CIS 631
Parallel Processing

Lecture 2: Parallel Architectures

Allen D. Malony
malony@cs.uoregon.edu

Department of Computer and Information Science
University of Oregon
Acknowledgements

- Portions of the lectures slides were adopted from:
  - David Patterson, CS252, University of California, Berkeley, 2001.
  - David E. Culler, CS 258, University of California, Berkeley, 1999.
Outline

- Parallel architecture types
- Instruction-level parallelism
- Vector processing
- SIMD
- Shared memory
  - Memory organization: UMA, NUMA
  - Coherency: CC-UMA, CC-NUMA
- Distributed memory
- Clusters
- Interconnection networks
Parallel Architecture Types

- **Uniprocessor**
  - Scalar processor
    - Processor
    - Memory
  - Vector processor
    - Processor
    - Vector
    - Memory
  - Single Instruction Multiple Data (SIMD)
    - Processor
    - Memory

- **Shared Memory Multiprocessor (SMP)**
  - Shared memory address space
    - Processor
    - Memory
  - Bus-based memory system
  - Interconnection network
    - Processor
    - Network
    - Memory
Introduction to Parallel Architectures

- Distributed Memory Multiprocessor
  - Message passing between nodes

  - Memory
  - Interconnection network
  - Processor
  - Memory

- Massively Parallel Processor (MPP)
  - Many, many processors

- Cluster of SMPs
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes

  - SMPs
  - Interconnection network

  - Can also be regarded as MPP if processor number is large
How do you get parallelism in the hardware?

- Instruction-Level Parallelism (ILP)
- Data parallelism
  - Increase amount of data that can be operated on
- Processor parallelism
  - Increase number of processors
- Memory system parallelism
  - Increase number of memory units
  - Increase bandwidth to memory
- Communication parallelism
  - Increase amount of interconnection between elements
  - Increase communication bandwidth
Instruction-Level Parallelism

- Opportunities for splitting up instruction processing
- Pipelining within instruction
- Pipelining between instructions
- Overlapped execution
- Multiple functional units
- Out of order execution
- Multi-issue execution
- Superscalar processing
- Superpipelining
- Very Long Instruction Word (VLIW)
Parallelism in Single Processor Computers

Unpipelined
- multiple E unit
  - CDC 6600
    - scoreboard
  - FPS AP-120B
    - VLIW

ILP
- only scalar instructions
  - horizontal control
    - CDC 6600
  - issue-when-ready
    - CDC 7600

Pipelined
- vector instructions
  - register to register
    - CRAY-1
  - memory to memory
    - CDC Cyber-205

Vector

CIS 631 - Parallel Processing
Vector Processing

- Scalar processing
  - Processor instructions operate on scalar values
  - Integer registers and floating point registers
- Vectors
  - Set of scalar data
  - Vector registers (integer, floating point (typically))
  - Vector instructions operate on vector registers (SIMD)
- Vector unit pipelining
- Multiple vector units
- Vector chaining

Cray 2

Liquid-cooled with inert fluorocarbon (That’s a waterfall fountain!!)
Data Parallel Architectures

- SIMD (Single Instruction Multiple Data)
  - Logical single thread (instruction) of control
  - Processor associated with data elements

- Architecture
  - Array of simple processors with memory
  - Processors arranged in a regular topology
  - Control processor issues instructions
    - All processors execute same instruction (maybe disabled)
  - Specialized synchronization and communication
  - Specialized reduction operations
  - Array processing
- Applied Memory Technology (AMT)
- Distributed Array Processor (DAP)
Thinking Machines Connection Machine

16,000 processors!!!

(Tucker, IEEE Computer, Aug. 1988)
Vector and SIMD Processing Timeline

(a) Multivector track

CDC 7600 (CDC, 1970)

CDC Cyber 205 (Levine, 1982)

Cray 1 (Russell, 1978)

Goodyear MPP (Batcher, 1980)

Illiac IV (Barnes et al, 1968)

BSP (Kuck and Stokes, 1982)

CM2 (TMC, 1990)

DAP 610 (AMT, Inc. 1987)

ETA 10 (ETA, Inc. 1989)

Cray Y-MP (Cray Research, 1989)

Cray/MPP (Cray Research, 1993)

MasPar MP1 (Nickolls, 1990)

IBM GF/11 (Beetem et al, 1985)

Fujitsu, NEC, Hitachi Models

(b) SIMD track
What’s the maximum parallelism in a program?


- Analyze the data dependencies in application execution.

Graph showing 512-point FFT.
Dataflow Architectures

- Represent computation as graph of dependencies
- Operations stored in memory until operands are ready
- Operations can be dispatched to processors
- Tokens carry tags of next instruction to processor
- Tag compared in matching store
- A match fires execution
- Machine does the hard parallelization work
- Hard to build right
Shared Physical Memory

- Add processors to single processor computer system
- Processors share computer system resources
  - Memory, storage, …
- Sharing physical memory
  - Any processor can reference any memory location
  - Any I/O controller can reference any memory address
  - Single physical memory address space
- Operating system runs on any processor, or all
  - OS see single memory address space
  - Uses shared memory to coordinate
- Communication occurs as a result of loads and stores
Caching in Shared Memory Systems

- Reduce average latency
  - automatic replication closer to processor
- Reduce average bandwidth
- Data is logically transferred from producer to consumer to memory
  - store reg --> mem
  - load reg <-- mem
- Processors can share data efficiently
- What happens when store and load are executed on different processors?
- Cache coherence problems
Shared Memory Multiprocessors (SMP)

- Architecture types
  - Single processor
    - P
    - M
  - Multiple processors
    - P
    - M
    - multi-port
    - P
    - M
    - shared bus
    - P
    - M
    - interconnection network

- Differences lie in memory system interconnection

What does this look like?
Bus-based SMP

- Memory bus handles all memory read/write traffic
- Processors share bus
- **Uniform Memory Access (UMA)**
  - Memory (not cache) uniformly equidistant
  - Take same amount of time (generally) to complete
- May have multiple memory modules
  - Interleaving of physical address space
- Caches introduce memory hierarchy
  - Lead to data consistency problems
  - Cache coherency hardware necessary (CC-UMA)
Crossbar SMP

- Replicates memory bus for every processor and I/O controller
  - Every processor has direct path
- UMA SMP architecture
- Can still have cache coherency issues
- Multi-bank memory or interleaved memory
- Advantages
  - Bandwidth scales linearly (no shared links)
- Problems
  - High incremental cost (cannot afford for many processors)
  - Use switched multi-stage interconnection network
“Dance Hall” SMP and Shared Cache

- Interconnection network connects processors to memory
- Centralized memory (UMA)
- Network determines performance
  - Continuum from bus to crossbar
  - Scalable memory bandwidth
- Memory is physically separated from processors
- Could have cache coherence problems
- Shared cache reduces coherence problem and provides fine grained data sharing
Center for Supercomputing Research and Development

- Multi-cluster scalable parallel computer
- Alliant FX/80
  - 8 processors w/ vectors
  - Shared cache
  - HW synchronization
- Omega switching network
- Shared global memory
- SW-based global memory coherency
Natural Extensions of the Memory System

Shared Cache

Crossbar, Interleaved

Centralized Memory Dance Hall, UMA

Distributed Memory (NUMA)
Non-Uniform Memory Access (NUMA) SMPs

- Distributed memory
- Memory is physically resident close to each processor
- Memory is still shared
- Non-Uniform Memory Access (UMA)
  - Local memory and remote memory
  - Access to local memory is faster, remote memory slower
  - Access is non-uniform
  - Performance will depend on data locality
- Cache coherency is still an issue (more serious)
- Interconnection network architecture is more scalable
Cache Coherency and SMPs

- Caches play key role in SMP performance
  - Reduce average data access time
  - Reduce bandwidth demands placed on shared interconnect

- Private processor caches create a problem
  - Copies of a variable can be present in multiple caches
  - A write by one processor may not become visible to others
    - They’ll keep accessing stale value in their caches
  - \( \Rightarrow \) Cache coherence problem

- What do we do about it?
  - Organize the memory hierarchy to make it go away
  - Detect and take actions to eliminate the problem
Definitions

- Memory operation (load, store, read-modify-write, …)
- Memory issue is operation presented to memory system
- Processor perspective
  - Write: subsequent reads return the value
  - Read: subsequent writes cannot affect the value
- Coherent memory system
  - There exists a serial order of memory operations on each location such that
    - operations issued by a process appear in order issued
    - value returned by each read is that written by previous write
  => write propagation + write serialization
Motivation for Memory Consistency

- Coherence implies that writes to a location become visible to all processors in the same order.
- But when does a write become visible?
- How do we establish orders between a write and a read by different processors?
  - Use event synchronization
- Implement hardware protocol for cache coherency
- Protocol will be based on model of memory consistency

```c
/* Assume initial value of A and flag is 0 */
A = 1;
while (flag == 0); /*spin idly*/
flag = 1;
print A;
```
Memory Consistency

- Specifies constraints on the order in which memory operations (from any process) can appear to execute with respect to each other
  - What orders are preserved?
  - Given a load, constrains the possible values returned by it
- Implications for both programmer and system designer
  - Programmer uses to reason about correctness
  - System designer can use to constrain how much accesses can be reordered by compiler or hardware
- Contract between programmer and system
Sequential Consistency

- Total order achieved by interleaving accesses from different processes
  - Maintains *program order*
  - Memory operations (from all processes) appear to issue, execute, and complete atomically with respect to others
  - As if there was a single memory (no cache)

“*A multiprocessor is sequentially consistent if the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program.*” [Lamport, 1979]
Sequential Consistency (Sufficient Conditions)

- There exist a total order consistent with the memory operations becoming visible in program order
- Sufficient Conditions
  - every process issues memory operations in program order
  - after write operation is issued, the issuing process waits for write to complete before issuing next memory operation (atomic writes)
  - after a read is issued, the issuing process waits for the read to complete and for the write whose value is being returned to complete (globally) before issuing its next memory operation
- Cache-coherent architectures implement consistency
Bus-based Cache-Coherent (CC) Architecture

- Bus Transactions
  - Single set of wires connect several devices
  - Bus protocol: arbitration, command/addr, data
  - Every device observes every transaction

- Cache block state transition diagram
  - FSM specifying how disposition of block changes
    - invalid, valid, dirty
  - Snoopy protocol

- Basic Choices
  - Write-through vs Write-back
  - Invalidate vs. Update
Snoopy Cache-Coherency Protocols

- Bus is a broadcast medium
- Caches know what they have
- Cache controller “snoops” all transactions on shared bus
  - relevant transaction if for a block its cache contains
  - take action to ensure coherence
    - invalidate, update, or supply value
  - depends on state of the block and the protocol
Example: Write-thru Invalidate
Intel Pentium Pro Quad Processor

- All coherence and multiprocessing glue in processor module
- Highly integrated, targeted at high volume
- Low latency and bandwidth
Next Class

- More cache-coherent SMPs
- Distributed memory parallel computers
- Clusters
- Clusters of SMPs