Trend toward Multiprocessing

- Slowdown in uniprocessor performance improvements.
  - There are diminishing returns from attempting to exploit more ILP.
  - Clock rates can no longer be increased due to thermal constraints.
- Growing use of servers and data-intensive applications, which have more thread level parallelism.
- Having multiple processors on a single chip allows for much faster communication.
- An improved understanding of how to effectively design and use multiprocessors.

Some Definitions

- process - a running program and the state needed for its execution
- threads - processes that share their address space (code and data), but also sometimes threads indicate conventional processes
- grain size - amount of computation assigned to a thread
- multiprocessor - a computer consisting of tightly coupled processors typically controlled by a single operating system and share memory through a shared address space
- multicore - use of multiple processors on a single die
Classes of Multiprocessors

- A **symmetric (or centralized) shared memory** multiprocessor (SMP) means that they share a single centralized memory.
  - Used for multiprocessors with a small number of processors chips.
  - They are sometimes called **uniform memory access** (UMA) multiprocessors.
- A **distributed shared memory** (DSM) multiprocessor means that the memory is distributed among the processors.
  - Easier to scale to more processors and improves latency/bandwidth for accesses to local memory.
  - Communicating data between processors is more complex.
  - Requires more effort to appropriately allocate data in the distributed memories.
  - They are sometimes called **nonuniform memory access** (NUMA) multiprocessors.

Communication Models

- **shared memory multiprocessor**
  - Processors share the same address space.
  - Communication is implicit through loads and stores.
  - Generally easier to program.
  - Lower overhead for communication.
  - Can reduce frequency of remote communication through caching.
  - Used for both SMPs and DSMs.
- **message passing multiprocessor**
  - Each processor has its own address space.
  - Communication is explicit through passing messages between the processors, which may simplify analysis.
  - Hardware may more easily scale.
  - Used for clusters and warehouse-scale computers.

Challenges of Parallel Processing

- Limited parallelism available (Amdahl’s Law).
  - Coarse-grain parallelism is difficult to find automatically and may require new algorithms in the application.
  - Sometimes not a lot of parallelism available in applications.
- High cost of communication between processors.
  - Can use hardware techniques to cache shared data.
  - Can use software techniques to restructure data to provide more local accesses.

Cache Coherence Problem

- **Private** data are used by a single processor and **shared** data are used by multiple processors.
- Caches in a multiprocessor support:
  - **migration** - A shared data item can be moved to a local cache, which reduces the access time.
  - **replication** - A shared data item can have copies in different local caches, which reduces access time and contention for read access.
- The cache coherence problem is that there may be different values for the same memory location depending on which processor is accessing it. A multiprocessor has to ensure that each processor accesses the most recent value of a variable.
Coherent Memory System

- A memory system is coherent if:
  - If a write to X by processor P is followed by a read from X by processor P, then the read always returns the value written by P if there are no intervening writes.
  - If a write to X by one processor is followed by a read from X by another processor, then the read returns the value written by the first processor if there are no intervening writes and the write and read are sufficiently separated in time.
  - Writes to the same location are serialized, meaning two writes to X by any two processors are seen in the same order by all processors.

Cache Coherence Protocols

- Cache-coherence protocols are used to maintain the coherence of shared data in multiprocessors.
  - Snooping - Each cache has a copy of the sharing status of the blocks of shared data in that cache. All caches monitor (snoop) on the shared-cache or memory bus to determine if they have a copy of the requested block of data. Typically used with SMPs.
  - Directory Based - The sharing status is kept in a single location called the directory. It requires a little more overhead, but can scale to a larger number of processors. Typically used with DSMs.
  - All recent multiprocessors use a write-invalidate protocol, which means that all other copies of the block are invalidated when a write occurs granting exclusive access to the writing processor.

Implementing the Snooping Write-Invalidate Protocol

- To invalidate a data item in other caches, a processor acquires bus access and broadcasts the address to be invalidated.
- The serialization of access to the shared bus enforces the serialization of writes.
- An extra state bit is used to indicate if a cache block is shared (like a valid or dirty bit). Here shared means that the data block is resident in more than one local cache. A write to a block that has its shared bit set would cause an invalidate for that address to be sent on the bus so other caches would invalidate their copy.
- Most multiprocessors use an L2 write-back policy to reduce contention to the L3 cache. If a processor issues a read access to a shared L3 data block and another processor has a dirty copy of that item, then it provides the block to the read request and the access to the L3 cache is not required.

Extensions to the MSI Snooping Coherence Protocol

- The basic snooping coherence protocol is oftened referred to as MSI (Modified, Shared, Invalid).
- There are two common extensions.
  - MESI separates the Modified state from an Exclusive state, where exclusive indicates that the block has not been modified, but is resident only in that cache. A subsequent write to an exclusive block need not acquire the bus or generate an invalidate.
  - MOESI adds the state Owned to indicate that the block is owned by the cache and is out-of-date in memory. Exclusive means only in one cache and not dirty. Modified means only in one cache and dirty. Owned means shared in other caches and dirty in this cache. Write backs only occur when a modified or owned block is replaced.
Coherence Misses

- cache miss categorizations (first three apply to uniprocessors)
  - capacity
  - compulsory
  - conflict
  - coherence misses
    - true sharing misses
    - false sharing misses

Implementing the Directory Protocol

- Snooping protocols do not scale well since communication with all caches is required on every cache miss.
- The directory protocol keeps the state of every potentially shared memory block that may be cached. It uses bit vectors to track which processors have a copy of a block.
  - shared - one or more nodes have cached the block and none are dirty
  - uncached - no nodes have cached the block
  - modified - one node has cached the block and it is dirty
- The directory state is distributed where the memory resides.
- Nodes associated with the directory.
  - local node - where the memory request originates
  - home node - where the memory location/directory entry of address resides
  - remote node - node that has a copy of the cache block

Synchronization Instructions

- Synchronization mechanisms are typically implemented with software library routines that rely on special hardware instructions.
- Synchronization instructions
  - atomic exchange - Exchanges a register with a value in memory.
  - test and set - Tests a value and sets it if the condition passed.
  - load linked, store conditional - Used when difficult to implement an atomic memory operation. Load linked is executed first and address specified by the instruction is kept in a link register. Store conditional fails if address specified by the store does not match the address in the link register, which is changed when another store conditional executes.

Synchronization Mechanisms

- spin locks - locks that a processor continuously tries to acquire
  
  Simple Version
  
  Spin First on Local Copy

  MOV R2,#1
  lockit:
  LD R2,0(R1)
  EXCH R2,0(R1)
  BNEZ R2,lockit
  BNEZ R2,lockit

  DADDUI R2,R0,#1
  EXCH R2,0(R1)
  BNEZ R2,lockit

- barrier - Forces all processes to wait until they all reach the barrier.
- queuing locks - Processes are queued and obtain the lock in a FIFO order.
Memory Consistency

- Memory consistency indicates when another process must see a value that has been updated by another processor.
- Can both of the following conditions on two processors evaluate to true when A and B are shared variables?
  
P1: A = 0; P2: B = 0;
  ...               ...
  A = 1;           B = 1;
  L1: if (B == 0)  L2: if (A == 0)
  ...               ...

Models of Memory Consistency

- **sequential consistency**
  - All shared memory accesses must occur in the order in which they are specified by the program.
  - Can delay the completion of any shared memory access until all the invalidations caused by that access are completed.
  - Considered too restrictive since it can impact performance.
- **relaxed consistency models**
  - Allows reads and writes to complete out of order, but uses synchronization operations to enforce ordering when it is needed.
  - May result in an performance advantage.
  - Increases complexity of debugging multiprocessor applications.

Multiprocessors and Multilevel Inclusion

- The *multilevel inclusion* property (also called the subset property) requires that every level of the cache hierarchy is a subset of the next lower level (all data at one level is contained in the next lower level).
- This property can reduce contention between coherence and processor traffic since we can snoop only at the lower level cache.
- Problem occurs when the block size for an upper level cache is smaller than the block size of a lower level cache, which is quite common.
- Most common solution is to invalidate the corresponding blocks in the upper level that are not in the lower level cache after a lower level block is replaced or invalidated.

Fallacies and Pitfalls

- Pitfall: Measuring performance of multiprocessors by linear speedup versus execution time.
- Fallacy: Linear speedups are needed to make multiprocessors cost-effective.
- Pitfall: Not developing the software to take advantage of, or optimize, for a multiprocessor architecture.