, more common at 2Gbit/s.
- **Infiniband**, `www.infinibandta.org`, `www.openib.org`
  SDR IB 4x is 10Gb/s, DDR 8x is 20Gb/s, and QDR 16x is 40Gb/s.
- **Proprietary/Custom**, customized networks stacks that offer low latency and high bandwidth, becoming less common with commodity networks like Ethernet and Infiniband

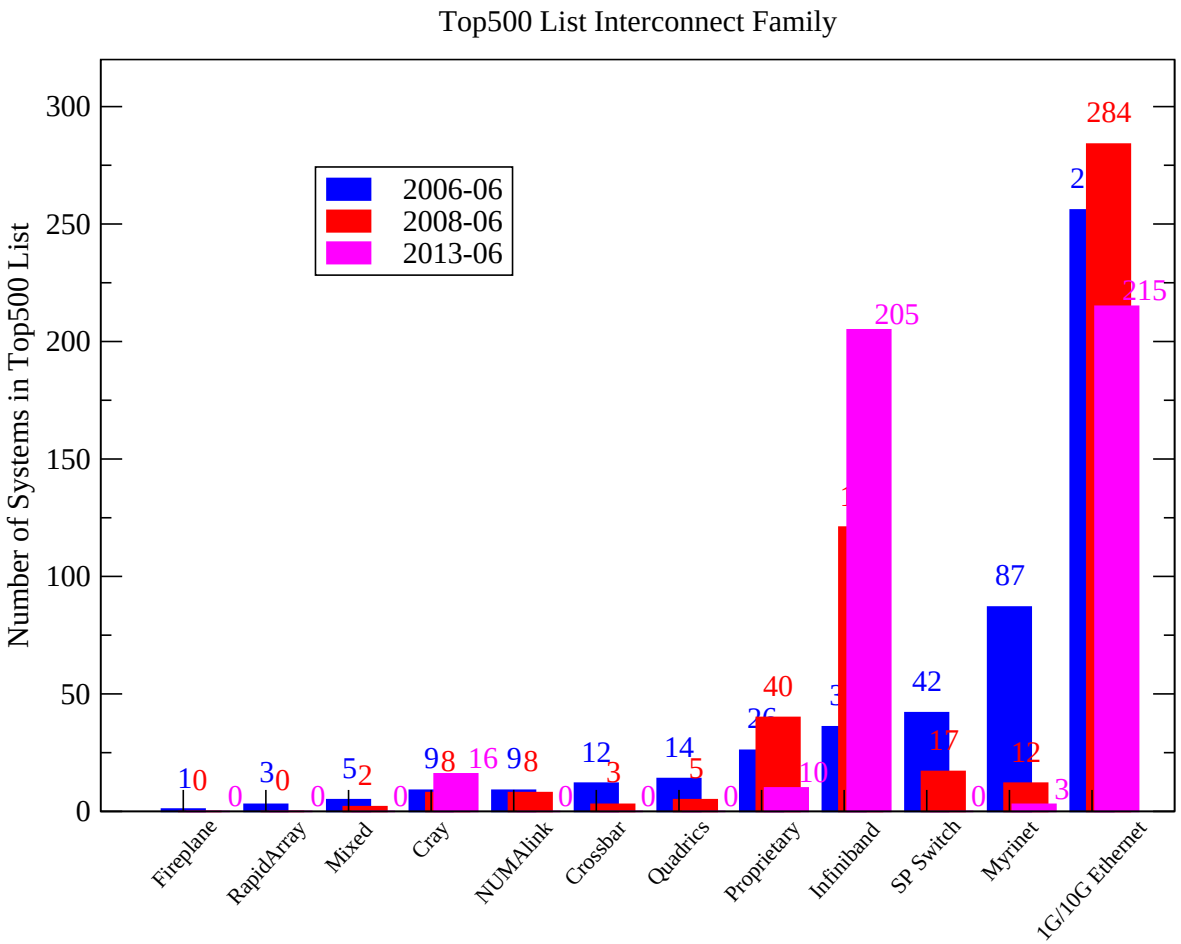
Cluster Network Evolution

Using the Top500 list for analyzing historical trends, we can look at the interconnect family:



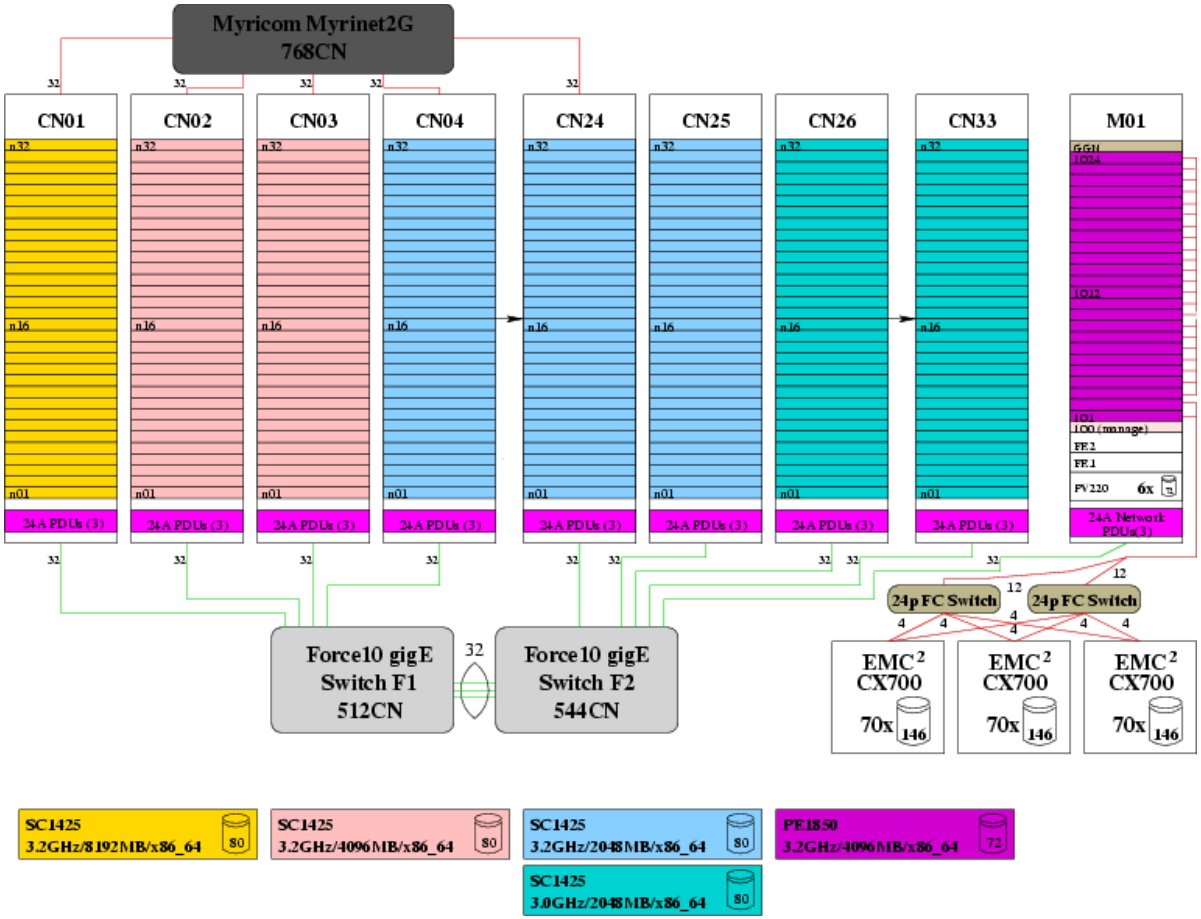
Cluster Network Evolution

... and note the trend now favors the commodity networks:



The Cluster Sandbox

Pictured below is an example of a dedicated compute cluster:

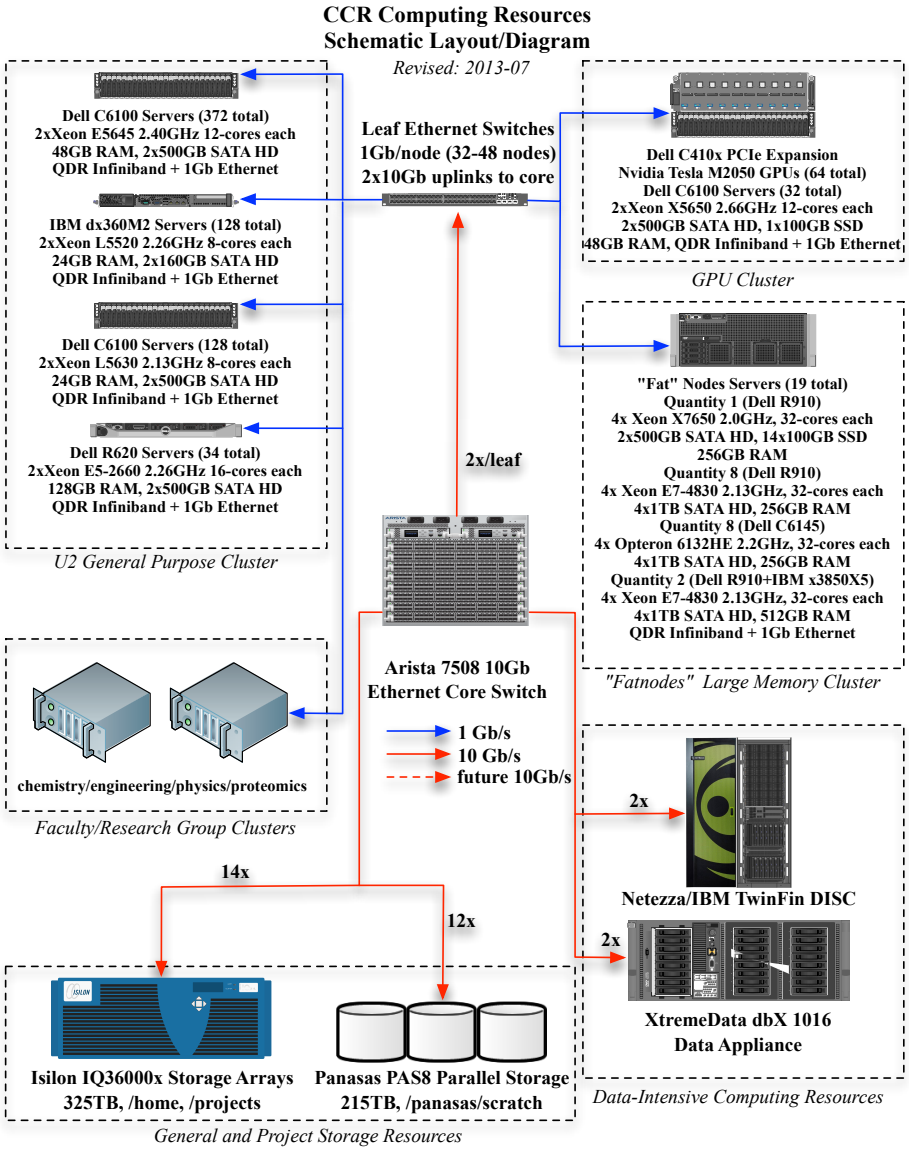


The preceding cluster example was the first incarnation of CCR's **U2**, It placed number 57 on the June 2006 Top500 list, using only the 768 Myrinet connected compute nodes. U2 has been substantially updated since then, but in several stages such that it is now relatively heterogeneous:

http://www.ccr.buffalo.edu/support/research_facilities/u2.html

which is typical for University-based HPC facilities.

Current State of U2



- Current (2012-08) state of **U2** after many upgrades.

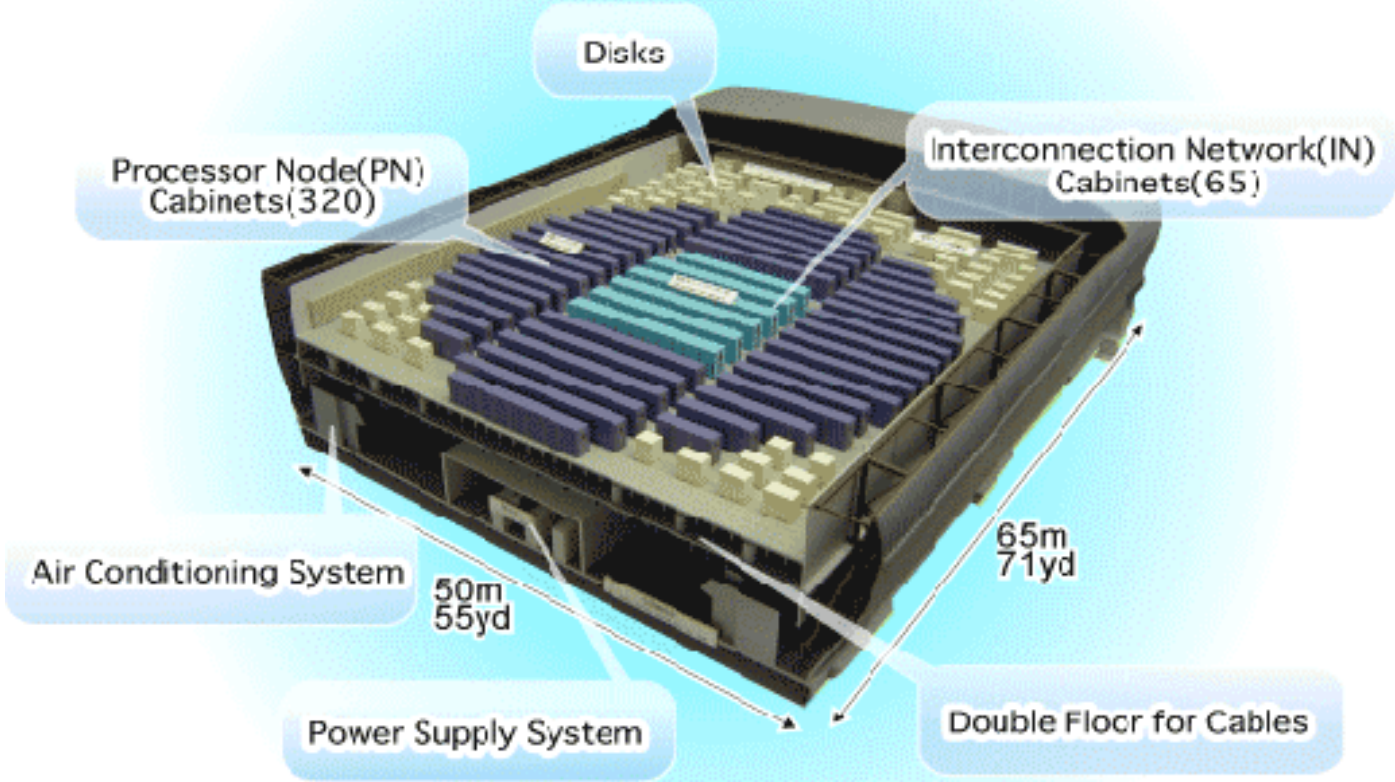
A Very Different Cluster Example

Now let's consider Japan's Earth Simulator, which dominated the Top500 List from 2002-2004:

- 640 8-processor SX-8 (vector) SMPs (peak of 8GFlop/s per processor)
- 10 TB of Memory
- Custom crossbar interconnect
- 700 TB disk + 1.2 PB Mass Storage
- Reportedly consumes about 12MW of power ...

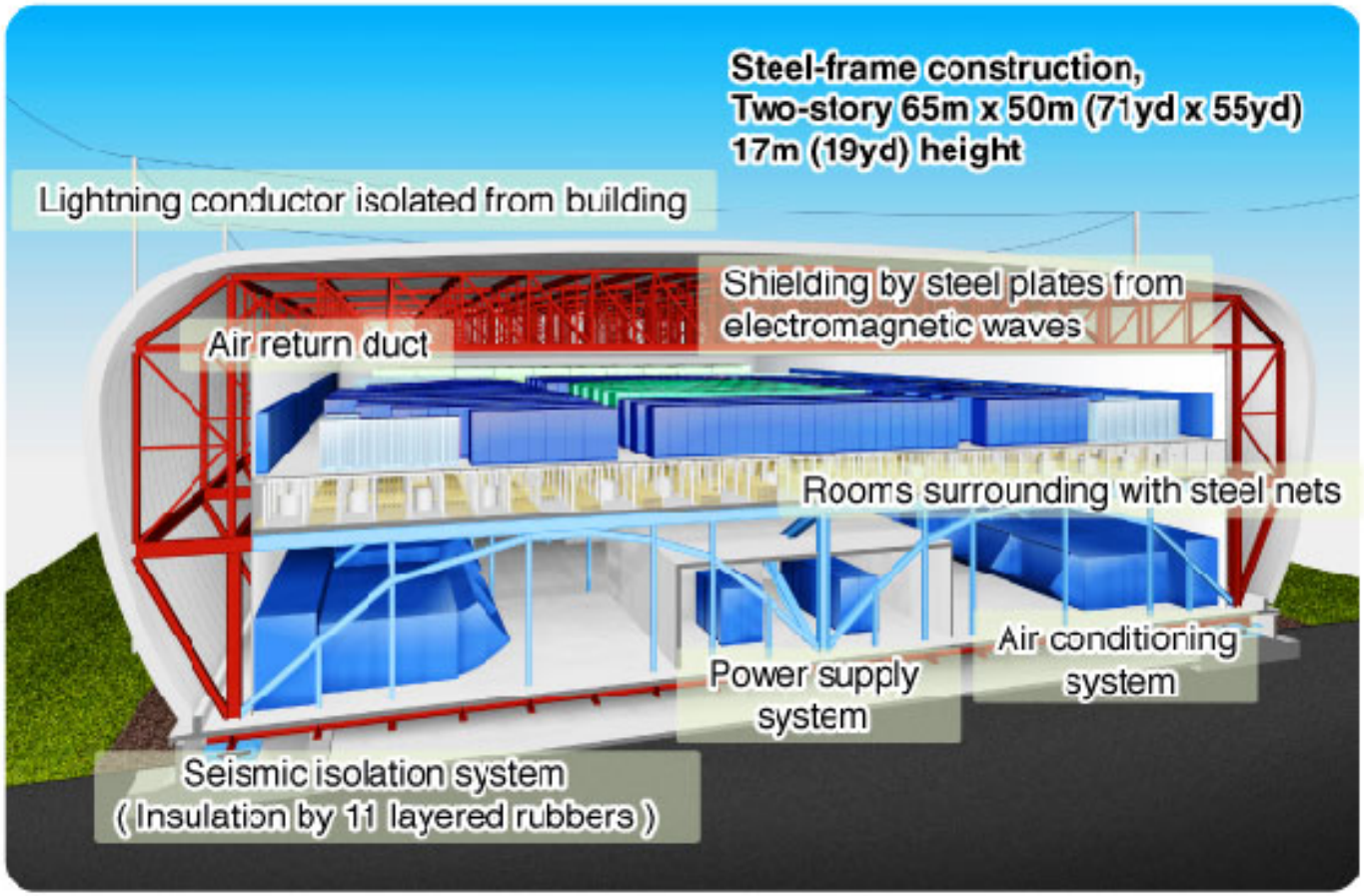
A Very Different Cluster Example (cont'd)

Japan's Earth Simulator:



A Very Different Cluster Example (cont'd)

Japan's Earth Simulator:



IBM Blue Gene

LLNL's BG/L system from 2007-11, and a single rack of BG/L from SC06 (photo on left courtesty IBM, right IBM/Wikipedia).

Highly Engineered "Supercluster"

Now consider IBM's BlueGene, holding 4 of the top 10 spots in the 2008-06 Top500 list:

- Building block is a "node" consisting of 2 700 MHz PowerPC 440 processors, 1024MB RAM
- Nodes configured on **5** different networks
  - 3D Torus (3D mesh where each node connected to 6 nearest-neighbors) for general message passing (well suited for decomposition problems), each link 1.4 Gb/s bidirectional, or 2.1 GB/s aggregate for each node. Worst case hardware latency is 6.4 μs.
  - Collective network (also does some point-to-point), latency of only 2.5 μs for a full 65,536 node BG/L, each link 2.8 Gb/s bidirectional
  - I/O network, with one dedicated I/O node for every 64 compute nodes (gigabit Ethernet)
  - Global interrupt network, can do full system hardware barrier in 1.5 μs (fast Ethernet)
  - Diagnostic & control network (JTAG/IEEE 1149.1)

- OS “kernel” on compute is a customized runtime kernel that provides no scheduling or context switching (said to be written in 5000 lines of C++), one thread/CPU, I/O nodes runs customized Linux kernel
- Native MPI is customized port of MPICH2

Blue Gene/L

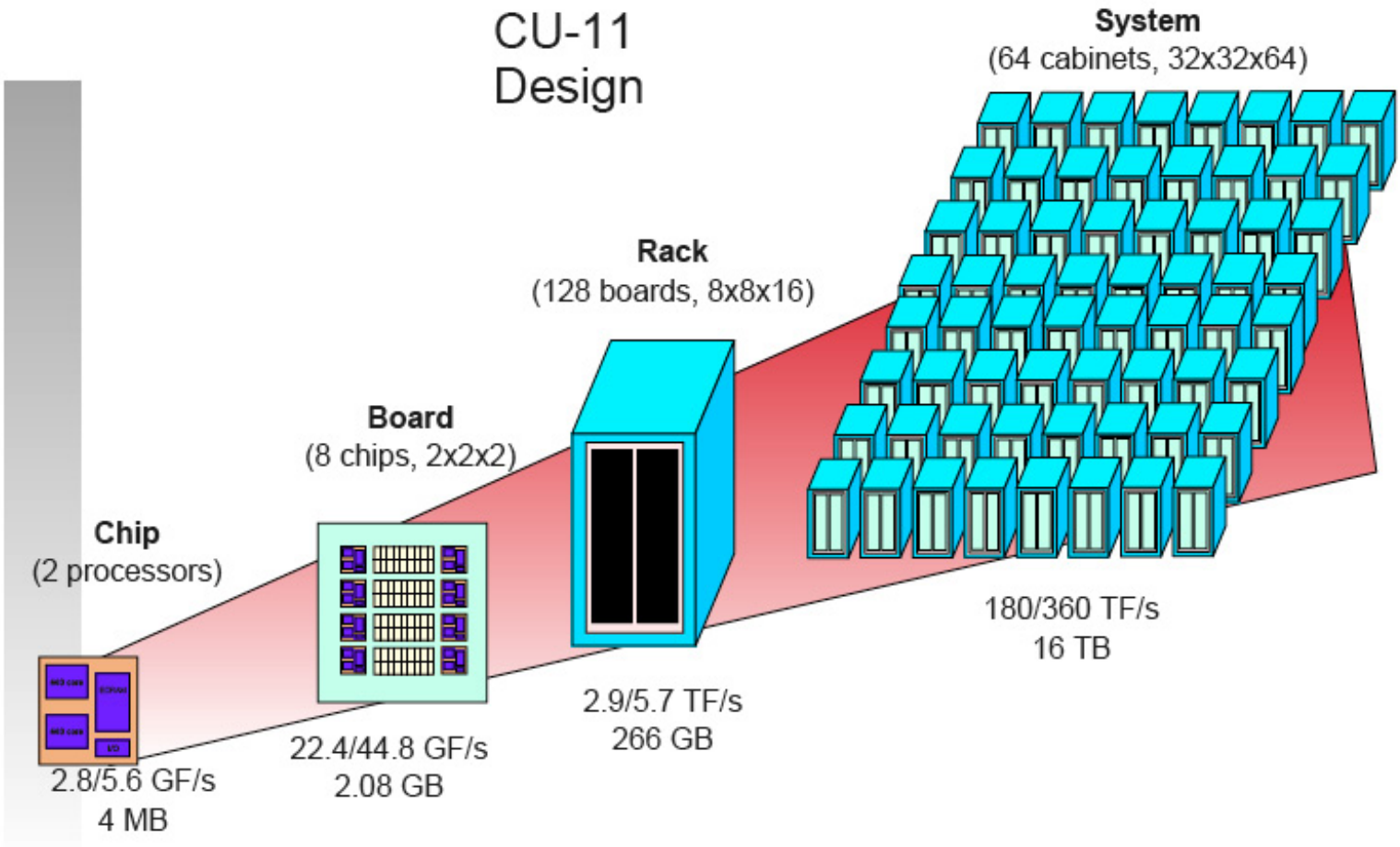


Figure courtesy IBM, total power requirements of 65,536 node system 1.5MW, or 20kW/rack.

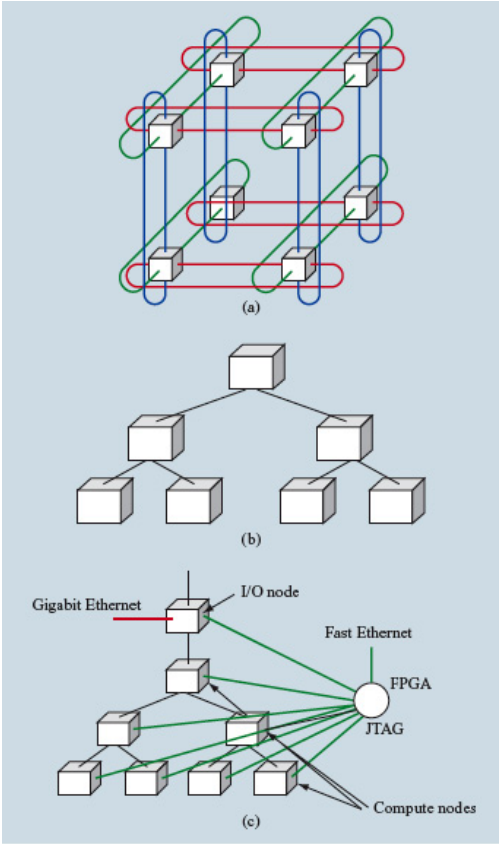


Figure 4

(a) Three-dimensional torus. (b) Global collective network. (c) Blue Gene/L control system network and Gigabit Ethernet networks.

Figure courtesy IBM, IBM J. Res. Devel. **49**, No. 2/3 (2005).

Development of the BG architecture has continued with BG/P (2007, chip upgrade to 4 PPC450 processors/chip), and BG/Q (ETA 2012, 16 cores/chip).

2008/2009 #1 in Top500 - Roadrunner

IBM's Roadrunner (photo courtesy IBM), #1 entry in 2008-06 Top500 list, 1.026 PFlop/s ...

2008/2009 #1 in Top500 - Roadrunner

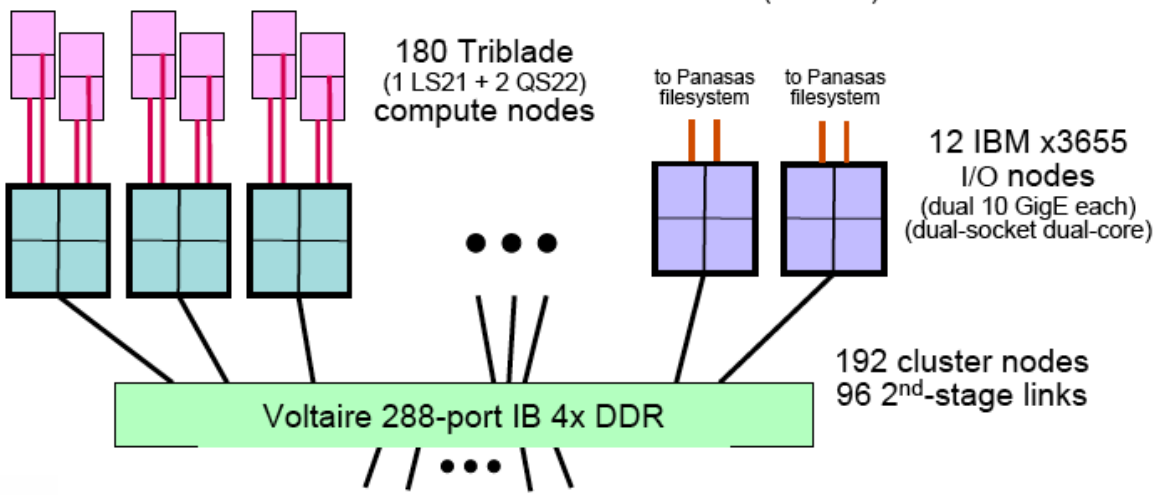
A Connected Unit is a powerful cluster

Connected Unit Specifications:

384 1.8 GHz dual-core Opterons
2.8 TF DP peak Opteron
1.5 TB Opteron memory

720 3.2 GHz Cell eDP chips
73.7 TF DP peak Cell eDP
2.88 TB Cell memory
15.4 TB/s Cell memory BW

192 IB 4X DDR cluster links
768 GB/s aggregate BW (bi-dir)
384 GB/s bi-section BW (bi-dir)
24 10 GigE I/O links on 12 I/O nodes
24 GB/s aggregate I/O BW (uni-dir)
(IB limited)



Los Alamos NATIONAL LABORATORY EST. 1943
Operated by the Los Alamos National Security, LLC for the DOE/NNSA

ASC NNSA IBM

CU from Roadrunner, hybrid Opteron/Cell architecture ...

2008/2009 #1 in Top500 - Roadrunner

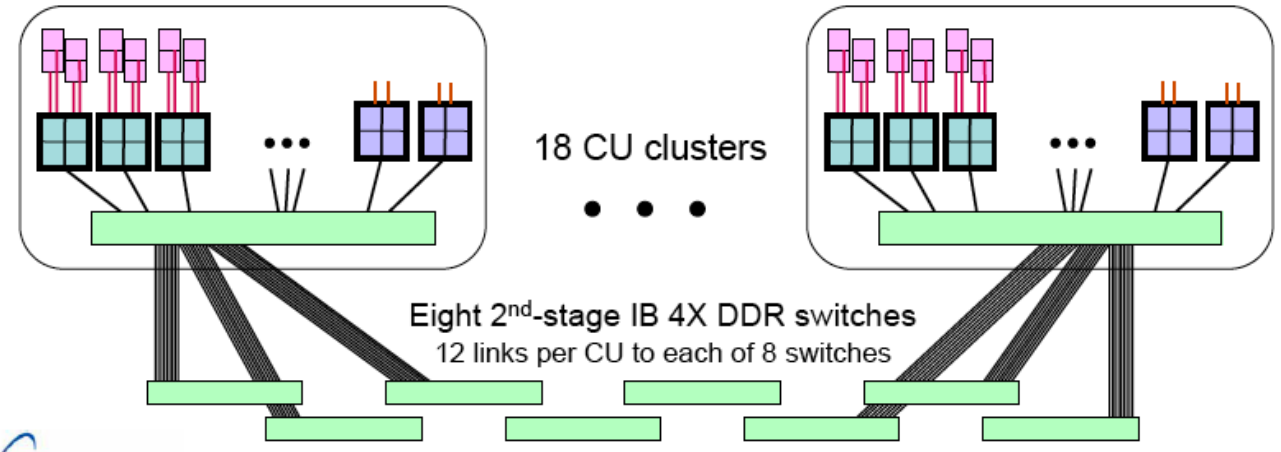
Roadrunner is a petascale system in 2008

Full Roadrunner Specifications:

6,912 dual-core Opterons
49.8 TF DP peak Opteron
27.6 TB Opteron memory

12,960 Cell eDP chips
aka IBM PowerXCell™
1.33 PF DP peak Cell eDP
2.65 PF SP peak Cell eDP
51.8 TB Cell memory
277 TB/s Cell memory BW

3,456 nodes on 2-stage IB 4X DDR
13.8 TB/s aggregate BW (bi-dir) (1st stage)
6.9 TB/s aggregate BW (bi-dir) (2nd stage)
3.5 TB/s bi-section BW (bi-dir) (2nd stage)
432 10 GigE I/O links on 216 I/O nodes
432 GB/s aggregate I/O BW (uni-dir)
(IB limited)



Los Alamos NATIONAL LABORATORY EST. 1943
Operated by the Los Alamos National Security, LLC for the DOE/NNSA

ASC NNSA IBM

Roadrunner first PFlop/s system on Top500, consumes $\sim$ 2.3MW of power

2009-2010 #1 in Top500 - Jaguar

Housed at the National Center for Computational Sciences at Oak Ridge National Laboratory, Jaguar is a Cray XT5 System with 37,376 6-core AMD Opteron processors (224,256 cores total), and Cray's propietary interconnect. Power usage $\sim$ 7MW.

2010-11 #1 in Top500 - Tianhe-1A

Tianhe-1A was installed at the National Supercomputer Center in Tianjin, China, and debuted in the #1 slot of the top500 list in 2010-11.

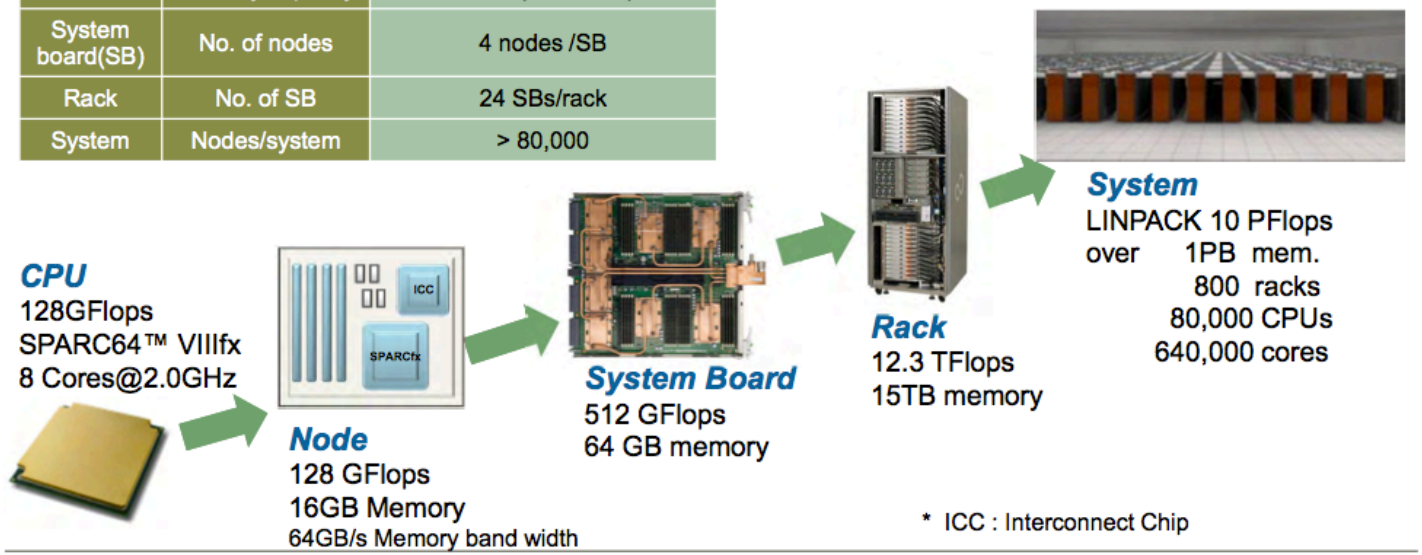
- 14,336 Intel Xeon processors
- 7,168 Nvidia Tesla M2050 GPUs
- 2048 NUDT FT1000 heterogeneous processors
- Proprietary interconnect "Arch"
- 4.04MW Power usage

2011 #1 in Top500 - K Computer at RIKEN

K computer Specifications

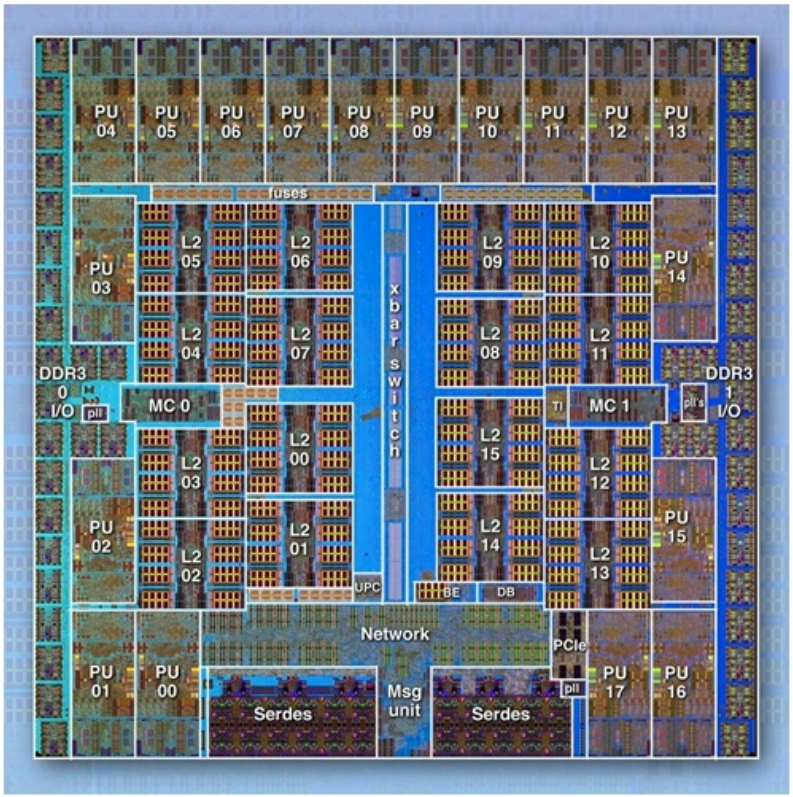
CPU (*SPARC64 VIIIfx*)	Cores/Node	8 cores (@2GHz)
	Performance	128GFlops
	Architecture	SPARC V9 + HPC extension
	Cache	L1(I/D) Cache : 32KB/32KB L2 Cache : 6MB
	Power	58W (typ. 30 C)
	Mem. bandwidth	64GB/s.
Node	Configuration	1 CPU / Node
	Memory capacity	16GB (2GB/core)
System board(SB)	No. of nodes	4 nodes /SB
Rack	No. of SB	24 SBs/rack
System	Nodes/system	> 80,000

Inter-connect	Topology	6D Mesh/Torus
	Performance	5GB/s. for each link
	No. of link	10 links/ node
	Additional feature	H/W barrier, reduction
	Architecture	Routing chip structure (no outside switch box)
Cooling	CPU, ICC*	Direct water cooling
	Other parts	Air cooling



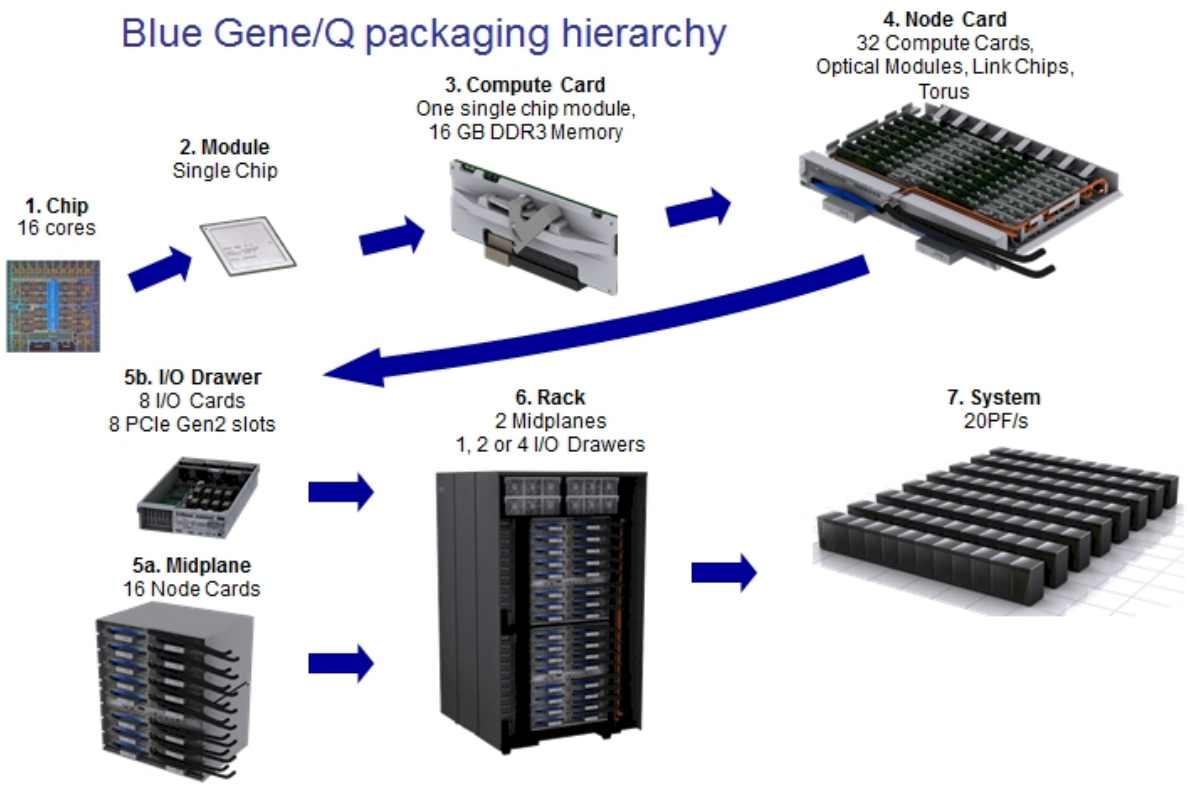
SC10 6 Copyright 2010 FUJITSU LIMITED

2012-06 #1 Sequoia, IBM BG/Q



IBM deployed BG/Q (3rd in the BlueGene line) in 2012. PowerPC A2 chip shown above, 18 cores total, 16 for computation, 1 for operating system (RHEL), 1 extra. 8 Flop/s per clock tick. SoC design, 1.6GHz core frequency at 55W (204.8GFlop/s). Water cooled.

2012-06 #1 Sequoia, IBM BG/Q



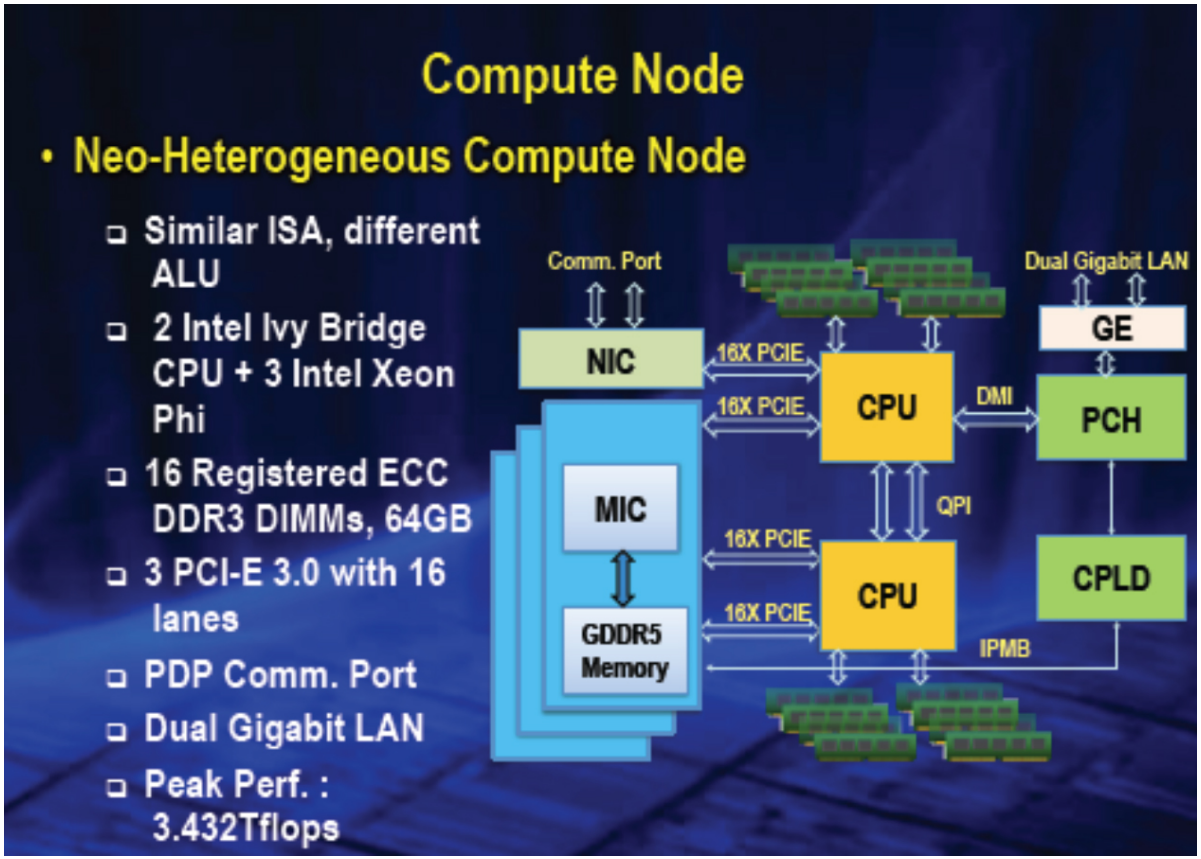
Each rack holds 1024 compute cards, or 16384 cores total. The Sequoia system at LLNL has targeted to have 96 racks total, or 1,572,864 cores (and consumes about 7.9MW of power).

2013-06 #1 Tianhe-2, Intel Phi/MIC

- 16,000 nodes, each with:
  - 2 Intel Xeon "Ivy Bridge" E5-2692 12-core processors (2.2GHz)
  - 3 Intel Xeon Phi 31S1P 57-core co-processors
  - 64 GB memory
- 4096 NUDT FT1500 heterogeneous processors (frontend)
- Proprietary interconnect "TH Express-2"
- 17.8 MW Power usage (24 MW with cooling)

Total cores = 3,120,000 (384,000 "Ivy Bridge" Xeon cores, plus 2,736,000 Phi cores). 34 PFlop/s (54 TPP).

2013-06 #1 Tianhe-2, Intel Phi/MIC



Lessons Learned

- Moore's Law (observation, really) continues - lithography in semiconductor manufacturing advances
- Sequential processor speeds are not advancing (or in a much more limited way):
  - Thermal wall - can not cool them fast enough
  - Memory wall - can not feed the processor data fast enough
  - ILP (instruction level parallelism) wall - limitations on speculative compilation
- What happens to all of these new transistors?
  - Make processors smaller
  - Make L2 caches bigger
  - Add lots and lots more cores
- The future? Massive numbers of multicore servers (8/16/32/... cores per server) harnessed in a distributed fashion