Final Review

CSE 471 Spring 2015

Mark Wyse

June 04, 2015

Lecture Critique

- Don’t forget!
- 1 page, summarize important points + your thoughts
- Exhibit good writing skills

Topics

- Multiprocessors
- Cache Coherence
- Synchronization
- MP Big Issues
- Multithreading
- Dataflow & Wavescalar
- GPUs

Multiprocessors

- Motivation:
  - Lots of transistors
  - Scaling (Moore and Denard) starting to fall over
  - Workload opportunities: scientific, server, media
- Two key types:
  - Bus based (low-end)
    - Simpler, physically centralized memory, UMA
  - Multiple-path interconnect (high-end)
    - Complex, scalable, physically distributed memory, NUMA

Cache Coherence

- Cache Coherence Problem: if core X writes address A, then core Y reads address A, what value does it read?
  - Assuming A was cached locally by both X and Y

A Definition of Coherence

1. Program Order
2. Coherent View of Memory
3. Write Serialization

Coherence

- Protocols
  - MSI
  - MESI
- Bus-based (snooping) vs Directory
  - Differences
  - Pros and cons (performance, scalability, etc.)
- Hardware support required

Synchronization

- Coherency protocols <u>do not</u> regulate access to shared data
- Critical sections
  - Mutual exclusion between threads
- Barriers
  - Point in execution which all threads must reach before any can proceed
- Locking
  - Atomic read-modify-write
  - Load-locked & Store Conditional
- Synchronization APIs
  - Spin locks
  - Blocking locks
  - Queueing locks

MP Big Issues

- Programming model for interprocessor communication
  - Shared Memory
  - Message Passing
- Execution Model
  - Control parallel
  - Data parallel
  - Dataflow
- Expressing error-free parallelism (hard)
  - How do we guarantee/maintain correctness of parallelism?

Multithreading

- Execute multiple threads on the same processor without context switches
  - Hardware contexts!
- Motivation:
  - Performance
  - Instruction throughput not scaling with issue width
- Coarse-grain MT
  - Switch on long latency operations
- Fine-grain MT
  - Can switch to different thread each cycle
  - Cray (Tera) MTA
- Simultaneous Multithreading (SMT)
  - CMP reduces horizontal waste; FGMT reduces vertical waste
  - Same cycle multithreading

Cray (Tera) MTA

- Goals
  - UMA, lightweight synchronization, heterogeneous parallelism
- Interesting Features
  - FGMT: different thread each cycle, round-robin; processor state for 128 hardware contexts!
  - No data caches
  - No paging
  - VLIW instructions
  - Tagged memory
  - Trade-off between avoiding memory bank conflicts & exploiting spatial locality for data

SMT

- Same cycle multithreading
  - Convert TLP to ILP
- Thread-shared hardware resources
- Goals:
  - Throughput gains for multiple threads
  - Minimize single thread performance
  - Minimize microarchitectural impact on conventional OoO superscalar design
- Duplicate hardware
- No special hardware for scheduling from multiple threads!
- Comparison to CMP (pros and cons of each)

Dataflow

- Von Neumann vs. Dataflow Execution Models
- Dataflow:
  - No PC
  - No Register File
  - Parallel execution only hindered by data dependencies
  - Exploit ILP on a massive scale
- Dataflow Firing Rule: execute when operands arrived on all input arcs, place computed value on output arc.
- Steer and Merge operations
- Problems:
  - Memory ordering
  - Language compatibility
  - Scalability (token store and wires)

Wavescalar

- Solves dataflow issues:
  - Language compatibility & memory ordering
  - scalability
- Waves
  - loop free sections of dataflow graph
  - Wave number
- Memory ordering
  - Wave numbers
  - Sequence numbers within a wave
- Hierarchical microarchitecture

GPUs

- Data parallelism
- Identical, Independent, Streaming computations
- SIMT
  - Multicore, multithreaded SIMT
  - 100k’s threads
  - Multiple threads in lockstep (single PC for group of threads)
- Caches
  - Maximize throughput
  - Very little locality to exploit