Simulation: The Third Pillar of Science

- Traditional scientific and engineering paradigm:
  1) Do theory or paper design.
  2) Perform experiments or build system.
- Limitations:
  - Too difficult -- build large wind tunnels.
  - Too expensive -- build a throw-away passenger jet.
  - Too slow -- wait for climate or galactic evolution.
  - Too dangerous -- weapons, drug design, climate experimentation.
- Computational science paradigm:
  3) Use high performance computer systems to simulate the phenomenon
     » Base on known physical laws and efficient numerical methods.
Computational Science Definition

Computational science is a rapidly growing multidisciplinary field that uses advanced computing capabilities to understand and solve complex problems. Computational science fuses three distinct elements:

- numerical and non-numerical algorithms and modeling and simulation software developed to solve science (e.g., biological, physical, and social), engineering, and humanities problems;
- advanced system hardware, software, networking, and data management components developed through computer and information science to solve computationally demanding problems;
- the computing infrastructure that supports both science and engineering problem solving and developmental computer and information science.

Relationships Between Computational Science, Computer Science, Mathematics and Applications

http://www.nitrd.gov/pitac/
Some Particularly Challenging Computations

- **Science**
  - Global climate modeling
  - Astrophysical modeling
  - Biology: genomics, protein folding, drug design
  - Computational Chemistry
  - Computational Material Sciences and Nanosciences

- **Engineering**
  - Crash simulation
  - Semiconductor design
  - Earthquake and structural modeling
  - Computation fluid dynamics (airplane design)
  - Combustion (engine design)

- **Business**
  - Financial and economic modeling
  - Transaction processing, web services and search engines

- **Defense**
  - Nuclear weapons -- test by simulations
  - Cryptography

Why Turn to Simulation?

- **When the problem is too . . .**
  - Complex
  - Large / small
  - Expensive
  - Dangerous

- **to do any other way.**
Complex Systems Engineering

R&D Team: Grand Challenge Driven
- Ames Research Center
- Glenn Research Center
- Langley Research Center

Engineering Team: Operations Driven
- Johnson Space Center
- Marshall Space Flight Center
- Industry Partners

Analytic and Visualization

Supercomputers, Storage, & Networks

Grand Challenges

Next Generation Codes & Algorithms

- OVERFLOW Honorable Mention, NASA Software of Year STS-107
- INS3D NASA Software of Year Turbopump Analysis
- CART3D NASA Software of Year STS-107

Modeling Environment (experts and tools)
- Compilers
- Scaling and Porting
- Parallelization Tools

Source: Walt Brooks, NASA

Economic Impact of HPC

- **Airlines:**
  - System-wide logistics optimization systems on parallel systems.
  - Savings: approx. $100 million per airline per year.

- **Automotive design:**
  - Major automotive companies use large systems (500+ CPUs) for:
    - CAD-CAM, crash testing, structural integrity and aerodynamics.
    - One company has 500+ CPU parallel system.
  - Savings: approx. $1 billion per company per year.

- **Semiconductor industry:**
  - Semiconductor firms use large systems (500+ CPUs) for:
    - Device electronics simulation and logic validation
  - Savings: approx. $1 billion per company per year.

- **Securities industry:**
  - Savings: approx. $15 billion per year for U.S. home mortgages.
Why Turn to Simulation?

- Climate / Weather Modeling
- Data intensive problems (data-mining, oil reservoir simulation)
- Problems with large length and time scales (cosmology)
**Titov’s Tsunami Simulation**

- **TITOV**
- **Titov’s Tsunami Simulation**
- **Global model**

---

**Cost (Economic Loss) to Evacuate 1 Mile of Coastline: $1M**

- We now over-warn by a factor of 3
- Average over-warning is 200 miles of coastline, or $200M per event
This problem demands a complete, stable environment (hardware and software)
- 100 TF to stay a factor of 10 ahead of the weather
- Streaming Observations
- Massive Storage and Meta Data Query
- Fast Networking
- Visualization
- Data Mining for Feature Detection

Units of High Performance Computing

<table>
<thead>
<tr>
<th>1 Mflop/s</th>
<th>1 Megaflop/s</th>
<th>10^6 Flop/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Gflop/s</td>
<td>1 Gigaflop/s</td>
<td>10^9 Flop/sec</td>
</tr>
<tr>
<td>1 Tflop/s</td>
<td>1 Teraflop/s</td>
<td>10^{12} Flop/sec</td>
</tr>
<tr>
<td>1 Pflop/s</td>
<td>1 Petaflop/s</td>
<td>10^{15} Flop/sec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1 MB</th>
<th>1 Megabyte</th>
<th>10^6 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GB</td>
<td>1 Gigabyte</td>
<td>10^9 Bytes</td>
</tr>
<tr>
<td>1 TB</td>
<td>1 Terabyte</td>
<td>10^{12} Bytes</td>
</tr>
<tr>
<td>1 PB</td>
<td>1 Petabyte</td>
<td>10^{15} Bytes</td>
</tr>
</tbody>
</table>
Let's say you can print:
- 5 columns of 100 numbers each; on both sides of the page = 1000 numbers (Kflop) in one second (1 Kflop/s)

1 page per second
500 numbers on each side = 1000 numbers

Let's say you can print:
- 5 columns of 100 numbers each; on both sides of the page = 1000 numbers (Kflop) per second

1000 pages about 10 cm = 10^6 numbers (Mflop)
2 reams of paper per second
1 Mflop/s
Let’s say you can print:
- 5 columns of 100 numbers each; on both sides of the page = 1000 numbers (Kflop) per second
- 1000 pages about 10 cm = 10^6 numbers (Mflop)
  2 reams of paper per second

10^9 numbers (Gflop) = 10000 cm = 100 m stack
- Height of Statue of Liberty (printed per second) 1 Gflop/s

10^{12} numbers (Tflop) = 100 km stack;
Altitude achieved by SpaceShipOne to win the prize. 1 Tflop/s (printed per second)
Let’s say you can print:

- 5 columns of 100 numbers each; on both sides of the page = 1000 numbers (Kflop) per second
- 1000 pages about 10 cm = 10^6 numbers (Mflop)
  2 reams of paper / sec
- 10^9 numbers (Gflop) = 10000 cm = 100 m stack; Height of Statue of Liberty per second
- 10^12 numbers (Tflop) = 100 km stack; SpaceShipOne’s distance to space / sec

10^15 numbers (Pflop) = 100,000 km stack printed per second 1 Pflop/s

(¼ distance to the moon)

Let’s say you can print:

- 5 columns of 100 numbers each; on both sides of the page = 1000 numbers (Kflop)/sec
- 1000 pages about 10 cm = 10^6 numbers (Mflop);
  2 reams of paper/sec
- 10^9 numbers (Gflop) = 10000 cm = 100 m stack; Height of Statue of Liberty/sec
- 10^12 numbers (Tflop) = 100 km stack
  SpaceShipOne’s distance/sec
- 10^15 numbers (Pflop) = 100,000 km stack
  ¼ distance to the moon/sec

10^16 numbers (10 Pflop) = 1,000,000 km stack / sec
10 Pflop/s
(distance to the moon and back and then some)
High-Performance Computing
Today

- In the past decade, the world has experienced one of the most exciting periods in computer development.
- Microprocessors have become smaller, denser, and more powerful.
- The result is that microprocessor-based supercomputing is rapidly becoming the technology of preference in attacking some of the most important problems of science and engineering.

Technology Trends:
Microprocessor Capacity

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

2X transistors/Chip Every 1.5 years
Called “Moore’s Law”

Microprocessors have become smaller, denser, and more powerful.
Not just processors, bandwidth, storage, etc
# Eniac and My Laptop

<table>
<thead>
<tr>
<th></th>
<th>Eniac</th>
<th>My Laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Year</strong></td>
<td>1945</td>
<td>2006</td>
</tr>
<tr>
<td><strong>Devices</strong></td>
<td>18,000</td>
<td>6,000,000,000</td>
</tr>
<tr>
<td><strong>Weight (kg)</strong></td>
<td>27,200</td>
<td>0.9</td>
</tr>
<tr>
<td><strong>Size (m³)</strong></td>
<td>68</td>
<td>0.0028</td>
</tr>
<tr>
<td><strong>Power (watts)</strong></td>
<td>20,000</td>
<td>60</td>
</tr>
<tr>
<td><strong>Cost (2000 dollars)</strong></td>
<td>4,630,000</td>
<td>1,000</td>
</tr>
<tr>
<td><strong>Memory (bytes)</strong></td>
<td>~200</td>
<td>1,073,741,824</td>
</tr>
<tr>
<td><strong>Performance (FP/sec)</strong></td>
<td>800</td>
<td>5,000,000,000</td>
</tr>
</tbody>
</table>

# No Exponential is Forever, But perhaps we can Delay it Forever

<table>
<thead>
<tr>
<th></th>
<th>Year of Introduction</th>
<th>Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>4004</td>
<td>1971</td>
<td>2,250</td>
</tr>
<tr>
<td>8086</td>
<td>1972</td>
<td>2,500</td>
</tr>
<tr>
<td>8080</td>
<td>1974</td>
<td>5,000</td>
</tr>
<tr>
<td>8086</td>
<td>1978</td>
<td>29,000</td>
</tr>
<tr>
<td>286</td>
<td>1982</td>
<td>120,000</td>
</tr>
<tr>
<td>Intel486™ processor</td>
<td>1985</td>
<td>275,000</td>
</tr>
<tr>
<td>Intel486™ processor</td>
<td>1989</td>
<td>1,180,000</td>
</tr>
<tr>
<td>Intel® Pentium® II processor</td>
<td>1993</td>
<td>3,100,000</td>
</tr>
<tr>
<td>Intel® Pentium® II processor</td>
<td>1997</td>
<td>7,500,000</td>
</tr>
<tr>
<td>Intel® Pentium® III processor</td>
<td>1999</td>
<td>24,000,000</td>
</tr>
<tr>
<td>Intel® Pentium® 4 processor</td>
<td>2000</td>
<td>42,000,000</td>
</tr>
<tr>
<td>Intel® Itanium® processor</td>
<td>2002</td>
<td>220,000,000</td>
</tr>
<tr>
<td>Intel® Itanium® 2 processor</td>
<td>2003</td>
<td>410,000,000</td>
</tr>
</tbody>
</table>
Today’s processors

- Some equivalences for the microprocessors of today
  - Voltage level
    » A flashlight (~1 volt)
  - Current level
    » An oven (~250 amps)
  - Power level
    » A light bulb (~100 watts)
  - Area
    » A postage stamp (~1 square inch)

Moore’s “Law”

- Something doubles every 18–24 months
- Something was originally the number of transistors
- Something is also considered performance
- Moore’s Law is an exponential
  - Exponentials can not last forever
    » However Moore’s Law has held remarkably true for ~30 years
- BTW: It is really an empiricism rather than a law (not a derogatory comment)
Today’s CPU Architecture:
Heat becoming an unmanageable problem

Square relationship between the cycle time and power.

Intel Developer Forum, Spring 2004 - Pat Gelsinger
(Pentium at 90 W)

Increasing CPU Performance:
A Delicate Balancing Act

We have seen increasing number of gates on a chip and increasing clock speed.

Heat becoming an unmanageable problem, Intel Processors > 100 Watts

We will not see the dramatic increases in clock speeds in the future.

However, the number of gates on a chip will continue to increase.

Intel Yonah will double the processing power on a per watt basis.
No Free Lunch For Traditional Software
(Without highly concurrent software it won’t get any faster!)

Additional operations per second if code can take advantage of concurrency

CPU Desktop Trends 2004-2010
- Relative processing power will continue to double every 18 months
- 256 logical processors per chip in late 2010
Percentage of peak

- A rule of thumb that often applies
  - A contemporary RISC processor, for a spectrum of applications, delivers (i.e., sustains) 10% of peak performance
- There are exceptions to this rule, in both directions
- Why such low efficiency?
- There are two primary reasons behind the disappointing percentage of peak
  - IPC (in)efficiency
  - Memory (in)efficiency

IPC

- Today the theoretical IPC (instructions per cycle) is 4 in most contemporary RISC processors (6 in Itanium)
- Detailed analysis for a spectrum of applications indicates that the average IPC is 1.2–1.4
- We are leaving ~75% of the possible performance on the table...
Why Fast Machines Run Slow

- **Latency**
  - Waiting for access to memory or other parts of the system

- **Overhead**
  - Extra work that has to be done to manage program concurrency and parallel resources the real work you want to perform

- **Starvation**
  - Not enough work to do due to insufficient parallelism or poor load balancing among distributed resources

- **Contention**
  - Delays due to fighting over what task gets to use a shared resource next. Network bandwidth is a major constraint.
Memory hierarchy

- Typical latencies for today’s technology

<table>
<thead>
<tr>
<th>Hierarchy</th>
<th>Processor clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register</td>
<td>1</td>
</tr>
<tr>
<td>L1 cache</td>
<td>2-3</td>
</tr>
<tr>
<td>L2 cache</td>
<td>6-12</td>
</tr>
<tr>
<td>L3 cache</td>
<td>14-40</td>
</tr>
<tr>
<td>Near memory</td>
<td>100-300</td>
</tr>
<tr>
<td>Far memory</td>
<td>300-900</td>
</tr>
<tr>
<td>Remote memory</td>
<td>( O(10^3) )</td>
</tr>
<tr>
<td>Message-passing</td>
<td>( O(10^3)-O(10^4) )</td>
</tr>
</tbody>
</table>

Hierarchy

- Most programs have a high degree of locality in their accesses
  - spatial locality: accessing things nearby previous accesses
  - temporal locality: reusing an item that was previously accessed
- Memory hierarchy tries to exploit locality

<table>
<thead>
<tr>
<th>Speed</th>
<th>1ns</th>
<th>10ns</th>
<th>100ns</th>
<th>10ms</th>
<th>10sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>B</td>
<td>KB</td>
<td>MB</td>
<td>GB</td>
<td>TB</td>
</tr>
</tbody>
</table>
Memory bandwidth

- To provide bandwidth to the processor, the bus either needs to be faster or wider.
- Busses are limited to perhaps 400-800 MHz.
- Links are faster:
  - Single-ended: 0.5–1 GT/s
  - Differential: 2.5–5.0 (future) GT/s
  - Increased link frequencies increase error rates requiring coding and redundancy, thus increasing power and die size and not helping bandwidth.
- Making things wider requires pin-out (Si real estate) and power:
  - Both power and pin-out are serious issues.

Processor-DRAM Memory Gap

- Processor performance (µProc) grows at 60%/yr. (2X/1.5yr).
- DRAM performance grows at 9%/yr. (2X/10 yrs).
- The Processor-Memory Performance Gap grows at 50%/year.

“Moore’s Law”
**Processor in Memory (PIM)**

- **PIM merges logic with memory**
  - Wide ALUs next to the row buffer
  - Optimized for memory throughput, not ALU utilization

- **PIM has the potential of riding Moore's law while**
  - greatly increasing effective memory bandwidth,
  - providing many more concurrent execution threads,
  - reducing latency,
  - reducing power, and
  - increasing overall system efficiency

- **It may also simplify programming and system design**

---

**Internet – 4th Revolution in Telecommunications**

- Telephone, Radio, Television
- Growth in Internet outstrips the others
- Exponential growth since 1985
- Traffic doubles every 100 days

---

**Growth of Internet Hosts**

*Sept. 1999 - Sept. 2002*

**Domain names**

*Hollow's Internet Timeline © 1998 Deborah Zink
http://www.isc.org/isEmptyInternet DTWU*
Peer to Peer Computing

- Peer-to-peer is a style of networking in which a group of computers communicate directly with each other.
- Wireless communication
- Home computer in the utility room, next to the water heater and furnace.
- Web tablets
- BitTorrent
- Imbedded computers in things all tied together.
  - Books, furniture, milk cartons, etc
- Smart Appliances
  - Refrigerator, scale, etc

Internet On Everything
SETI@home: Global Distributed Computing

- Running on 500,000 PCs, ~1000 CPU Years per Day
  - 485,821 CPU Years so far
- Sophisticated Data & Signal Processing Analysis
- Distributes Datasets from Arecibo Radio Telescope

SETI@home

- Use thousands of Internet-connected PCs to help in the search for extraterrestrial intelligence.
- When their computer is idle or being wasted this software will download a 300 kilobyte chunk of data for analysis. Performs about 3 Tflops for each client in 15 hours.
- The results of this analysis are sent back to the SETI team, combined with thousands of other participants.

- Largest distributed computation project in existence
  - Averaging 40 Tflop/s
- Today a number of companies trying this for profit.
Google query attributes
- 150M queries/day (2000/second)
- 100 countries
- 8.0B documents in the index

Data centers
- 100,000 Linux systems in data centers around the world
  - 15 TFlop/s and 1000 TB total capability
  - 40-80 1U/2U servers/cabinet
  - 100 MB Ethernet switches/cabinet with gigabit Ethernet uplink
- growth from 4,000 systems (June 2000)
  - 18M queries then

Performance and operation
- simple reissue of failed commands to new servers
- no performance debugging
  - problems are not reproducible

Next Generation Web

- To treat CPU cycles and software like commodities.
- Enable the coordinated use of geographically distributed resources – in the absence of central control and existing trust relationships.
- Computing power is produced much like utilities such as power and water are produced for consumers.
- Users will have access to “power” on demand
- This is one of our efforts at UT.
Why Parallel Computing

- Desire to solve bigger, more realistic applications problems.
- Fundamental limits are being approached.
- More cost effective solution

Principles of Parallel Computing

- Parallelism and Amdahl's Law
- Granularity
- Locality
- Load balance
- Coordination and synchronization
- Performance modeling

All of these things makes parallel programming even harder than sequential programming.
“Automatic” Parallelism in Modern Machines

- Bit level parallelism
  - within floating point operations, etc.
- Instruction level parallelism (ILP)
  - multiple instructions execute per clock cycle
- Memory system parallelism
  - overlap of memory operations with computation
- OS parallelism
  - multiple jobs run in parallel on commodity SMPs

Limits to all of these -- for very high performance, need user to identify, schedule and coordinate parallel tasks

Finding Enough Parallelism

- Suppose only part of an application seems parallel
- Amdahl's law
  - let $f_s$ be the fraction of work done sequentially, $(1-f_s)$ is fraction parallelizable
  - $N = \text{number of processors}$

- Even if the parallel part speeds up perfectly may be limited by the sequential part
Amdahl’s Law places a strict limit on the speedup that can be realized by using multiple processors. Two equivalent expressions for Amdahl’s Law are given below:

\[ t_N = \frac{f_p}{N} + f_s t_1 \]  
Effect of multiple processors on run time

\[ S = \frac{1}{f_s + f_p/N} \]  
Effect of multiple processors on speedup

Where:
- \( f_s \) = serial fraction of code
- \( f_p \) = parallel fraction of code = 1 - \( f_s \)
- \( N \) = number of processors

Illustration of Amdahl’s Law

It takes only a small fraction of serial content in a code to degrade the parallel performance. It is essential to determine the scaling behavior of your code before doing production runs using large numbers of processors.
Overhead of Parallelism

- *Given enough parallel work, this is the biggest barrier to getting desired speedup*
- Parallelism overheads include:
  - cost of starting a thread or process
  - cost of communicating shared data
  - cost of synchronizing
  - extra (redundant) computation
- Each of these can be in the range of milliseconds (=millions of flops) on some systems
- Tradeoff: Algorithm needs sufficiently large units of work to run fast in parallel (i.e. large granularity), but not so large that there is not enough parallel work

Locality and Parallelism

- Large memories are slow, fast memories are small
- Storage hierarchies are large and fast on average
- Parallel processors, collectively, have large, fast $\triangleright$ the slow accesses to “remote” data we call “communication”
- Algorithm should do most work on local data
Load Imbalance

- Load imbalance is the time that some processors in the system are idle due to
  - insufficient parallelism (during that phase)
  - unequal size tasks
- Examples of the latter
  - adapting to “interesting parts of a domain”
  - tree-structured computations
  - fundamentally unstructured problems
- Algorithm needs to balance load

What is Ahead?

- Greater instruction level parallelism?
- Bigger caches?
- Multiple processors per chip?
- Complete systems on a chip? (Portable Systems)
- High performance LAN, Interface, and Interconnect
Directions

- Move toward shared memory
  - SMPs and Distributed Shared Memory
  - Shared address space with deep memory hierarchy
- Clustering of shared memory machines for scalability
- Efficiency of message passing and data parallel programming
  - Helped by standards efforts such as MPI and HPF

High Performance Computers

- ~ 20 years ago
  - $1 \times 10^6$ Floating Point Ops/sec (Mflop/s)
    - Scalar based
- ~ 10 years ago
  - $1 \times 10^9$ Floating Point Ops/sec (Gflop/s)
    - Vector & Shared memory computing, bandwidth aware
    - Block partitioned, latency tolerant
- ~ Today
  - $1 \times 10^{12}$ Floating Point Ops/sec (Tflop/s)
    - Highly parallel, distributed processing, message passing, network based
    - Data decomposition, communication/computation
- ~ 5 years away
  - $1 \times 10^{15}$ Floating Point Ops/sec (Pflop/s)
    - Many more levels, combination/grid/HiPC
    - More adaptive, LT and bandwidth aware, fault tolerant, extended precision, attention to SMP nodes
Top 500 Computers

- Listing of the 500 most powerful Computers in the World
- Yardstick: Rmax from LINPACK MPP

\[ Ax = b, \text{ dense problem} \]

Updated twice a year

SC‘xy in the States in November
Meeting in Germany in June

What is a Supercomputer?

- A supercomputer is a hardware and software system that provides close to the maximum performance that can currently be achieved.

- Over the last 12 years the range for the Top500 has increased greater than Moore’s Law

- 1993:
  - #1 = 59.7 GFlop/s
  - #500 = 422 MFlop/s

- 2004:
  - #1 = 280 TFlop/s
  - #500 = 1.64 TFlop/s

Why do we need them?

Almost all of the technical areas that are important to the well-being of humanity use supercomputing in fundamental and essential ways.

Computational fluid dynamics, protein folding, climate modeling, national security, in particular for cryptanalysis and for simulating nuclear weapons to name a few.
Architecture/Systems Continuum

- **Tightly Coupled**
  - Custom processor with custom interconnect
    - Cray X1
    - NEC SX-8
    - IBM Regatta
    - IBM Blue Gene/L
  - Commodity processor with custom interconnect
    - SGI Altix
    - Intel Itanium 2
    - Cray XT3, XD1
    - AMD Opteron
  - Commodity processor with commodity interconnect
    - Clusters
      - Pentium, Itanium, Opteron, Alpha
      - SGI, Infiniband, Myrinet, Quadrics
    - NEC TX7
    - IBM eServer
    - Dawning

- **Loosely Coupled**

---

26th List: The TOP10

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Computer</th>
<th>Rmax (TF/s)</th>
<th>Year</th>
<th>Country</th>
<th>Site</th>
<th>Proc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 IBM</td>
<td>BlueGene/S, eServer Blue Gene</td>
<td>280.6</td>
<td>2005</td>
<td>USA</td>
<td>DOE/INNSA/LLNL</td>
<td>custom 131072</td>
</tr>
<tr>
<td>2 IBM</td>
<td>BGW eServer Blue Gene</td>
<td>91.29</td>
<td>2005</td>
<td>USA</td>
<td>IBM Thomas Watson</td>
<td>custom 40960</td>
</tr>
<tr>
<td>3 IBM</td>
<td>ASC Purple Power5 g575</td>
<td>63.39</td>
<td>2005</td>
<td>USA</td>
<td>DOE/INNSA/LLNL</td>
<td>custom 10240</td>
</tr>
<tr>
<td>4 SGI</td>
<td>Columbia Altix, Itanium/Infiniband</td>
<td>51.87</td>
<td>2004</td>
<td>USA</td>
<td>NASA Ames</td>
<td>hybrid 10160</td>
</tr>
<tr>
<td>5 Dell</td>
<td>Thunderbird Pentium/Infiniband</td>
<td>38.27</td>
<td>2005</td>
<td>USA</td>
<td>Sandia</td>
<td>custom 8000</td>
</tr>
<tr>
<td>6 Cray</td>
<td>Red Storm Cray XT3 AMD</td>
<td>36.19</td>
<td>2005</td>
<td>USA</td>
<td>Sandia</td>
<td>hybrid 10880</td>
</tr>
<tr>
<td>7 NEC</td>
<td>Earth-Simulator SX-6</td>
<td>35.86</td>
<td>2002</td>
<td>Japan</td>
<td>Earth Simulator Center</td>
<td>custom 5120</td>
</tr>
<tr>
<td>8 IBM</td>
<td>MareNostrum PPC 970/Myrinet</td>
<td>27.91</td>
<td>2005</td>
<td>Spain</td>
<td>Barcelona Supercomputer Center</td>
<td>custom 4800</td>
</tr>
<tr>
<td>9 IBM</td>
<td>eServer Blue Gene</td>
<td>27.45</td>
<td>2005</td>
<td>Netherlands</td>
<td>ASTRON University Groningen</td>
<td>custom 12288</td>
</tr>
<tr>
<td>10 Cray</td>
<td>Jaguar Cray XT3 AMD</td>
<td>20.53</td>
<td>2005</td>
<td>USA</td>
<td>Oak Ridge National Lab</td>
<td>hybrid 5200</td>
</tr>
</tbody>
</table>
Interconnects / Systems

Top500 Conclusions

- Microprocessor based supercomputers have brought a major change in accessibility and affordability.
- MPPs continue to account of more than half of all installed high-performance computers worldwide.
Distributed and Parallel Systems

**Distributed systems**
- **heterogeneous**
  - SET@home
  - Entropia
  - Grid Computing
  - Beowulf
  - Berkeley NOW
  - SNL Cplant
  - Parallel Dismem
  - ASCI Tflops

**Massively parallel systems**
- **homogeneous**

- Gather (unused) resources
- Steal cycles
- System SW manages resources
- System SW adds value
- 10% - 20% overhead is OK
- Resources drive applications
- Time to completion is not critical
- Time-shared

- Bounded set of resources
- Apps grow to consume all cycles
- Application manages resources
- System SW gets in the way
- 5% overhead is maximum
- Apps drive purchase of equipment
- Real-time constraints
- Space-shared

**Virtual Environments**

```
0.32E-08 0.00E+00 0.00E+00 0.00E+00 0.38E-06 0.13E-05 0.22E-05 0.33E-05 0.59E-05 0.11E-04
0.18E-04 0.23E-04 0.23E-04 0.21E-04 0.67E-04 0.38E-03 0.90E-03 0.18E-02 0.30E-02 0.43E-02
0.50E-02 0.51E-02 0.49E-02 0.44E-02 0.39E-02 0.35E-02 0.31E-02 0.28E-02 0.27E-02 0.26E-02
0.26E-02 0.27E-02 0.28E-02 0.30E-02 0.33E-02 0.36E-02 0.38E-02 0.39E-02 0.39E-02 0.38E-02
0.34E-02 0.30E-02 0.27E-02 0.24E-02 0.21E-02 0.18E-02 0.14E-02 0.11E-02 0.94E-03 0.79E-03
0.63E-03 0.48E-03 0.35E-03 0.24E-03 0.15E-03 0.80E-04 0.34E-04 0.86E-05 0.16E-05
0.18E-06 0.34E-06 0.00E+00 0.00E+00 0.00E+00 0.00E+00 0.00E+00 0.00E+00 0.00E+00 0.00E+00
0.30E-00 0.00E+00 0.00E+00 0.00E+00 0.24E-08 0.00E+00 0.00E+00 0.00E+00 0.29E-06 0.11E-05
0.19E-05 0.30E-05 0.53E-05 0.56E-05 0.15E-04 0.20E-04 0.20E-04 0.18E-04 0.27E-04 0.23E-03
0.65E-03 0.14E-02 0.27E-02 0.40E-02 0.49E-02 0.51E-02 0.49E-02 0.45E-02 0.40E-02 0.35E-02
0.31E-02 0.28E-02 0.27E-02 0.26E-02 0.26E-02 0.27E-02 0.28E-02 0.30E-02 0.33E-02 0.36E-02
0.38E-02 0.39E-02 0.39E-02 0.37E-02 0.34E-02 0.30E-02 0.27E-02 0.24E-02 0.21E-02 0.18E-02
0.16E-02 0.14E-02 0.12E-02 0.98E-03 0.81E-03 0.65E-03 0.51E-03 0.38E-03 0.27E-03 0.17E-03
0.99E-04 0.47E-04 0.16E-04 0.36E-05 0.62E-05 0.41E-07 0.75E-10 0.00E+00 0.00E+00 0.00E+00
0.00E+00 0.00E+00 0.00E+00 0.00E+00 0.00E+00 0.00E+00 0.00E+00 0.00E+00 0.00E+00 0.00E+00
0.20E-00 0.00E+00 0.19E-06 0.84E-06 0.16E-05 0.27E-05 0.47E-05 0.82E-05 0.13E-04 0.17E-04
0.17E-04 0.15E-04 0.16E-04 0.15E-03 0.41E-03 0.11E-02 0.23E-02 0.37E-02 0.48E-02 0.51E-02
0.49E-02 0.45E-02 0.40E-02 0.35E-02 0.31E-02 0.28E-02 0.27E-02 0.26E-02 0.26E-02 0.27E-02
0.28E-02 0.31E-02 0.33E-02 0.36E-02 0.38E-02 0.39E-02 0.38E-02 0.36E-02 0.33E-02 0.29E-02
```

Do they make any sense?
Performance Improvements for Scientific Computing Problems

![Graph showing performance improvements over time for different computational methods like Multi-grid, Conjugate Gradient, and Sparse GE.](image)
Different Architectures

- **Parallel computing**: single systems with many processors working on same problem
- **Distributed computing**: many systems loosely coupled by a scheduler to work on related problems
- **Grid Computing**: many systems tightly coupled by software, perhaps geographically distributed, to work together on single problems or on related problems

Types of Parallel Computers

- The simplest and most useful way to classify modern parallel computers is by their memory model:
  - shared memory
  - distributed memory
Shared vs. Distributed Memory

Shared memory - single address space. All processors have access to a pool of shared memory. (Ex: SGI Origin, Sun E10000)

Distributed memory - each processor has its own local memory. Must do message passing to exchange data between processors. (Ex: CRAY T3E, IBM SP, clusters)

Shared Memory: UMA vs. NUMA

Uniform memory access (UMA): Each processor has uniform access to memory. Also known as symmetric multiprocessors (Sun E10000)

Non-uniform memory access (NUMA): Time for memory access depends on location of data. Local access is faster than non-local access. Easier to scale than SMPs (SGI Origin)
Distributed Memory: MPPs vs. Clusters

- Processors-memory nodes are connected by some type of interconnect network
  - Massively Parallel Processor (MPP): tightly integrated, single system image.
  - Cluster: individual computers connected by s/w

Both shared and distributed memory systems have:
1. processors: now generally commodity RISC processors
2. memory: now generally commodity DRAM
3. network/interconnect: between the processors and memory (bus, crossbar, fat tree, torus, hypercube, etc.)

We will now begin to describe these pieces in detail, starting with definitions of terms.
Interconnect-Related Terms

- **Latency**: How long does it take to start sending a "message"? Measured in microseconds.
  
  (Also in processors: How long does it take to output results of some operations, such as floating point add, divide etc., which are pipelined?)

- **Bandwidth**: What data rate can be sustained once the message is started? Measured in Mbytes/sec.

---

Interconnect-Related Terms

**Topology**: the manner in which the nodes are connected.

- Best choice would be a fully connected network (every processor to every other). Unfeasible for cost and scaling reasons.
- Instead, processors are arranged in some variation of a grid, torus, or hypercube.

![3-d hypercube](image1)

![2-d mesh](image2)

![2-d torus](image3)
Highly Parallel Supercomputing: Where Are We?

**Performance:**
- Sustained performance has dramatically increased during the last year.
- On most applications, sustained performance per dollar now exceeds that of conventional supercomputers. But...
- Conventional systems are still faster on some applications.

**Languages and compilers:**
- Standardized, portable, high-level languages such as HPF, PVM and MPI are available. But...
- Initial HPF releases are not very efficient.
- Message passing programming is tedious and hard to debug.
- Programming difficulty remains a major obstacle to usage by mainstream scientists.

**Operating systems:**
- Robustness and reliability are improving.
- New system management tools improve system utilization. But...
- Reliability still not as good as conventional systems.

**I/O subsystems:**
- New RAID disks, HiPPI interfaces, etc. provide substantially improved I/O performance. But...
- I/O remains a bottleneck on some systems.
The Importance of Standards - Software

- Writing programs for MPP is hard ...
- But ... one-off efforts if written in a standard language
- Past lack of parallel programming standards ...
  - ... has restricted uptake of technology (to "enthusiasts")
  - ... reduced portability (over a range of current architectures and between future generations)
- Now standards exist: (PVM, MPI & HPF), which ...
  - ... allows users & manufacturers to protect software investment
  - ... encourage growth of a "third party" parallel software industry & parallel versions of widely used codes

The Importance of Standards - Hardware

- Processors
  - commodity RISC processors
- Interconnects
  - high bandwidth, low latency communications protocol
  - no de-facto standard yet (ATM, Fibre Channel, HPPI, FDDI)
- Growing demand for total solution:
  - robust hardware + usable software
- HPC systems containing all the programming tools / environments / languages / libraries / applications packages found on desktops
The Future of HPC

- The expense of being different is being replaced by the economics of being the same
- HPC needs to lose its "special purpose" tag
- Still has to bring about the promise of scalable general purpose computing ...
- ... but it is dangerous to ignore this technology
- Final success when MPP technology is embedded in desktop computing
- Yesterday's HPC is today's mainframe is tomorrow's workstation

Achieving TeraFlops

- In 1991, 1 Gflop/s
- 1000 fold increase
  - Architecture
    - exploiting parallelism
  - Processor, communication, memory
    - Moore's Law
  - Algorithm improvements
    - block-partitioned algorithms
Future: Petaflops ( $10^{15}$ fl pt ops/s)

Today $\approx \sqrt{10^{15}}$ flops for our workstations

- A Pflop for 1 second $\approx$ a typical workstation computing for 1 year.
- From an algorithmic standpoint:
  - concurrency
  - data locality
  - latency & sync
  - floating point accuracy
  - dynamic redistribution of workload
  - new language and constructs
  - role of numerical libraries
  - algorithm adaptation to hardware failure

A Petaflops Computer System

- 1 Pflop/s sustained computing
- Between 10,000 and 1,000,000 processors
- Between 10 TB and 1PB main memory
- Commensurate I/O bandwidth, mass store, etc.
- If built today, cost $40 B and consume 1 TWatt.
- May be feasible and “affordable” by the year 2010