ECE 7660
Parallel Computer Architecture

Instructor: Dr. Song Jiang
The ECE Department
sjiang@eng.wayne.edu
http://www.ece.eng.wayne.edu/~sjiang/ECE7660-fall-15/ECE7660.htm

Lecture: M/W 5:30pm --- 7:20pm
028 MANO

Office hours: M/W 2:00pm --- 3:00pm or by appointments
Engineering Building, Room 3150
Today’s Goal:

- Introduce you to Parallel Computer Architecture
- Answer your questions about ECE7660
- Provide you a sense of the trends that shape the field
What will you get out of ECE7660?

• In-depth understanding of the design and engineering of modern parallel computers
  – technology forces
  – fundamental architectural issues
    » naming, replication, communication, synchronization
  – basic design techniques
    » cache coherence, protocols, networks, pipelining, …
  – methods of evaluation
  – underlying engineering trade-offs

• from moderate to very large scale
• across the hardware/software boundary
Will it be worthwhile?

• Absolutely!
  – even through few of you will become parallel architecture designers

• The fundamental issues and solutions translate across a wide spectrum of systems.
  – Crisp solutions in the context of parallel machines.

• Pioneered at the thin-end of the platform pyramid on the most-demanding applications
  – migrate downward with time

• Understand implications on software

![Diagram of platform pyramid]

- SuperServers
- Departmental Servers
- Workstations
- Personal Computers
The textbook

- **Parallel Computer Architecture: a Hardware/Software Approach**, Morgan Kaufmann Publishers  
  David E. Culler, University of California, Berkeley; Jaswinder Pal Singh, Princeton University; and Anoop Gupta, Stanford University

- **Book provides a framework and complete background.**
  - You do the reading
  - We’ll discuss it

- **Projects will go “beyond”**

The slides are adapted from the authors’ teaching materials.
How will grading work?

- 10%: Class attendance
- 15%: Homework assignments
- 15%: Lab projects
- 20%: Midterm
- 25%: Final
- 15%: Presentation
What is Parallel Architecture?

• A parallel computer is a collection of processing elements that cooperate to solve large problems fast

• Some broad issues:
  – Resource Allocation:
    » how large a collection?
    » how powerful are the elements?
    » how much memory?
  – Data access, Communication and Synchronization
    » how do the elements cooperate and communicate?
    » how are data transmitted between processors?
    » what are the abstractions and primitives for cooperation?
  – Performance and Scalability
    » how does it all translate into performance?
    » how does it scale?
Why Study Parallel Architecture?

Role of a computer architect:

To design and engineer the various levels of a computer system to maximize *performance* and *programmability* within limits of *technology* and *cost*.

Parallelism:

- Provides alternative to faster clock for performance
- Applies at all levels of system design
- Is a fascinating perspective from which to view architecture
- Is increasingly central in information processing
Why Study it Today?

• History: diverse and innovative organizational structures, often tied to novel programming models

• Rapidly maturing under strong technological constraints
  – The “killer micro” is ubiquitous
  – Laptops and supercomputers are fundamentally similar!
  – Technological trends cause diverse approaches to converge

• Technological trends make parallel computing inevitable

• Need to understand fundamental principles and design tradeoffs, not just taxonomies
  – Naming, Ordering, Replication, Communication performance
Is Parallel Computing Inevitable?

• Application demands: Our insatiable need for computing cycles
• Technology Trends
• Architecture Trends
• Economics
• Current trends:
  – Servers and workstations becoming MP: Sun, SGI, DEC, COMPAQ!...
  – Today’s microprocessors are multiprocessors
Application Trends

• Application demand for performance fuels advances in hardware, which enables new apps, which...
  – Cycle drives exponential increase in microprocessor performance
  – Drives parallel architecture harder
    » most demanding applications

• Range of performance demands
  – Need range of system performance with progressively increasing cost
Speedup

- Speedup (p processors) = \frac{Performance (p \text{ processors})}{Performance (1 \text{ processor})}

- For a fixed problem size (input data set), performance = \frac{1}{time}

- Speedup fixed problem (p processors) = \frac{Time (1 \text{ processor})}{Time (p \text{ processors})}
Commercial Computing

• Relies on parallelism for high end
  – Computational power determines scale of business that can be handled

• Databases, online-transaction processing, decision support, data mining, data warehousing ...

• TPC benchmarks (TPC-C order entry, TPC-D decision support)
  – Explicit scaling criteria provided
  – Size of enterprise scales with size of system
  – Problem size not fixed as $p$ increases.
  – Throughput is performance measure (transactions per minute or tpm)
Scientific Computing Demand

Grand Challenge problems
- Global change
- Human genome
- Fluid turbulence
- Vehicle dynamics
- Ocean circulation
- Viscous fluid dynamics
- Superconductor modeling
- Quantum chromo dynamics
- Vision

<table>
<thead>
<tr>
<th>Storage requirement</th>
<th>Computational performance requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 TB</td>
<td>1 TFLOPS</td>
</tr>
<tr>
<td>100 GB</td>
<td>100 GFLOPS</td>
</tr>
<tr>
<td>10 GB</td>
<td>10 GFLOPS</td>
</tr>
<tr>
<td>1 GB</td>
<td>1 GFLOPS</td>
</tr>
<tr>
<td>100 MB</td>
<td>100 MFLOPS</td>
</tr>
<tr>
<td>10 MB</td>
<td>100 MFLOPS</td>
</tr>
</tbody>
</table>

- 2D airfoil
- Oil reservoir modeling
- 48-hour weather
- 3D plasma modeling
- 72-hour weather
- Vehicle signature
- Structural biology
- Pharmaceutical design
- Chemical dynamics
Engineering Computing Demand

• Large parallel machines are a mainstay in many industries
  – Petroleum (reservoir analysis)
  – Automotive (crash simulation, drag analysis, combustion efficiency),
  – Aeronautics (airflow analysis, engine efficiency, structural mechanics, electromagnetism),
  – Computer-aided design
  – Pharmaceuticals (molecular modeling)
  – Visualization
    » in all of the above
    » entertainment (films like Toy Story)
    » architecture (walk-throughs and rendering)
  – Financial modeling (yield and derivative analysis)
  – etc.
Is better parallel arch enough?

- AMBER molecular dynamics simulation program
- Starting point was vector code for Cray-1
- 145 MFLOPS on Cray90, 406 for final version on 128-processor Paragon, 891 on 128-processor Cray T3D
Summary of Application Trends

• Transition to parallel computing has occurred for scientific and engineering computing
• In rapid progress in commercial computing
  – Database and transactions as well as financial
  – Usually smaller-scale, but large-scale systems also used
• Desktop also uses multithreaded and parallel programs on multicores.
• Demand for improving throughput on sequential workloads
  – Greatest use of small-scale multiprocessors
• Solid application demand exists and will increase
Technology Trends

- Today the natural building-block is also fastest!
Can’t we just wait for it to get faster?

- **Microprocessor performance** increases 50% - 100% per year
- **Transistor count** doubles every 3 years
- **DRAM size** quadruples every 3 years
- Huge investment per generation is carried by huge commodity market

![Graph showing performance improvements over time]
Technology: A Closer Look

- Basic advance is *decreasing feature size* $(\lambda)$
  - Circuits become either faster or lower in power

“If the automobile industry advanced as rapidly as the semiconductor industry, a Rolls Royce would get $\frac{1}{2}$ million miles per gallon and it would be cheaper to throw it away than to park it.”

Gordon Moore,
Intel Corporation
• Die size is growing too
  – Clock rate improves roughly proportional to improvement in $\lambda$
  – Number of transistors improves like $\lambda^2$ (or faster)

• Performance > 100x per decade
  – clock rate < 10x, rest is transistor count

• How to use more transistors?
  – Parallelism in processing
    » multiple operations per cycle reduces CPI
  – Locality in data access
    » avoids latency and reduces CPI
    » also improves processor utilization
  – Both need resources, so tradeoff

• Fundamental issue is resource distribution, as in uniprocessors
Architectural Trends

• Architecture translates technology’s gifts into performance and capability

• Resolves the tradeoff between parallelism and locality
  – Current microprocessor: 1/3 compute, 1/3 cache, 1/3 off-chip connect
  – Tradeoffs may change with scale and technology advances

• Understanding microprocessor architectural trends
  => Helps build intuition about design issues or parallel machines
  => Shows fundamental role of parallelism even in “sequential” computers
Architectural Trends

- Greatest trend in VLSI generation is increase in parallelism
  - Up to 1985: bit level parallelism: 4-bit -> 8 bit -> 16-bit
    » slows after 32 bit
    » adoption of 64-bit now under way

  - Mid 80s to mid 90s: instruction level parallelism
    » pipelining and simple instruction sets + compiler advances (RISC)
    » on-chip caches and functional units = superscalar execution
    » greater sophistication: out of order execution, speculation, prediction
      - to deal with control transfer and latency problems
Why ILP is slowing

• Branch prediction accuracy is already > 90%
  – Hard to improve it even more

• Number of pipeline stages is already deep (~20-30 stages)
  – But critical dependence loops do not change
  – Memory latency requires more clock cycles to satisfy

• Processor width is already high
  – Quadratically increasing complexity to increase the width

• Cache size
  – Effective, but also shows diminishing returns
  – In general, the size must be doubled to reduce miss rate by a half
ILP tapped out + end of frequency scaling

Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)

Processor clock rate stops increasing

No further benefit from ILP

Image credit: “The free Lunch Is Over” by Herb Sutter, Dr. Dobbs 2005
The “power wall”

Per transistor:
Dynamic power $\propto \text{capacitive load} \times \text{voltage}^2 \times \text{frequency}$
Static power: transistors burn power even when inactive due to leakage

High power = high heat
Power is a critical design constraint in modern processors

<table>
<thead>
<tr>
<th>Hardware</th>
<th>TDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core i7 (in this laptop)</td>
<td>45W</td>
</tr>
<tr>
<td>Intel Core i7 2700K (fast desktop CPU)</td>
<td>95W</td>
</tr>
<tr>
<td>NVIDIA GTX 780 GPU</td>
<td>250W</td>
</tr>
<tr>
<td>Mobile phone processor</td>
<td>$\frac{1}{2}$ - 2W</td>
</tr>
<tr>
<td>World’s fastest supercomputer</td>
<td>megawatts</td>
</tr>
<tr>
<td>Standard microwave oven</td>
<td>700W</td>
</tr>
</tbody>
</table>

Source: Intel, NVIDIA, Wikipedia, Top500.org
Threads Level Parallelism on Chip with SMT

- **Simultaneous multithreading (SMT)** (or hardware multithreading, and called Hyper-threading at Intel) allows multiple independent threads of execution to better utilize the resources
  - **compared to** temporal multithreading, for which only one thread of instructions can execute in any given pipeline stage at a time.
Threads Level Parallelism on Chip with Multicore

Intel Haswell (2013)

- Quad-core CPU + multi-core GPU integrated on one chip
Threads Level Parallelism “on board”

- Microprocessor on a chip makes it natural to connect many to shared memory
  - dominates server and enterprise market, moving down to desktop
- Faster processors began to saturate bus, then bus technology advanced
Historical Perspectives

• 80s – early 90s: prime time for parallel architecture research
  – A microprocessor is not powerful enough, so naturally need multiple chips (and processors)
  – J-machine, M-machine, Alewife, Tera, HEP, etc.

• 90s: at the low end, uniprocessor systems’ speed grows much faster than parallel systems’ speed
  – A microprocessor fits on a chip. So do branch predictor, multiple functional units, large caches, etc!
  – Microprocessor also exploits parallelism (pipelining, multiple-issue, VLIW) – parallelisms originally invented for multiprocessors
  – Many parallel computer vendors went bankrupt
  – Prestigious but small high-performance computing market
Historical Perspectives (Cont’d)

• 90s: emergence of distributed (vs. parallel) machines
  – Progress in network technologies:
    » Network bandwidth grows faster than Moore’s law
    » Fast interconnection network getting cheap
  – Connects cheap uniprocessor systems into a large distributed machine
  – Network of Workstations, Clusters, GRID

• 00s: parallel architectures are back
  – Transistors per chip >> microproc transistors
  – Harder to get more performance from a uniprocessor
  – SMT (Simultaneous multithreading), CMP (Chip Multi-Processor), ultimately Massive CMP
  – E.g. Intel Pentium D, Core Duo, AMD Dual Core, IBM Power5, Sun Niagara, etc.
What about Storage Trends?

• Divergence between memory capacity and speed even more pronounced
  – Capacity increased by 1000x from 1980-95, speed only 2x
  – Gigabit DRAM by c. 2000, but gap with processor speed much greater

• Larger memories are slower, while processors get faster
  – Need to transfer more data in parallel
  – Need deeper cache hierarchies
  – How to organize caches?

• Parallelism increases effective size of each level of hierarchy, without increasing access time

• Parallelism and locality within memory systems too
  – New designs fetch many bits within memory chip; follow with fast pipelined transfer across narrower interface
  – Buffer caches most recently accessed data

• Disks too: Parallel disks plus caching
Economics

- Commodity microprocessors not only fast but CHEAP
  - Development costs tens of millions of dollars
  - BUT, many more are sold compared to supercomputers
  - Crucial to take advantage of the investment, and use the commodity building block

- Multiprocessors being pushed by software vendors (e.g. database) as well as hardware vendors

- Standardization makes small, bus-based SMPs commodity

- Desktop: few smaller processors versus one larger one?

- Multiprocessor on a chip? Yes!
Consider Scientific Supercomputing

• Proving ground and driver for innovative architecture and techniques
  – Market smaller relative to commercial as MPs become mainstream
  – Dominated by vector machines starting in 70s
  – Microprocessors have made huge gains in floating-point performance
    » high clock rates
    » pipelined floating point units (e.g., multiply-add every cycle)
    » instruction-level parallelism
    » effective use of caches (e.g., automatic blocking)
  – Plus economics

• Large-scale multiprocessors replace vector supercomputers
## 500 Fastest Computers ([http://www.top500.org](http://www.top500.org))

<table>
<thead>
<tr>
<th>RANK</th>
<th>SITE</th>
<th>SYSTEM</th>
<th>CORES</th>
<th>RMAX (TFLOP/S)</th>
<th>RPEAK (TFLOP/S)</th>
<th>POWER (KW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Super Computer Center in Guangzhou China</td>
<td>Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT</td>
<td>3,120,000</td>
<td>33,862.7</td>
<td>54,902.4</td>
<td>17,808</td>
</tr>
<tr>
<td>2</td>
<td>DOE/SC/Oak Ridge National Laboratory United States</td>
<td>Titan - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.</td>
<td>560,640</td>
<td>17,590.0</td>
<td>27,112.5</td>
<td>8,209</td>
</tr>
<tr>
<td>3</td>
<td>DOE/NNSA/LLNL United States</td>
<td>Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM</td>
<td>1,572,864</td>
<td>17,173.2</td>
<td>20,132.7</td>
<td>7,890</td>
</tr>
<tr>
<td>4</td>
<td>RIKEN Advanced Institute for Computational Science (AICS) Japan</td>
<td>K computer, SPARC64 VIII fx 2.0GHz, Tofu interconnect Fujitsu</td>
<td>705,024</td>
<td>10,510.0</td>
<td>11,280.4</td>
<td>12,660</td>
</tr>
<tr>
<td>5</td>
<td>DOE/SC/Argonne National Laboratory United States</td>
<td>Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM</td>
<td>786,432</td>
<td>8,586.6</td>
<td>10,066.3</td>
<td>3,945</td>
</tr>
</tbody>
</table>
Summary: Why Parallel Architecture?

• Increasingly attractive
  – Economics, technology, architecture, application demand

• Increasingly central and mainstream

• Parallelism exploited at many levels
  – Instruction-level parallelism
  – Multiprocessor servers
  – Large-scale multiprocessors ("MPPs")

• Focus of this class: multiprocessor level of parallelism

• Same story from memory system perspective
  – Increase bandwidth, reduce average latency with many local memories

• Spectrum of parallel architectures make sense
  – Different cost, performance and scalability