Multiprocessors and Multiprocessing

more is better?

Classifying Multiprocessors

- Interconnection Network
- Memory Topology
- Programming Model

Interconnection Network

- Bus
- Network
- pros/cons?
Memory Topology

- UMA (Uniform Memory Access)
- NUMA (Non-uniform Memory Access)
- pros/cons?

Programming Model

- Shared Memory -- every processor can name every address location
- Message Passing -- each processor can name only its local memory. Communication is through explicit messages (multicomputer).
- pros/cons?

find the max of 100,000 integers on 10 processors.

Parallel Programming

Processor A

i = 47

index = i++;

Processor B

index = i++;

- Shared-memory programming requires synchronization to provide mutual exclusion and prevent race conditions
  - locks (semaphores)
  - barriers

Multiprocessor Caches (Shared Memory)

- the problem -- cache coherency
- the solution?
What Does Coherence Mean?

- Informally:
  - Any read must return the most recent write
  - Too strict and very difficult to implement
- Better:
  - A processor sees its own writes to a location in the correct order.
  - Any write must eventually be seen by a read
  - All writes are seen in order ("serialization"). Writes to the same location are seen in the same order by all processors.
- Without these guarantees, synchronization doesn’t work.

Cache Coherency

- A good cache coherency protocol can avoid sending unnecessary (and expensive) invalidate or update messages.
- Allows each cache line to be in one of several states.
- MESI (Illinois)
  - modified
  - exclusive
  - shared
  - invalid

- How do you know when an external read/write occurs?
- Snooping protocols
- Directory protocols

- write-update
  - on each write, each cache holding that location updates its value
- write-invalidate \(\leq\) most common
  - on each write, each cache holding that location invalidates the cache line.

- both schemes MUCH easier on a bus-based multiprocessor
- potentially requires a LOT of messages, but...
**Potential Solutions**

- **Snooping Solution (Snoopy Bus):**
  - Send all requests for unknown data to all processors
  - Processors snoop to see if they have a copy and respond accordingly
  - Requires “broadcast”, since caching information is at processors
  - Works well with bus (natural broadcast medium)
  - Dominates for small scale machines

- **Directory-Based Schemes**
  - Keep track of what is being shared in one centralized place
  - Distributed memory => distributed directory (avoids bottlenecks)
  - Send point-to-point requests to processors
  - Scales better than Snoop
  - Actually existed BEFORE Snoop-based schemes

---

**Multiprocessors – Key Points**

- Network vs. Bus
- Message-passing vs. Shared Memory
- Shared Memory is more intuitive, but creates problems for both the programmer (memory consistency, requiring synchronization) and the architect (cache coherency).