

(Advanced) Computer Architechture

Prof. Dr. Hasan Hüseyin BALIK (12th Week)



Outline

- 5. Parallel organization
 - -Parallel Processing
 - -Multicore Computers





5.1 Parallel Processing



5.1 Outline

- Multiple Processors Organization
- Symmetric Multiprocessors
- Cache Coherence and the MESI Protocol
- Multithreading and Chip Multiprocessors
- Clusters
- Nonuniform Memory Access



Multiple Processor Organization

- Single instruction, single data (SISD) stream
 - Single processor executes a single instruction stream to operate on data stored in a single memory
 - Uniprocessors fall into this category
- Single instruction, multiple data (SIMD) stream
 - A single machine instruction controls the simultaneous execution of a number of processing elements on a lockstep basis
 - Vector and array processors fall into this category

- Multiple instruction, single data (MISD) stream
 - A sequence of data is transmitted to a set of processors, each of which executes a different instruction sequence
 - Not commercially implemented
- Multiple instruction, multiple data (MIMD) stream
 - A set of processors simultaneously execute different instruction sequences on different data sets
 - SMPs, clusters and NUMA systems fit this category



A Taxonomy of Parallel Processor Architectures





Alternative Computer Organizations



(c) MIMD (with shared memory)

CU = control unit
IS = instruction stream
PU = processing unit
DS = data stream
MU = memory unit
LM = local memory

SISD = single instruction, single data stream SIMD = single instruction, multiple data stream MIMD = multiple instruction, multiple data stream



(b) SIMD (with distributed memory)



(d) MIMD (with distributed memory)



Symmetric Multiprocessor (SMP)

A stand alone computer with the following characteristics:

Two or more similar processors of comparable capacity Processors share same memory and I/O facilities

- Processors are connected by a bus or other internal connection
- Memory access time is approximately the same for each processor

All processors share access to I/O devices

• Either through same channels or different channels giving paths to same devices All processors can perform the same functions (hence "symmetric") System controlled by integrated operating system

 Provides interaction between processors and their programs at job, task, file and data element levels



Multiprogramming and Multiprocessing



(a) Interleaving (multiprogramming, one processor)



(b) Interleaving and overlapping (multiprocessing; two processors)





Generic Block Diagram of a Tightly Coupled Multiprocessor





Symmetric Multiprocessor Organization





The bus organization has several attractive features:

- Simplicity
 - Simplest approach to multiprocessor organization
- Flexibility
 - Generally easy to expand the system by attaching more processors to the bus
- Reliability
 - The bus is essentially a passive medium and the failure of any attached device should not cause failure of the whole system



Disadvantages of the bus organization:

- Main drawback is performance
 - All memory references pass through the common bus
 - Performance is limited by bus cycle time
- Each processor should have cache memory
 - Reduces the number of bus accesses
- Leads to problems with *cache coherence*
 - If a word is altered in one cache it could conceivably invalidate a word in another cache
 - To prevent this the other processors must be alerted that an update has taken place
 - Typically addressed in hardware rather than the operating system



Multiprocessor Operating System Design Considerations

Simultaneous concurrent processes

- OS routines need to be reentrant to allow several processors to execute the same IS code simultaneously
- OS tables and management structures must be managed properly to avoid deadlock or invalid operations

Scheduling

- Any processor may perform scheduling so conflicts must be avoided
- Scheduler must assign ready processes to available processors

Synchronization

- With multiple active processes having potential access to shared address spaces or I/O resources, care must be taken to provide effective synchronization
- Synchronization is a facility that enforces mutual exclusion and event ordering

Memory management

- In addition to dealing with all of the issues found on uniprocessor machines, the OS needs to exploit the available hardware parallelism to achieve the best performance
- Paging mechanisms on different processors must be coordinated to enforce consistency when several processors share a page or segment and to decide on page replacement

Reliability and fault tolerance

- OS should provide graceful degradation in the face of processor failure
- Scheduler and other portions of the operating system must recognize the loss of a processor and restructure accordingly



Cache Coherence (1 of 2)

Software Solutions

- Attempt to avoid the need for additional hardware circuitry and logic by relying on the compiler and operating system to deal with the problem
- Attractive because the overhead of detecting potential problems is transferred from run time to compile time, and the design complexity is transferred from hardware to software
 - However, compile-time software approaches generally must make conservative decisions, leading to inefficient cache utilization



Cache Coherence (2 of 2)

Hardware-Based Solutions

- Generally referred to as cache coherence protocols
- These solutions provide dynamic recognition at run time of potential inconsistency conditions
- Because the problem is only dealt with when it actually arises there is more effective use of caches, leading to improved performance over a software approach
- Approaches are transparent to the programmer and the compiler, reducing the software development burden
- Can be divided into two categories:
 - Directory protocols
 - Snoopy protocols



Directory Protocols

Collect and maintain information about copies of data in cache

Directory stored in main memory

Effective in large scale systems with complex interconnection schemes

Creates central bottleneck

Requests are checked against directory Appropriate transfers are performed



Snoopy Protocols

- Distribute the responsibility for maintaining cache coherence among all of the cache controllers in a multiprocessor
 - A cache must recognize when a line that it holds is shared with other caches
 - When updates are performed on a shared cache line, it must be announced to other caches by a broadcast mechanism
 - Each cache controller is able to "snoop" on the network to observe these broadcast notifications and react accordingly
- Suited to bus-based multiprocessor because the shared bus provides a simple means for broadcasting and snooping
 - Care must be taken that the increased bus traffic required for broadcasting and snooping does not cancel out the gains from the use of local caches
- Two basic approaches have been explored:
 - Write invalidate
 - Write update (or write broadcast)



Write Invalidate

- Multiple readers, but only one writer at a time
- When a write is required, all other caches of the line are invalidated
- Writing processor then has exclusive (cheap) access until line is required by another processor
- Most widely used in commercial multiprocessor systems such as the x86 architecture
- State of every line is marked as modified, exclusive, shared or invalid
 - For this reason the write-invalidate protocol is called MESI



Write Update

Can be multiple readers and writers

When a processor wishes to update a shared line the word to be updated is distributed to all others and caches containing that line can update it

Some systems use an adaptive mixture of both write-invalidate and write-update mechanisms



MESI Protocol

To provide cache consistency on an SMP the data cache supports a protocol known as MESI:

- Modified
 - The line in the cache has been modified and is available only in this cache
- Exclusive
 - The line in the cache is the same as that in main memory and is not present in any other cache
- Shared
 - The line in the cache is the same as that in main memory and may be present in another cache
- Invalid
 - The line in the cache does not contain valid data



MESI Cache Line States

	M Modified	E Exclusive	S Shared	l Invalid
This cache line valid?	Yes	Yes	Yes	No
The memory copy is	out of date	valid	valid	-
Copies exist in other caches?	No	No	Maybe	Maybe
A write to this line …	does not go to bus	does not go to bus	goes to bus and updates cache	goes directly to bus



MESI State Transition Diagram





Relationship Between Cache Lines in Cooperating Caches





Read Miss

- When a read miss occurs in the local cache, the processor initiates a memory read to read the line of main memory containing the missing address
- The processor inserts a signal on the bus that alerts all other processor/cache units to snoop the transaction
- There are a number of possible outcomes resulting from this process



Read Hit





Write Miss

- When a write miss occurs in the local cache, the processor initiates a memory read to read the line of main memory containing the missing address
- For this purpose, the processor issues a signal on the bus that means *read-with-intent-to-modify* (RWITM)
- When the line is loaded, it is immediately marked modified
- With respect to other caches, two possible scenarios precede the loading of the line of data
 - Some other cache may have a modified copy of this line
 - No other cache has a modified copy of the requested line



Write Hit

- When a write hit occurs on a line currently in the local cache, the effect depends on the current state of that line in the local cache:
 - Shared
 - Before performing the update, the processor must gain exclusive ownership of the line
 - The processor signals its intent on the bus
 - Each processor that has a shared copy of the line in its cache transitions the sector from shared to invalid
 - The initiating processor then performs the update and transitions its copy of the line from shared to modified
 - Exclusive
 - The processor already has exclusive control of this line, and so it simply performs the update and transitions its copy of the line from exclusive to modified
 - Modified
 - The processor already has exclusive control of this line and has the line marked as modified, and so it simply performs the update



Initiator Reads from Writeback Cache





Initiator Writes to Writeback Cache





Multithreading and Chip Multiprocessors

- Processor performance can be measured by the rate at which it executes instructions
- MIPS rate = f * IPC
 - f = processor clock frequency, in MHz
 - IPC = average instructions per cycle
- Increase performance by increasing clock frequency and increasing instructions that complete during cycle
- Multithreading
 - Allows for a high degree of instruction-level parallelism without increasing circuit complexity or power consumption
 - Instruction stream is divided into several smaller streams, known as threads, that can be executed in parallel



Definitions of Threads

and Processes

Thread switch

process

switch

Thread:

process

turn to another thread

Thread in multithreaded processors may or may not be the same as the concept of software threads in a multiprogrammed operating system



Process switch

 Operation that switches the processor from one process to another by saving all the process control data, registers, and other information for the first and replacing them with the process information for the second

Thread is concerned with scheduling and execution, whereas a process is concerned with both scheduling/execution and resource and resource ownership

Process:

- An instance of program running on computer
- Two key characteristics:
- Resource ownership
- Scheduling/execution



Implicit and Explicit Multithreading

- All commercial processors and most experimental ones use explicit multithreading
 - Concurrently execute instructions from different explicit threads
 - Interleave instructions from different threads on shared pipelines or parallel execution on parallel pipelines
- Implicit multithreading is concurrent execution of multiple threads extracted from single sequential program
 - Implicit threads defined statically by compiler or dynamically by hardware



Approaches to Explicit Multithreading

- Interleaved
 - Fine-grained
 - Processor deals with two or more thread contexts at a time
 - Switching thread at each clock cycle
 - If thread is blocked it is skipped
- Simultaneous (SMT)
 - Instructions are simultaneously issued from multiple threads to execution units of superscalar processor

- Blocked
 - Coarse-grained
 - Thread executed until event causes delay
 - Effective on in-order processor
 - Avoids pipeline stall
- Chip multiprocessing
 - Processor is replicated on a single chip
 - Each processor handles separate threads
 - Advantage is that the available logic area on a chip is used effectively



Approaches to Executing Multiple Threads



(SMT)

VLIW

ISTANBUL AU

Clusters

- Alternative to SMP as an approach to providing high performance and high availability
- Particularly attractive for server applications
- Defined as:
 - A group of interconnected whole computers working together as a unified computing resource that can create the illusion of being one machine
 - (The term *whole computer* means a system that can run on its own, apart from the cluster)
- Each computer in a cluster is called a node
- Benefits:
 - Absolute scalability
 - Incremental scalability
 - High availability
 - Superior price/performance



Cluster Configurations



(a) Standby server with no shared disk





(b) Shared disk

Clustering Methods: Benefits and Limitations

Clustering Method	Description	Benefits	Limitations
Passive Standby	A secondary server takes over in case of primary server failure.	Easy to implement.	High cost because the secondary server is unavailable for other processing tasks.
Active Secondary:	The secondary server is also used for processing tasks.	Reduced cost because secondary servers can be used for processing.	Increased complexity.
Separate Servers	Separate servers have their own disks. Data is continuously copied from primary to secondary server.	High availability.	High network and server overhead due to copying operations.
Servers Connected to Disks	Servers are cabled to the same disks, but each server owns its disks. If one server fails, its disks are taken over by the other server.	Reduced network and server overhead due to elimination of copying operations.	Usually requires disk mirroring or RAID technology to compensate for risk of disk failure.
Servers Share Disks	Multiple servers simul- taneously share access to disks.	Low network and server overhead. Reduced risk of downtime caused by disk failure.	Requires lock manager software. Usually used with disk mirroring or RAID technology.



Nonuniform Memory Access (NUMA)

- Alternative to SMP and clustering
- Uniform memory access (UMA)
 - All processors have access to all parts of main memory using loads and stores
 - Access time to all regions of memory is the same
 - Access time to memory for different processors is the same
- Nonuniform memory access (NUMA)
 - All processors have access to all parts of main memory using loads and stores
 - Access time of processor differs depending on which region of main memory is being accessed
 - Different processors access different regions of memory at different speeds
- Cache-coherent NUMA (CC-NUMA)
 - A NUMA system in which cache coherence is maintained among the caches of the various processors



Motivation

SMP has practical limit to number of processors that can be used

• Bus traffic limits to between 16 and 64 processors

In clusters each node has its own private main memory

- Applications do not see a large global memory
- Coherency is maintained by software rather than hardware

NUMA retains SMP flavor while giving large scale multiprocessing Objective with NUMA is to maintain a transparent system wide memory while permitting multiple multiprocessor nodes, each with its own bus or internal interconnect system



CC-NUMA Organization





NUMA Pros and Cons

- Main advantage of a CC-NUMA system is that it can deliver effective performance at higher levels of parallelism than SMP without requiring major software changes
- Bus traffic on any individual node is limited to a demand that the bus can handle
- If many of the memory accesses are to remote nodes, performance begins to break down

- Does not transparently look like an SMP
- Software changes will be required to move an operating system and applications from an SMP to a CC-NUMA system
- Concern with availability

