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 Fuses Three Distinct Elements:

- Application Science/Engineering
- Computer Science
- Mathematics

---

### Computational Science As An Emerging Academic Pursuit

- Many Programs in Computational Science
  - College for Computing
    - Georgia Tech, NJIT, CMU, ...
  - Degrees
    - Rice, Utah, UCSB, ...
  - Minor
    - Penn State, U Wisc, SUNY Brockport
  - Certificate
    - Old Dominion, U of Georgia, Boston U, ...
  - Concentration
    - Cornell, Northeastern, Colorado State, ...
  - Courses

---

### At the University of Tennessee

- A few years ago there was a discussion to create a program in Computational Science
- This program evolved out of a set of meetings and discussions with faculty, students, and administration.
- Modeled on a similar minor degree program in the Statistics Department on campus.
- Appeared in the 2007-2008 Graduate Catalog

---

### Graduate Minor in Computational Science

- Students in one of the three general areas of Computational Science;
  - Applied Mathematics,
  - Computer related, or
  - a Domain Science

- will become exposed to and better versed in the other two areas that are currently outside their “home” area.
- A pool of courses which deals with each of the three main areas has been put together by participating department for students to select from.
- Interdisciplinary Graduate Minor in Computational Science (IGMCS)
IGMCS: Requirements

- The Minor requires a combination of course work from three disciplines - Computer related, Mathematics/Stat, and a participating Science/Engineering domain (e.g., Chemical Engineering, Chemistry, Physics).

- At the Masters level a minor in Computational Science will require 9 hours (3 courses) from the pools.
  - At least 6 hours (2 courses) must be taken outside the student's home area.
  - Students must take at least 3 hours (1 course) from each of the 2 non-home areas.

- At the Doctoral level a minor in computation science will require 15 hours (5 courses) from the pools.
  - At least 9 hours (3 courses) must be taken outside the student's home area.
  - Students must take at least 3 hours (1 course) from each of the 2 non-home areas.

IGMCS Process for Students

1. A student, with guidance from their faculty advisor, lays out a program of courses
2. Next, discussion with department’s IGMCS liaison
3. Form generated with courses to be taken
4. Form is submitted for approval by the IGMCS Program Committee

IGMCS Participating Departments

<table>
<thead>
<tr>
<th>Department</th>
<th>IGMCS Liaison</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biochemistry &amp; Cellular and Molecular Biology</td>
<td>Dr. Cynthia Peterson</td>
<td><a href="mailto:cbpeters@utk.edu">cbpeters@utk.edu</a></td>
</tr>
<tr>
<td>Chemical Engineering</td>
<td>Dr. David Klette</td>
<td><a href="mailto:dkeffer@utk.edu">dkeffer@utk.edu</a></td>
</tr>
<tr>
<td>Chemistry</td>
<td>Dr. Robert Hinde</td>
<td><a href="mailto:rhinde@utk.edu">rhinde@utk.edu</a></td>
</tr>
<tr>
<td>Ecology &amp; Evolutionary Biology</td>
<td>Dr. Luigi Gross</td>
<td><a href="mailto:lgross@utk.edu">lgross@utk.edu</a></td>
</tr>
<tr>
<td>Electrical Engineering and Computer Science</td>
<td>Dr. Jack Dongarra</td>
<td><a href="mailto:jdongarra@cs.utk.edu">jdongarra@cs.utk.edu</a></td>
</tr>
<tr>
<td>Genome Science &amp; Technology</td>
<td>Dr. Cynthia Peterson</td>
<td><a href="mailto:cbpeters@utk.edu">cbpeters@utk.edu</a></td>
</tr>
<tr>
<td>Geography</td>
<td>Dr. Bruce Ralston</td>
<td><a href="mailto:bralston@vbi.uab.edu">bralston@vbi.uab.edu</a></td>
</tr>
<tr>
<td>Information Science</td>
<td>Dr. Peiling Wang</td>
<td><a href="mailto:pwl@vbi.uab.edu">pwl@vbi.uab.edu</a></td>
</tr>
<tr>
<td>Mathematics</td>
<td>Dr. Chuck C. Collins</td>
<td><a href="mailto:ccollins@math.utk.edu">ccollins@math.utk.edu</a></td>
</tr>
<tr>
<td>Physics</td>
<td>Dr. Thomas Popebrooke</td>
<td><a href="mailto:tpopebrooke@vbi.uab.edu">tpopebrooke@vbi.uab.edu</a></td>
</tr>
<tr>
<td>Statistics</td>
<td>Dr. Thomas Popebrooke</td>
<td><a href="mailto:tpopebrooke@vbi.uab.edu">tpopebrooke@vbi.uab.edu</a></td>
</tr>
</tbody>
</table>

Students in Departments Not Participating in the IGMCS Program

- A student in such a situation can still participate.
  - Student and advisor should submit to the Chair of the IGMCS Program Committee the courses to be taken.
  - Requirement is still the same: Minor requires a combination of course work from three disciplines - Computer Science related, Mathematics/Stat, and a participating Science/Engineering domain (e.g., Chemical Engineering, Chemistry, Physics).
- Student’s department should be encouraged to participate in the IGMCS program.
  - Easy to do, needs approved set of courses and a liaison

Internship

- This is optional but strongly encouraged.
- Students in the program can fulfill 3 hrs. of their requirement through an Internship with researchers outside the student’s major.
- The internship may be taken offsite, e.g. ORNL, or on campus by working with a faculty member in another department.
- Internships must have the approval of the IGMCS Program Committee.
Why Turn to Simulation?

- When the problem is too...
  - Complex
  - Large / small
  - Expensive
  - Dangerous
- to do any other way.

Weather and Economic Loss

- $10T U.S. economy
  - 40% is adversely affected by weather and climate
- $1M in loss to evacuate each mile of coastline
  - we now over warn by 3X!
- Improved forecasts
  - lives saved and reduced cost
- LEAD
  - Linked Environments for Atmospheric Discovery
    - Oklahoma, Indiana, UCAR, Colorado State, Howard, Alabama, Millersville, NCSA, North Carolina

Example Science and Engineering Drivers

- Climate
- Nuclear Energy
- Combustion
- Advanced Materials
- 
- LEAD
  - Linked Environments for Atmospheric Discovery
- Basic Science
- Common Needs
  - Multiscale
  - Uncertainty Quantification
  - Rare Event Statistics

Look at the Fastest Computers

- Strategic importance of supercomputing
  - Essential for scientific discovery
  - Critical for national security
  - Fundamental contributor to the economy and competitiveness through use in engineering and manufacturing
- Supercomputers are the tool for solving the most challenging problems through simulations

Units of High Performance Computing

<table>
<thead>
<tr>
<th>Units</th>
<th>Prefix</th>
<th>Symbol</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MFlop/s</td>
<td>1 Mega</td>
<td>MFlop</td>
<td>$10^6$ Flop/sec</td>
</tr>
<tr>
<td>1 GFlop/s</td>
<td>1 Giga</td>
<td>GFlop</td>
<td>$10^9$ Flop/sec</td>
</tr>
<tr>
<td>1 TFlop/s</td>
<td>1 Tera</td>
<td>TFlop</td>
<td>$10^{12}$ Flop/sec</td>
</tr>
<tr>
<td>1 PFlop/s</td>
<td>1 Peta</td>
<td>PFlop</td>
<td>$10^{15}$ Flop/sec</td>
</tr>
<tr>
<td>1 MB</td>
<td>1 Mega</td>
<td>byte</td>
<td>$10^6$ Bytes</td>
</tr>
<tr>
<td>1 GB</td>
<td>1 Giga</td>
<td>byte</td>
<td>$10^9$ Bytes</td>
</tr>
<tr>
<td>1 TB</td>
<td>1 Tera</td>
<td>byte</td>
<td>$10^{12}$ Bytes</td>
</tr>
<tr>
<td>1 PB</td>
<td>1 Peta</td>
<td>byte</td>
<td>$10^{15}$ Bytes</td>
</tr>
</tbody>
</table>
**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**


Number of devices/chip doubles every 18 months

2X transistors/Chip Every 1.5 years

Called “Moore’s Law”

**Distributed and Parallel Systems**

- Distributed systems heterogeneous
- SETI@home
- Entropy
- Grid Computing
- Berkley NOW
- 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

**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

**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

- Largest distributed computation project in existence
  - Potential 769 Tflop/s
- Guinness World Records as the largest computation in history.
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

Looking at the Gordon Bell Prize

- 1 GFlop/s; 1988; Cray Y-MP; 8 Processors
  - Static finite element analysis
- 1 TFlop/s; 1998; Cray T3E; 1024 Processors
  - Modeling of metallic magnet atoms, using a variation of the locally self-consistent multiple scattering method.
- 1 PFlop/s; 2008; Cray XT5; $1 \times 10^5$ Processors
  - Superconductive materials
- 1 EFlop/s; ?; $1 \times 10^7$ Processors ($10^9$ threads)
Performance Development in Top500

Countries Share

Customer Segments

Performance of Top20 Over 10 Years

Pflop/s Club (11 systems; Peak)
Industrial Use of Supercomputers

- Of the 500 Fastest Supercomputer
- Worldwide, Industrial Use is > 60%

Power is an Industry Wide Problem

Google facilities
- leveraging hydroelectric power
- old aluminum plants

Microsoft and Yahoo are building big data centers upstream in Wenatchee and Quincy, Wash.
- To keep up with Google, which means they need cheap electricity and readily accessible data networking

Microsoft Quincy, Wash. 470,000 Sq Ft. 47MW!
### Commercial Data Centers

- **Microsoft**: 700,000 sq ft in Chicago
- **Apple**: 500,000 sq ft in Rural NC at 4 cents kW/h

### 36th List: The TOP10

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>Computer</th>
<th>Country</th>
<th>Cores</th>
<th>Pflops</th>
<th>% of Peak</th>
<th>Power (MW)</th>
<th>Peak W</th>
<th>Rmax % of Peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Nat. Supercomputer Center in Tianjin</td>
<td>NUDT X5670 2 x 106</td>
<td>China</td>
<td>186,368</td>
<td>2.57</td>
<td>55</td>
<td>2.95</td>
<td>4.04</td>
<td>616</td>
</tr>
<tr>
<td>2</td>
<td>DOE / ORNL Oak Ridge Nat Lab</td>
<td>Jaguar / ORNL</td>
<td>USA</td>
<td>204,168</td>
<td>1.76</td>
<td>75</td>
<td>3.09</td>
<td>7.10</td>
<td>261</td>
</tr>
<tr>
<td>3</td>
<td>Nat. Supercomputer Center in Shanghai</td>
<td>NEFU Supercomputer System in Shanghai</td>
<td>China</td>
<td>120,640</td>
<td>1.27</td>
<td>43</td>
<td>2.26</td>
<td>2.58</td>
<td>491</td>
</tr>
<tr>
<td>4</td>
<td>IEF Center, Tokyo Institute of Technology</td>
<td>Tera-100 Bull bips supercomputer</td>
<td>Japan</td>
<td>73,178</td>
<td>1.19</td>
<td>52</td>
<td>2.6 GHz</td>
<td>1.40</td>
<td>262</td>
</tr>
<tr>
<td>5</td>
<td>DOE / NERSC National Energy Research Supercomputing Center</td>
<td>Hopper, IBM QS22/LS21</td>
<td>USA</td>
<td>152,408</td>
<td>1.054</td>
<td>83</td>
<td>2.1 GHz</td>
<td>2.91</td>
<td>262</td>
</tr>
<tr>
<td>6</td>
<td>Center for Scientific Computing (CEA)</td>
<td>Roadrunner / IBM</td>
<td>France</td>
<td>118,368</td>
<td>1.050</td>
<td>84</td>
<td>2.1 GHz</td>
<td>4.59</td>
<td>219</td>
</tr>
<tr>
<td>7</td>
<td>DOE / NERSC National Energy Research Supercomputing Center</td>
<td>Jaguar / ORNL</td>
<td>USA</td>
<td>102,400</td>
<td>1.04</td>
<td>76</td>
<td>2.1 GHz</td>
<td>4.04</td>
<td>261</td>
</tr>
<tr>
<td>8</td>
<td>Japan Atomic Energy Agency</td>
<td>Roadrunner / IBM</td>
<td>Japan</td>
<td>98,938</td>
<td>0.83</td>
<td>81</td>
<td>2.1 GHz</td>
<td>2.91</td>
<td>262</td>
</tr>
<tr>
<td>9</td>
<td>Forschungszentrum Jülich</td>
<td>BlueGene/L</td>
<td>Germany</td>
<td>294,912</td>
<td>8.05</td>
<td>61</td>
<td>2.1 GHz</td>
<td>2.91</td>
<td>262</td>
</tr>
<tr>
<td>10</td>
<td>DOE / NERSC National Energy Research Supercomputing Center</td>
<td>BlueGene/L</td>
<td>USA</td>
<td>107,152</td>
<td>0.817</td>
<td>79</td>
<td>2.1 GHz</td>
<td>2.91</td>
<td>262</td>
</tr>
</tbody>
</table>

### China

- **Has 3 Pflops systems**
  - NUDT, Tianhe-1A, located in Tianjin
    - Dual-Intel 6 core + Nvidia Fermi w/ custom interconnect
    - Budget: 600 M RMB
    - DOE: NERSC National Energy Research Supercomputing Center
  - CIT, Dawning 6000, Neubula, located in Shenzhen
    - Dual-Intel 6 core + Nvidia Fermi w/QDR Infiniband
    - Budget: 600 M RMB
    - AC: NERSC National Energy Research Supercomputing Center
  - Mole-8.5 Cluster/320x2 Intel QC Xeon E5520 2.26 GHz + 320x Nvidia Tesla C2050/QDR Infiniband
- Fourth one planned for Shandong

### Tianhe-1A

**Main configuration of TH-1A system**

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute nodes (TH-X5670-FEP)</td>
<td>7,168</td>
</tr>
<tr>
<td>6-core CPU and 1 GPU per node</td>
<td>2 x 6-core CPU and 1 GPU per node</td>
</tr>
<tr>
<td>CPU: Xeon X5670 (Westmere)</td>
<td>2 x 6-core CPU and 1 GPU per node</td>
</tr>
<tr>
<td>Processor speed: 2.835 GHz</td>
<td>2 x 6-core CPU and 1 GPU per node</td>
</tr>
<tr>
<td>GPU: NVIDIA M2090</td>
<td>2 x 6-core CPU and 1 GPU per node</td>
</tr>
<tr>
<td>Connected with CPU by PCIe</td>
<td>2 x 6-core CPU and 1 GPU per node</td>
</tr>
<tr>
<td>32GB memory per node</td>
<td>2 x 6-core CPU and 1 GPU per node</td>
</tr>
<tr>
<td>71016 (32-bit) x 12 PGFlops or 4.7 Teraflops</td>
<td>2 x 6-core CPU and 1 GPU per node</td>
</tr>
<tr>
<td>Total: 4.791,451 (TGFlops)</td>
<td>2 x 6-core CPU and 1 GPU per node</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute nodes x 10 GPU</td>
<td>71016 (32-bit) x 12 PGFlops or 4.7 Teraflops</td>
</tr>
<tr>
<td>Total: 4.791,451 (TGFlops)</td>
<td>2 x 6-core CPU and 1 GPU per node</td>
</tr>
</tbody>
</table>
Chinese Multicore Development

- Loongson (Chinese: 龙芯; academic name: Godson, also known as Dragon chip) is a family of general-purpose MIPS-compatible CPUs developed at the Institute of Computing Technology, Chinese Academy of Sciences.
- The chief architect is Professor Weiwu Hu.
- The 65 nm Loongson 3 (Godson-3) is able to run at a clock speed between 1.0 to 1.2 GHz, with 4 CPU cores (10W) first and 8 cores later (20W), and it is expected to debut in 2010.
- Will use this chip as basis for Petascale system in 2010.

NSF Supercomputing Centers

University of Illinois - Blue Waters will be the powerhouse of the National Science Foundation’s strategy to support supercomputers for scientists nationwide.

<table>
<thead>
<tr>
<th>T1</th>
<th>Blue Waters</th>
<th>NCSA/Illinois</th>
<th>10 Pflop/s peak; 1 Pflop/s sustained per second in 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2</td>
<td>Kraken</td>
<td>NICS/U of Tennessee</td>
<td>1 Pflop/s peak per second</td>
</tr>
<tr>
<td></td>
<td>Ranger</td>
<td>TACC/U of Texas</td>
<td>564 Tflop/s peak per second</td>
</tr>
<tr>
<td>T3</td>
<td>Campuses across the U.S.</td>
<td>Several sites</td>
<td>50-100 Tflop/s peak per second</td>
</tr>
</tbody>
</table>

General Purpose CPU Concurrency Trends

- Today
  - Typical server node chip - 8 cores
  - 1k node cluster - 8,000 cores
  - Laptop - 2 cores (low power)
- By 2020
  - Typical server node chip - 400 cores
  - 1k node cluster - 400,000 cores
  - Laptop - 100 cores (low power)

Future Computer Systems

- Most likely be a hybrid design
- Think standard multicore chips and accelerator (GPUs)
- Today accelerators are attached
- Next generation more integrated
- Intel’s “Knights Corner” and “Knights Ferry” to come.
  - 48 x86 cores
- AMD’s Fusion in 2011 - 2013
  - Multicore with embedded graphics ATI
- Nvidia’s Project Denver plans to develop an integrated chip using ARM architecture in 2013.
**Evolution of GPUs**

**GPUs**: excelling in graphics rendering

<table>
<thead>
<tr>
<th>Scene model</th>
<th>Graphics pipelined</th>
<th>Final image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeated fast over and over: e.g. TV refresh rate is 30 fps; limit is 60 fps</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This type of computation:
- Requires enormous computational power
- Allows for high parallelism
- Needs high bandwidth vs low latency
  (as low latencies can be compensated with deep graphics pipeline)

Obviously, this pattern of computation is common with many other applications

---

**Challenges of using GPUs**

- **High levels of parallelism**
  - Many GPU cores, serial kernel execution
    - e.g. 240 in the Nvidia Tesla; up to 512 in Fermi - to have concurrent kernel execution

- **Hybrid/heterogeneous architectures**
  - Match algorithmic requirements to architectural strengths
    - e.g. small, non-parallelizable tasks to run on CPU, large and parallelizable on GPU

- **Compute vs communication gap**
  - Exponentially growing gap; persistent challenge
    - Processor speed improves 59%, memory bandwidth 23%, latency 5.5% (on all levels, e.g. a GPU Tesla C1070 (4 x C1060) has compute power $O(1,000)$ Gflop/s but GPUs communicate through the CPU using $O(1)$ GB/s connection)

---

**Moore’s Law is Alive and Well**

- Transistors (in Thousands)
- Frequency (MHz)
- Power (W)

- 15 Years of exponential growth ~2x year has ended

Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanović
Slide from Kathy Yelick

---

**But Clock Frequency Scaling Replaced by Scaling Cores / Chip**

- Transistors (in Thousands)
- Frequency (MHz)
- Power (W)
- Cores

- Power is the root cause of all this

Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanović
Slide from Kathy Yelick

---

**Performance Has Also Slowed, Along with Power**

- Power is the root cause of all this

Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanović
Slide from Kathy Yelick

---

**Power Cost of Frequency**

- Power $\propto$ Voltage$^2$ x Frequency (V$^2$F)
- Frequency $\propto$ Voltage
- Power $\propto$ Frequency$^3$

<table>
<thead>
<tr>
<th>Core</th>
<th>V</th>
<th>Freq</th>
<th>Perf</th>
<th>Power</th>
<th>PE (Supercomputer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superscalar</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New Superscalar</td>
<td>1X</td>
<td>1.5X</td>
<td>1.5X</td>
<td>3.3X</td>
<td>0.45X</td>
</tr>
</tbody>
</table>

Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanović
Slide from Kathy Yelick
Power Cost of Frequency

- Power ∝ Voltage² x Frequency (V²F)
- Frequency ∝ Voltage
- Power ∝ Frequency²

<table>
<thead>
<tr>
<th></th>
<th>Cores</th>
<th>V</th>
<th>Freq</th>
<th>Perf</th>
<th>Power</th>
<th>FE max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supercal</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>&quot;New&quot; Supercal</td>
<td>1X</td>
<td>1.5X</td>
<td>1.5X</td>
<td>1.6X</td>
<td>3.3X</td>
<td>0.45X</td>
</tr>
<tr>
<td>Multicore</td>
<td>2X</td>
<td>0.75X</td>
<td>0.75X</td>
<td>1.5X</td>
<td>0.6X</td>
<td>1.85X</td>
</tr>
</tbody>
</table>

50% more performance with 20% less power
Preferable to use multiple slower devices, than one superfast device

Moore’s Law Reinterpreted

- Number of cores per chip doubles every 2 year, while clock speed decreases (not increases).
  - Need to deal with systems with millions of concurrent threads
    - Future generation will have billions of threads!
  - Need to be able to easily replace inter-chip parallelism with intro-chip parallelism
- Number of threads of execution doubles every 2 year

Major Changes to Software

- Must rethink the design of our software
  - Another disruptive technology
    - Similar to what happened with cluster computing and message passing
  - Rethink and rewrite the applications, algorithms, and software

Exascale Computing

- Exascale systems are likely feasible by 2017±2
  - 10-100 Million processing elements (cores or mini-cores) with chips perhaps as dense as 1,000 cores per socket, clock rates will grow more slowly
  - 3D packaging likely
  - Large-scale optics based interconnects
  - 10-100 PB of aggregate memory
  - Hardware and software based fault management
  - Heterogeneous cores
  - Performance per watt – stretch goal 100 GF/watt of sustained performance
  - Power, area and capital costs will be significantly higher than for today’s fastest systems

Factors that Necessitate Redesign of Our Software

- Steepness of the ascent from terascale to petascale to exascale
  - Preparing for million/billion way parallelism
- Tightening memory/bandwidth bottleneck
  - Limits on power/clock speed implication on multicore
  - Reducing communication will become much more intense
  - Memory per core changes, byte-to-flop ratio will change
- Necessary Fault Tolerance
  - MTTF will drop
  - Checkpoint/restart has limitations

Software infrastructure does not exist today
Conclusions

- For the last decade or more, the research investment strategy has been overwhelmingly biased in favor of hardware.
- This strategy needs to be rebalanced - barriers to progress are increasingly on the software side.
- Moreover, the return on investment is more favorable to software.
  - Hardware has a half-life measured in years, while software has a half-life measured in decades.
- High Performance Ecosystem out of balance
  - Hardware, OS, Compilers, Software, Algorithms, Applications
  - No Moore’s Law for software, algorithms and applications

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>Technology</th>
<th>Speed</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>L1 cache</td>
<td>2-3</td>
<td>KB</td>
</tr>
<tr>
<td>L2 cache</td>
<td>6-12</td>
<td>MB</td>
</tr>
<tr>
<td>L3 cache</td>
<td>14-40</td>
<td>GB</td>
</tr>
<tr>
<td>Near memory</td>
<td>100-300</td>
<td>TB</td>
</tr>
<tr>
<td>Far memory</td>
<td>300-900</td>
<td></td>
</tr>
<tr>
<td>Remote memory</td>
<td>$O(10^4)$</td>
<td></td>
</tr>
<tr>
<td>Message-passing</td>
<td>$O(10^5)$-$O(10^6)$</td>
<td></td>
</tr>
</tbody>
</table>
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.

Increasing the number of gates into a tight knot and decreasing the cycle time of the processor.

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

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

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...

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 issue
**Processor-DRAM Memory Gap**

- **μProc 60%/yr.** (2X/1.5yr)
- **DRAM 9%/yr.** (2X/10 yrs)
- "Moore’s Law"
- Processor-Memory Performance Gap:
  - (grows 50% / year)

**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**

**Principles of Parallel Computing**

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

All of these things make parallel programming even harder than sequential programming.

**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
  - $N =$ number of processors
- Even if the parallel part speeds up perfectly may be limited by the sequential part

**Amdahl’s Law**

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:

\[ t_N = \frac{t_S}{N + \frac{f_s}{N}} \]

Effect of multiple processors on run time

\[ S = \frac{1}{f_s + \frac{f_p}{N}} \]

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

**“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.
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.

Illustration of Amdahl’s Law

- 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

Overhead of Parallelism

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 $\$$
- 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 w/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
Virtual Environments

Do they make any sense?

Performance Improvements for Scientific Computing Problems

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 some type of interconnect network.

**Processors, Memory, & Networks**

- 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**

- **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.

- **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.

**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 scientist.
Highly Parallel Supercomputing: Where Are We?

- **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, HiPPI, 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