Lecture 7
Thread Level Parallelism (1)

EEC 171 Parallel Architectures
John Owens
UC Davis
Credits

What We Know

• What new techniques have we learned that make ...
  • ... control go fast?
  • ... datapath go fast?
Cook analogy

• We want to prepare food for several banquets, each of which requires many dinners.

• We have two positions we can fill:
  • The boss (control), who gets all the ingredients and tells the chef what to do
  • The chef (datapath), who does all the cooking

• ILP is analogous to:
  • One ultra-talented boss with many hands
  • One ultra-talented chef with many hands
Cook analogy

- We want to prepare food for several banquets, each of which requires many dinners.
- But one boss and one chef isn’t enough to do all our cooking.
- What are our options?
Chef scaling

- What’s the cheapest way to cook more?
- Is it easy or difficult to share (ingredients, cooked food, etc.) between chefs?
- Which method of scaling is most flexible?
“Sea change in computing”

- “... today’s processors ... are nearing an impasse as technologies approach the speed of light...”
- Transputer had bad timing (uniprocessor performance increased)
  → Procrastination rewarded: 2X seq. perf. / 1.5 years
- “We are dedicating all of our future product development to multicore designs ... This is a sea change in computing.”
  - Paul Otellini, President, Intel (2005)
- All microprocessor companies switch to MP (2X CPUs / 2 yrs)
  → Procrastination penalized: 2X sequential perf. / 5 yrs
Flynn’s Classification Scheme

- **SISD** – single instruction, single data stream
  - Uniprocessors

- **SIMD** – single instruction, multiple data streams
  - single control unit broadcasting operations to multiple datapaths

- **MISD** – multiple instruction, single data
  - no such machine (although some people put vector machines in this category)

- **MIMD** – multiple instructions, multiple data streams
  - aka multiprocessors (SMPs, MPPs, clusters, NOWs)
Performance beyond single thread ILP

- There can be much higher natural parallelism in some applications (e.g., database or scientific codes)
- Explicit **Thread Level Parallelism** or **Data Level Parallelism**
- **Thread**: process with own instructions and data
  - Thread may be a subpart of a parallel program (“thread”), or it may be an independent program (“process”)
  - Each thread has all the state (instructions, data, PC, register state, and so on) necessary to allow it to execute
  - Many kitchens, each with own boss and chef
- **Data Level Parallelism**: Perform identical operations on data, and lots of data
  - 1 kitchen, 1 boss, many chefs
Continuum of Granularity

- **“Coarse”**
  - Each processor is more powerful
  - Usually fewer processors
  - Communication is more expensive between processors
  - Processors are more loosely coupled
  - Tend toward MIMD

- **“Fine”**
  - Each processor is less powerful
  - Usually more processors
  - Communication is cheaper between processors
  - Processors are more tightly coupled
  - Tend toward SIMD
The Rest of the Class

- Next 3 weeks:

- 3 weeks hence:
  - Data-level parallelism. Fine-grained parallelism. MMX, SSE. Vector & stream processors. GPUs.
Thread Level Parallelism

• ILP exploits implicit parallel operations within a loop or straight-line code segment

• TLP explicitly represented by the use of multiple threads of execution that are inherently parallel
  
  • You must **rewrite your code to be thread-parallel**.

• Goal: Use multiple instruction streams to improve
  
  • Throughput of computers that run many programs
  
  • Execution time of multi-threaded programs

• TLP could be more cost-effective to exploit than ILP
Organizing Many Processors

- Multiprocessor—multiple processors with a single shared address space
- Symmetric multiprocessors: All memory is the same distance away from all processors (UMA = uniform memory access)
Organizing Many Processors

- Cluster—multiple computers (each with their own address space) connected over a local area network (LAN) functioning as a single system
- “Constellation”: cluster of multiprocessors
Applications Needing “Supercomputing”

- Energy [plasma physics (simulating fusion reactions), geophysical (petroleum) exploration]
- DoE stockpile stewardship (to ensure the safety and reliability of the nation’s stockpile of nuclear weapons)
- Earth and climate (climate and weather prediction, earthquake, tsunami prediction and mitigation of risks)
- Transportation (improving vehicles’ airflow dynamics, fuel consumption, crashworthiness, noise reduction)
- Bioinformatics and computational biology (genomics, protein folding, designer drugs)
- Societal health and safety (pollution reduction, disaster planning, terrorist action detection)
- Financial (calculate options pricing, etc.)
In the last 8 years uniprocessor and SIMDs disappeared while Clusters and Constellations grew from 3% to 80%.

Top 500: Application Area

Application Area Share Over Time
1993-2008

- Not Specified
- Geophysics
- Finance
- Telecom
- Weather and Climate
- Research
- Automotive
- Database
- Aerospace
- Semiconductor
- Electronics
- Information Processing
- Service
- Energy
- Others
Top 500: Historicals
Top 500: Countries
Top 500: Customers
Top 500: Interconnect

Interconnect / Systems
November 2008

- Gigabit Ethernet
- Infiniband
- Others
- XT3 Internal Interconnect
- Myrinet
- Federation
- Infiniband DDR 4x
- XT4 Internal Interconnect
- Proprietary
- Infiniband DDR
Top 500: Processor Family

Processor Family / Systems (November 2006)
November 2006

Processor Family / Systems
November 2008
Top 500: Processor Count

Number of Processors / Systems
November 2008

- 2049-4096
- 4k-8k
- 1025-2048
- 8k-16k
- 16k-32k
- 32k-64k
- Others
For most apps, most execution units lie idle.

## Source of Wasted Slots

<table>
<thead>
<tr>
<th>Source of Wasted Issue Slots</th>
<th>Possible Latency-Hiding or Latency-Reducing Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>instruction tlb miss, data tlb miss</td>
<td>decrease the TLB miss rates (e.g., increase the TLB sizes); hardware instruction prefetching; hardware or software data prefetching; faster servicing of TLB misses</td>
</tr>
<tr>
<td>I cache miss</td>
<td>larger, more associative, or faster instruction cache hierarchy; hardware instruction prefetching</td>
</tr>
<tr>
<td>D cache miss</td>
<td>larger, more associative, or faster data cache hierarchy; hardware or software prefetching; improved instruction scheduling; more sophisticated dynamic execution</td>
</tr>
<tr>
<td>branch misprediction</td>
<td>improved branch prediction scheme; lower branch misprediction penalty</td>
</tr>
<tr>
<td>control hazard</td>
<td>speculative execution; more aggressive if-conversion</td>
</tr>
<tr>
<td>load delays (first-level cache hits)</td>
<td>shorter load latency; improved instruction scheduling; dynamic scheduling</td>
</tr>
<tr>
<td>short integer delay</td>
<td>improved instruction scheduling</td>
</tr>
<tr>
<td>long integer, short fp, long fp delays</td>
<td>(multiply is the only long integer operation, divide is the only long floating point operation) shorter latencies; improved instruction scheduling</td>
</tr>
<tr>
<td>memory conflict</td>
<td>(accesses to the same memory location in a single cycle) improved instruction scheduling</td>
</tr>
</tbody>
</table>
Single-threaded CPU

Introduction to Multithreading, Superthreading and Hyperthreading
By Jon Stokes
http://arstechnica.com/articles/paedia/cpu/hyperthreading.ars
We can add more CPUs ...

- ... and we’ll talk about this later in the class
- Note we have multiple CPUs reading out of the same instruction store
- Is this more efficient than having one CPU?
Symmetric Multiprocessing
Conventional Multithreading

- How does a microprocessor run multiple processes / threads “at the same time”?
- How does one program interact with another program?
- What is preemptive multitasking vs. cooperative multitasking?
New Approach: Multithreaded Execution

- Multithreading: multiple threads to share the functional units of 1 processor via overlapping
  - processor must duplicate independent state of each thread e.g., a separate copy of register file, a separate PC, and for running independent programs, a separate page table
  - memory shared through the virtual memory mechanisms, which already support multiple processes
  - HW for fast thread switch; much faster than full process switch \( \approx 100\text{s to 1000s of clocks} \)
Superthreading
Simultaneous multithreading (SMT)
Simultaneous multithreading (SMT)
## Multithreaded Categories

<table>
<thead>
<tr>
<th>Superscalar</th>
<th>Fine-Grained</th>
<th>Coarse-Grained</th>
<th>Multiprocessing</th>
<th>Simultaneous Multithreading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread 1</td>
<td>Thread 3</td>
<td>Thread 4</td>
<td>Thread 5</td>
<td>Thread 2</td>
</tr>
</tbody>
</table>
“Hyperthreading”

http://www.2cpu.com/Hardware/ht_analysis/images/taskmanager.html
Multithreaded Execution

• When do we switch between threads?
  • Alternate instruction per thread (fine grain)
  • When a thread is stalled, perhaps for a cache miss, another thread can be executed (coarse grain)
Fine-Grained Multithreading

- Switches between threads on each instruction, causing the execution of multiple threads to be interleaved
- Usually done in a round-robin fashion, skipping any stalled threads
- CPU must be able to switch threads every clock
- Advantage is it can hide both short and long stalls, since instructions from other threads executed when one thread stalls
- Disadvantage is it slows down execution of individual threads, since a thread ready to execute without stalls will be delayed by instructions from other threads
- Used on Sun’s Niagara (will see later)
Coarse-Grained Multithreading

- Switches threads only on costly stalls, such as L2 cache misses

- Advantages
  - Relieves need to have very fast thread-switching
  - Doesn’t slow down thread, since instructions from other threads issued only when the thread encounters a costly stall

- Disadvantage is hard to overcome throughput losses from shorter stalls, due to pipeline start-up costs
  - Since CPU issues instructions from 1 thread, when a stall occurs, the pipeline must be emptied or frozen
  - New thread must fill pipeline before instructions can complete

- Because of this start-up overhead, coarse-grained multithreading is better for reducing penalty of high cost stalls, where pipeline refill << stall time

- Used in IBM AS/400
P4Xeon Microarchitecture

- Replicated
  - Register renaming logic
  - Instruction pointer, other architectural registers
  - ITLB
  - Return stack predictor

- Partitioned
  - Reorder buffers
  - Load/store buffers
  - Various queues: scheduling, uop, etc.

- Shared
  - Caches (trace, L1/L2/L3)
  - Microarchitectural registers
  - Execution units

- If configured as single-threaded, all resources go to one thread
Partitioning: Static vs. Dynamic
Design Challenges in SMT

- Since SMT makes sense only with fine-grained implementation, impact of fine-grained scheduling on single thread performance?
  - A preferred thread approach sacrifices neither throughput nor single-thread performance?
  - Unfortunately, with a preferred thread, the processor is likely to sacrifice some throughput, when preferred thread stalls

- Larger register file needed to hold multiple contexts

- Not affecting clock cycle time, especially in
  - Instruction issue—more candidate instructions need to be considered
  - Instruction completion—choosing which instructions to commit may be challenging

- Ensuring that cache and TLB conflicts generated by SMT do not degrade performance
Problems with SMT

• One thread monopolizes resources
  • Example: One thread ties up FP unit with long-latency instruction, other thread tied up in scheduler

• Cache effects
  • Caches are unaware of SMT—can’t make warring threads cooperate
  • If both warring threads access different memory and have cache conflicts, constant swapping
Hyperthreading Neutral!

![Bar chart showing LAME 3.92MMX performance with and without Hyperthreading]

Seconds to Encode (Lower is Better)

- HT - 2.0ghz: 58 seconds
- No HT - 2.0ghz: 58 seconds
- HT - 2.4ghz: 49 seconds
- No HT - 2.4ghz: 49 seconds

Source: [http://www.2cpu.com/articles/43_1.html](http://www.2cpu.com/articles/43_1.html)
Hyperthreading Good!

[Bar chart showing the comparison of TMPGEnc - MPEG1 encoding times with and without hyperthreading for different clock speeds. The chart indicates that hyperthreading results in faster encoding times.]

http://www.2cpu.com/articles/43_1.html
Hyperthreading Bad!

DivX Pro v5.0.2 Encoding

Frames/Second (Higher is Better)

http://www.2cpu.com/articles/43_1.html
SPEC vs. SPEC (PACT ‘03)

- Avg. multithreaded speedup 1.20 (range 0.90–1.58)

“Initial Observations of the Simultaneous Multithreading Pentium 4 Processor”, Nathan Tuck and Dean M. Tullsen (PACT ‘03)