Outline

• Motivation & Applications

• Programming Models & A Generic Parallel Machine

• Issues in Programming Models
  – Function: naming, operations, & ordering
  – Performance: latency, bandwidth, etc.
Programming Model Design Issues

• **Naming**: How is communicated data and/or partner node referenced?

• **Operations**: What operations are allowed on named data?

• **Ordering**: How can producers and consumers of data coordinate their activities?

• **Performance**
  – **Latency**: How long does it take to communicate in a protected fashion?
  – **Bandwidth**: How much data can be communicated per second? How many operations per second?
Issue: Naming

- **Single Global Linear-Address-Space** (shared memory)
- **Multiple Local Address/Name Spaces** (message passing)

- **Naming strategy affects**
  - Programmer / Software
  - Performance
  - Design complexity
Issue: Operations

• **Uniprocessor RISC**
  – Ld/St and atomic operations on memory
  – Arithmetic on registers

• **Shared Memory Multiprocessor**
  – Ld/St and atomic operations on local/global memory
  – Arithmetic on registers

• **Message Passing Multiprocessor**
  – Send/receive on local memory
  – Arithmetic on registers
  – Broadcast

• **Data Parallel**
  – Ld/St
  – Global operations (add, max, etc.)
Issue: Ordering

• Uniprocessor
  – Programmer sees order as program order
  – Out-of-order execution (Tomasulo’s algorithm) actually changes order
  – Write buffers
  – Important to maintain true (RAW) dependencies

• Multiprocessor
  – What is order among several threads accessing shared data?
  – What affect does this have on performance?
  – What if implicit order is insufficient?
  – Memory consistency model specifies rules for ordering
Issue: Order/Synchronization

• Coordination mainly takes three forms:
  – Mutual exclusion (e.g., spin-locks)
  – Event notification
    » Point-to-point (e.g., producer-consumer)
    » Global (e.g., end of phase indication, all or subset of processes)
  – Global operations (e.g., sum)

• Issues:
  – Synchronization name space (entire address space or portion)
  – Granularity (per byte, per word, ... → overhead)
  – Low latency, low serialization (hot spots)
  – Variety of approaches
    » Test&set, compare&swap, LoadLocked-StoreConditional
    » Full/Empty bits and traps
    » Queue-based locks, fetch&op with combining
Performance Issue: Latency

• Must deal with latency when using fast processors

• Options:
  – Reduce frequency of long latency events
    » Algorithmic changes, computation and data distribution
  – Reduce latency
    » Cache shared data, network interface design, network design
  – Tolerate latency
    » Message passing overlaps computation with communication (program controlled)
    » Shared memory overlaps access completion and computation using consistency model and prefetching
Performance Issue: Bandwidth

• Private and global bandwidth requirements

• Private bandwidth requirements can be supported by:
  – Distributing main memory among PEs
  – Application changes, local caches, memory system design

• Global bandwidth requirements can be supported by:
  – Scalable interconnection network technology
  – Distributed main memory and caches
  – Efficient network interfaces
  – Avoiding contention (hot spots) through application changes
Cost of Communication

Cost = Frequency \times (Overhead + Latency + Xfer size/BW - Overlap)

- **Frequency** = number of communications per unit of work
  - Algorithm, placement, replication, bulk data transfer
- **Overhead** = processor cycles spent initiating or handling communication
  - Protection checks, status, buffer mgmt, copies, events
- **Latency** = time to move bits from source to dest
  - Communication assist, topology, routing, congestion
- **Transfer time** = time through bottleneck
  - Comm assist, links, congestions
- **Overlap** = portion overlapped with useful work
  - Comm assist, comm operations, processor design
Amdahl’s Law Updated (Marty & Hill)

- PRESENTATION
DISCUSSION
Summary

• Motivation & Applications

• Programming Models & A Generic Parallel Machine

• Issues in Programming Models