MultiProcessor and Thread-Level Parallelism

Chapter 6 - Computer Architecture : *A Quantitative Approach* - Hennessy & Patterson

Distributed Shared-Memory

- Multiprocessors with distributed shared-memory may include or exclude cache coherence.
- The hardware is simple if the cache coherence is excluded.
- Snooping protocol is not scalable.
- One alternative to the snooping protocol is a directory protocol.

No Cache Coherence

- Only **private data** are cacheable, but shared data are not.
- **Software cache coherence** is possible to a limited extent e.g. structured loop-level parallelism.
- Often the software-based coherence **algorithm is conservative**. It treates every block that is **potentially sharable** as **shared**, with exccess overhead.

No Cache Coherence

- **Without cache coherence a multiprocessor loses the benefit of spacial locality of a cached block as every word is to be fetched from the remote memory.**

Snooping Protocol

- Snooping protocol requirs communication with every cache on every cache miss.
- The advantage is that there is no centralized data structure to track the state of the cache.
- But the bandwidth demand on memory bus increases with the increase in the number of processors.

Scalable Shared-Mmeory Architecture

- **Includes cache coherence.**
- **Alternative to snooping cache coherence protocol is a directory protocol.**
- **An hybrid approach is also possible.**

Directory Protocol

- **A directory keeps information of every memory block that is cached.**
- **Directory information includes - caches that have the copy of the block, dirty block etc.**
- **In existing systems every memory block has a directory entry.**

Directory Protocol

- The amount of directory information is proportional to the product of the number of memory blocks and the number of processors.
- This overhead is tolerable (?) for about 200 processors.
- Proposed methods to scale the directory structure keeps entries only for the cached blocks or keep lesser number of bits per entry.

Directory Protocol

- **Directories are distributed along with the memories, so that directory access would not be a bottleneck.**
- **The sharing status of a block is available at a single known location and a broadcast can be avoided in the protocol.**

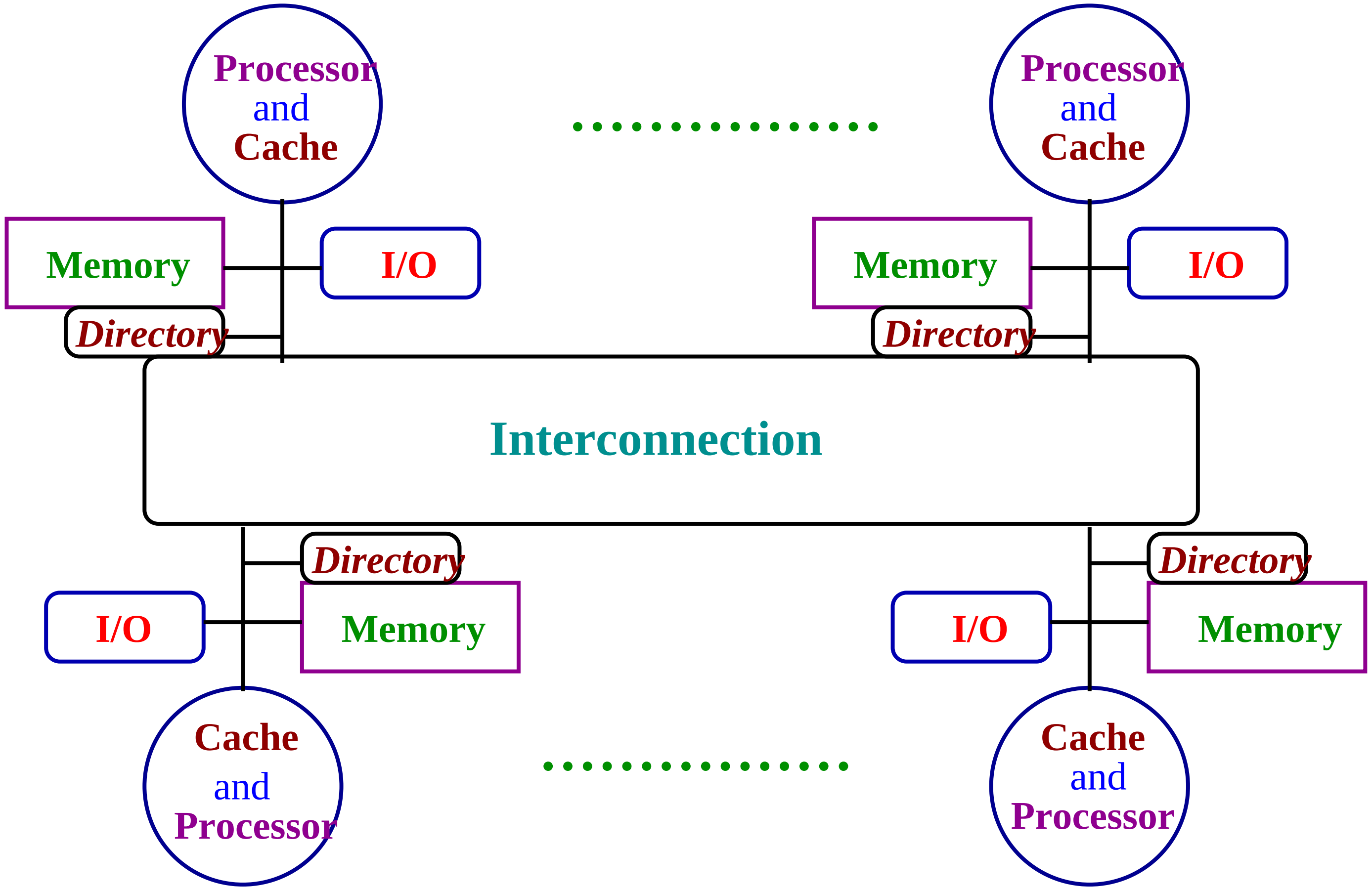

Figure 1: **Cache Coherence Directory**

Essential Operations

- The protocol implements **two primary operations**.
- Handling **read miss**.
- Handling **write in a clean cache block**.
- **Write miss in a shared block** is a combination of these two.
- There are **three cache states** in the **simplified version** of the protocol.

States of a Data Block

- **Shared: One or more processors have cached the block, the memory is up to date.**
- **Uncached: No cache has a copy of the block.**
- **Exclusive: Only one processor has a copy of the block, which has writen in it. The processor is called the owner of the block.**

States of a Memory Block

- **It is also necessary to track the processors that share the block. Caches in all these processors are to be invalidated on write.**

Bit Vectors

- Keep a **bit vector** for **each memory block**, one bit for a processor. A bit indicates whether the block is **shared** by the processor.
- The bit vector can also be used to **track the owner** of the block (**exclusive state** - one bit is set).
- The state of **each cache block** in the **individual cache** is also tracked for efficiency reason.

State Transitions

- The states and the state transitions of each cache are identical to the snooping cache.
- The actions on transitions are different.

Simplifying Assumption

- Any attempt to **write data in a cache** that is **not exclusive** in the writer's cache will generate a **write miss.**
- To **minimize** the **type of messages** and the **complexity of the protocol** we consider a **simple model of consistency**.
  - The messages will be **received** and **acted upon** in the **order they were sent**.

Interconnect

- **A single point arbitration is not possible as the interconnect is not a bus.**
- **Messages have explicit responses on the ineterconnect.**

Note $\cdots$

- **Local node - the node where the request originates.**
- **Home node - the node where the memory location and the directory entry of the address resides.**
- **Remote node - that has a copy of a data block in a cache (shared or exclusive). It may be same as local or home node.**

Note · · ·

- The **physical address space** is **statically distributed**.
- The **higher order bits** of the **physical address** is used to **identify a node** and the **lower order bits** are used as **offset** within the memory space in the node.

Message Types

- **Read miss at address A of processor P.**
- **Source - local cache, Destination - home directory.**
- **Message: (P, A).**
- **Action - request data and make P as a read sharer.**

Message Types

- **Write miss at address A of processor P.**
- **Source - local cache, Destination - home directory.**
- **Message: (P, A).**
- **Action - request data and make P as the exclusive owner.**

Message Types

- **Invalidate -**
- **Source - home directory, Destination - remote cache.**
- **Message:** A.
- **Action - invalidate a shared copy of data at address** A **in every remote cache.**

Message Types

- **Fetch data at address A.**
- **Source - home directory, Destination - remote cache.**
- **Message: A.**
- **Action - Fetch the block at address A and send it to to its home directory; change the state of A in remote cache to shared.**

Message Types

- **Fetch/Invalidate data at address A.**
- **Source - home directory, Destination - remote cache.**
- **Message: A.**
- **Action - Fetch the block at address A and send it to to its home directory; invalidate the block in remote cache.**

Message Types

- **Data value Reply** -
- **Source** - home directory, **Destination** - local cache.
- **Message:** D.
- **Action** - return a data value from the home directory.

Message Types

- **Data write back.**
- **Source - remote cache, Destination - home directory.**
- **Message:** (A, D).
- **Action - write back a data value for the address** A.

Simplified State Transition Diagrams

- **Following state transition diagrams are simplified.**
- **The first diagram is for a cache block and the second one is for a directory.**

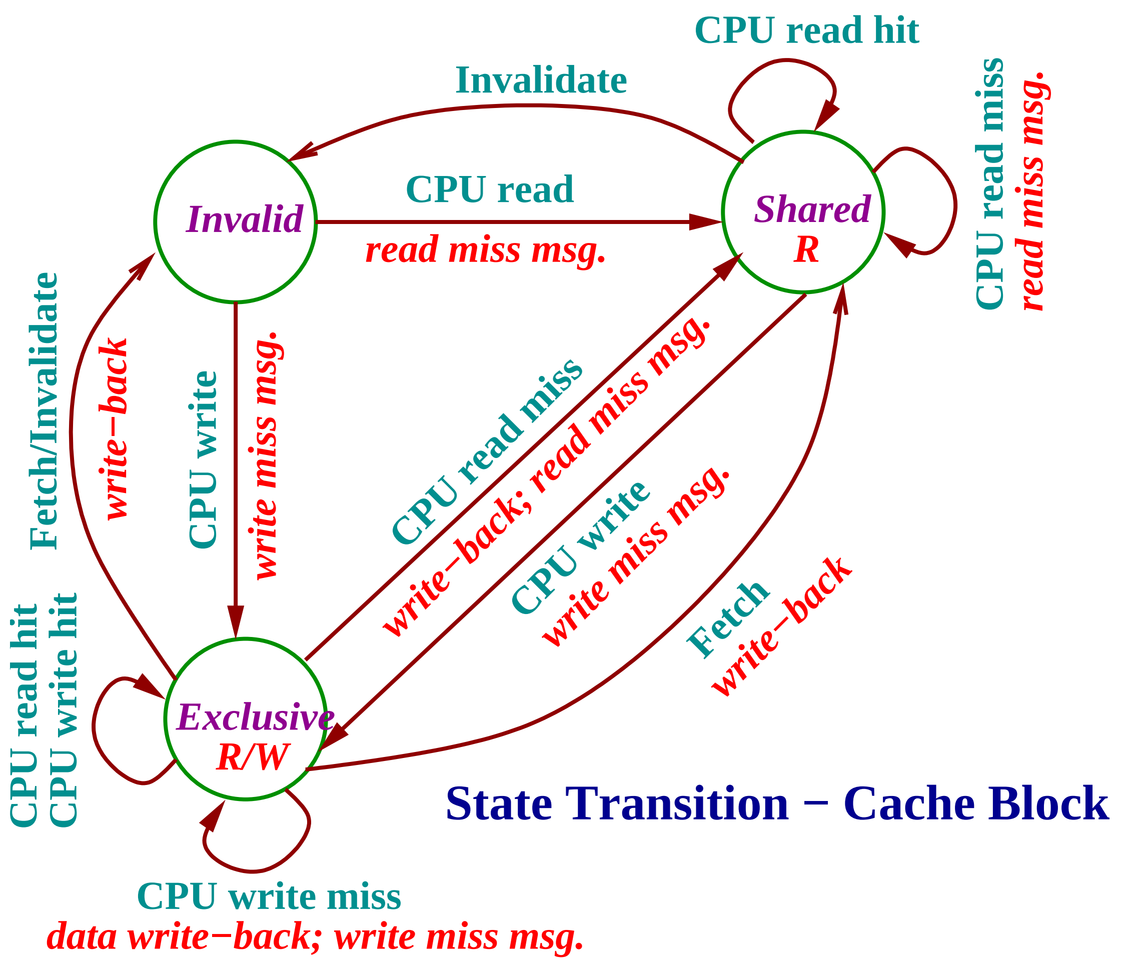

Figure 2: **Directory Protocol - Cache**

FSM in Directory

- The other half of the protocol is with the **directory**.
- A directory takes **two different types actions** on **receiving a message** - **state of the directory changes** and a **imessage may be sent in reply**.
- The directory state is the **combined state of all cache copies of a memory blocki** - **not cached**, **exclusive**, **shared in different cache blocks**.

FSM in Directory

- **A directory also have information about the processors that share a block - sharers.**
- **A directory receives three types of messages - read miss, write miss and data write back.**

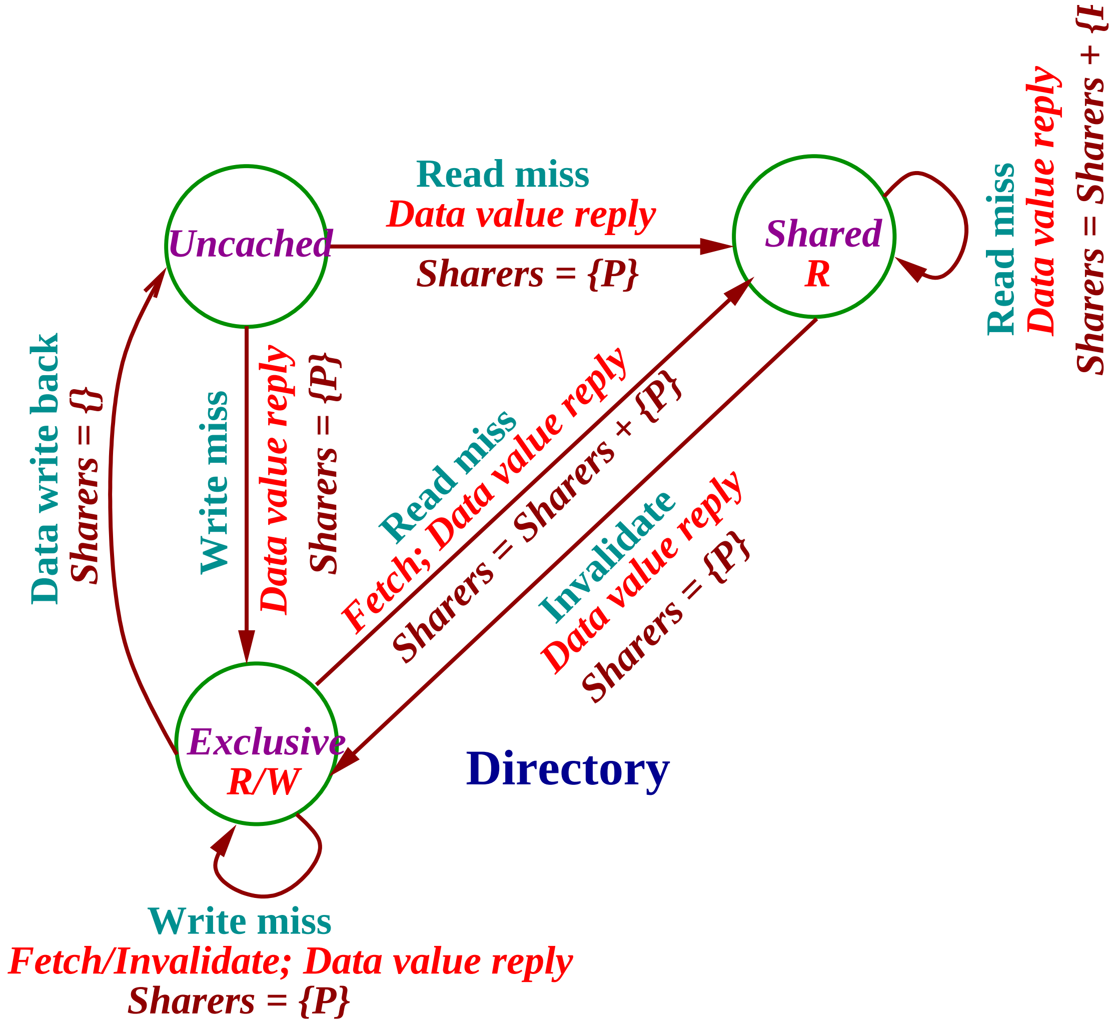

Figure 3: **Directory Protocol - Directory**