

# Caches for Multiprocessor Architectures

Most slides adopted from David Patterson

David Andrews  
Computer Engineering Group  
University of Paderborn

`dandrews@ittc.ku.edu`



**UNIVERSITÄT PADERBORN**  
*Die Universität der Informationsgesellschaft*

## First, Where Do Multiprocessors Fit ?

---

- SISD (Single Instruction Single Data)
  - ◆ Uniprocessors
- MISD (Multiple Instruction Single Data)
  - ◆ ???; multiple processors on a single data stream
- SIMD (Single Instruction Multiple Data)
  - ◆ Historical Examples: Illiac-IV, CM-2
    - Simple programming model
    - Low overhead
    - Flexibility
    - All custom integrated circuits
  - ◆ Recent Example: IBM Blue Gene, Cell Processor
- MIMD (Multiple Instruction Multiple Data)
  - ◆ Historical Examples: Cray T3D, SGI Origin
    - Flexible
    - *Use off-the-shelf micros*
  - ◆ Recent Example: Virtex 2 P30 with 2 PPC 405's
- Within General Computing MIMD current winner:
  - ◆ Clusters, x Core PC's

## Major MIMD Styles

---

- Generalized Taxonomies Along Memory Access Lines
- Centralized Shared Memory
  - Uniform Memory Access or Shared Memory Processor
  - Global Address Space for All. Constant Access Time from Anywhere
- Decentralized memory (memory module with CPU)
  - get more memory bandwidth, lower memory latency
  - Drawback: Longer communication latency
  - Drawback: Software model more complex
- Multiprocessor Systems on Chip Taxonomies Include
  - Variability of “Cores”
    - Homogeneous: All Processors Identical
    - Heterogeneous: Different Cores {CPU + DSP}

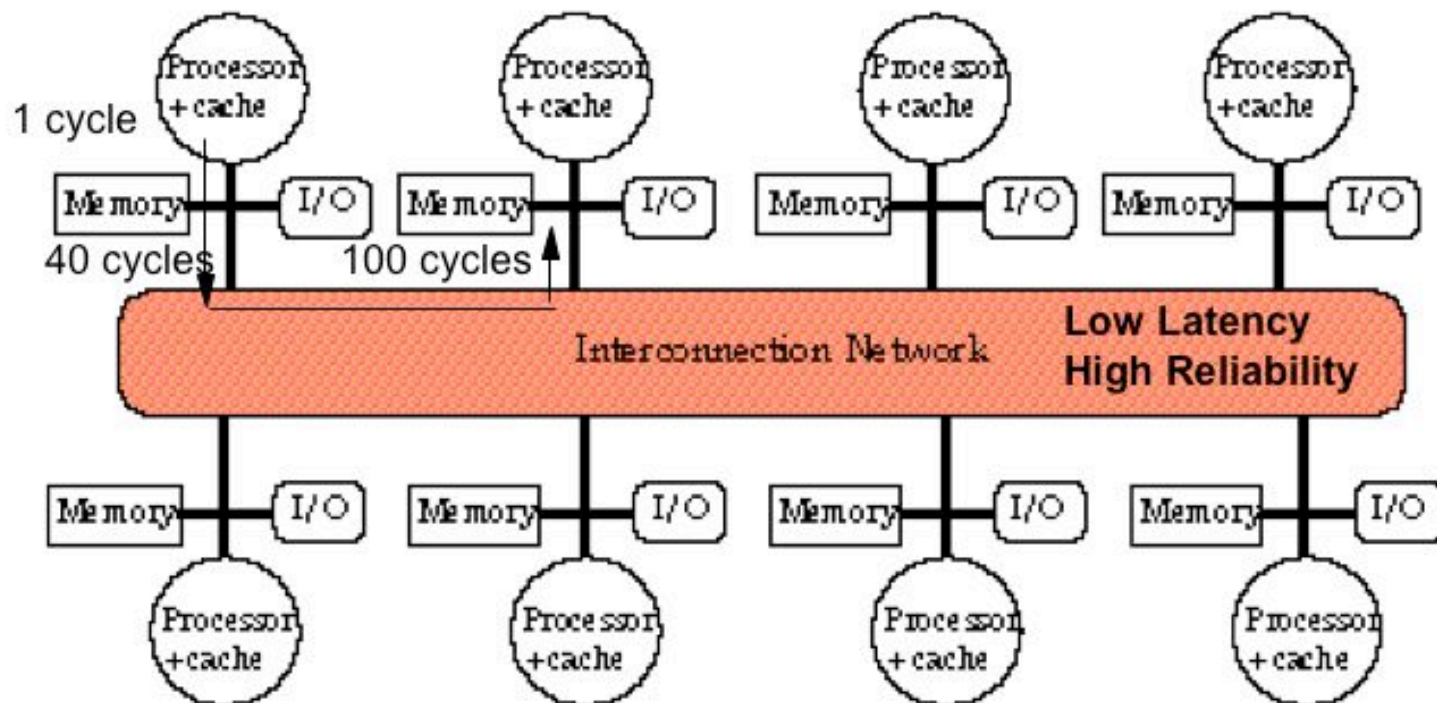
# SMP—Shared Memory Organization

---

- Caches serve to:
  - ◆ Increase bandwidth versus bus/memory
  - ◆ Reduce latency of access
  - ◆ Valuable for both private data and shared data
- I/O & Memory Global Access

## Large-Scale MP Designs

- **Memory:** distributed with nonuniform access time (“numa”) and scalable interconnect (distributed memory)
- **Examples:** T3E: (see Ch. 1, Figs 1-21, page 45 of [CSG96])



# Classic Cache Coherency For Multiprocessors

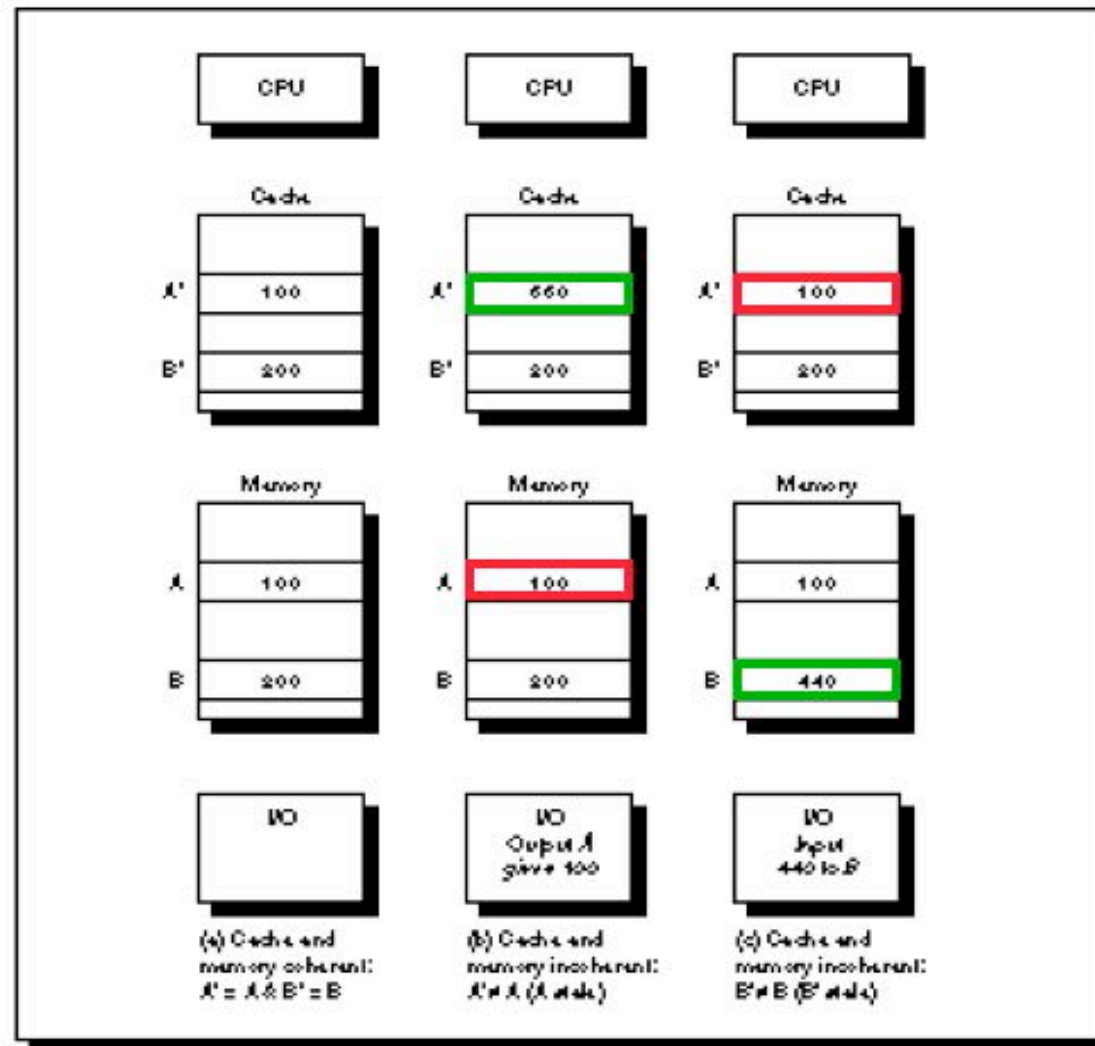
**-SMP Architectures := Snoopy Cache**

**-NUMA Architectures := Global Directory Cache**



**UNIVERSITÄT PADERBORN**  
*Die Universität der Informationsgesellschaft*

# The Problem of Cache Coherency



pr.'98 ©UCB 25

---

## Potential HW Coherency Solutions

- Snooping Solution (Small Numbers of PE's):
  - ◆ Send all requests for 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 (most of the market)
- Directory-Based Schemes (Larger Numbers of PE's)
  - ◆ Keep track of what is being shared in 1 centralized place (logically)
  - ◆ Distributed memory => distributed directory for scalability (avoids bottlenecks)
  - ◆ Send point-to-point requests to processors via network
  - ◆ Scales better than Snooping
  - ◆ Actually existed BEFORE Snooping-based schemes

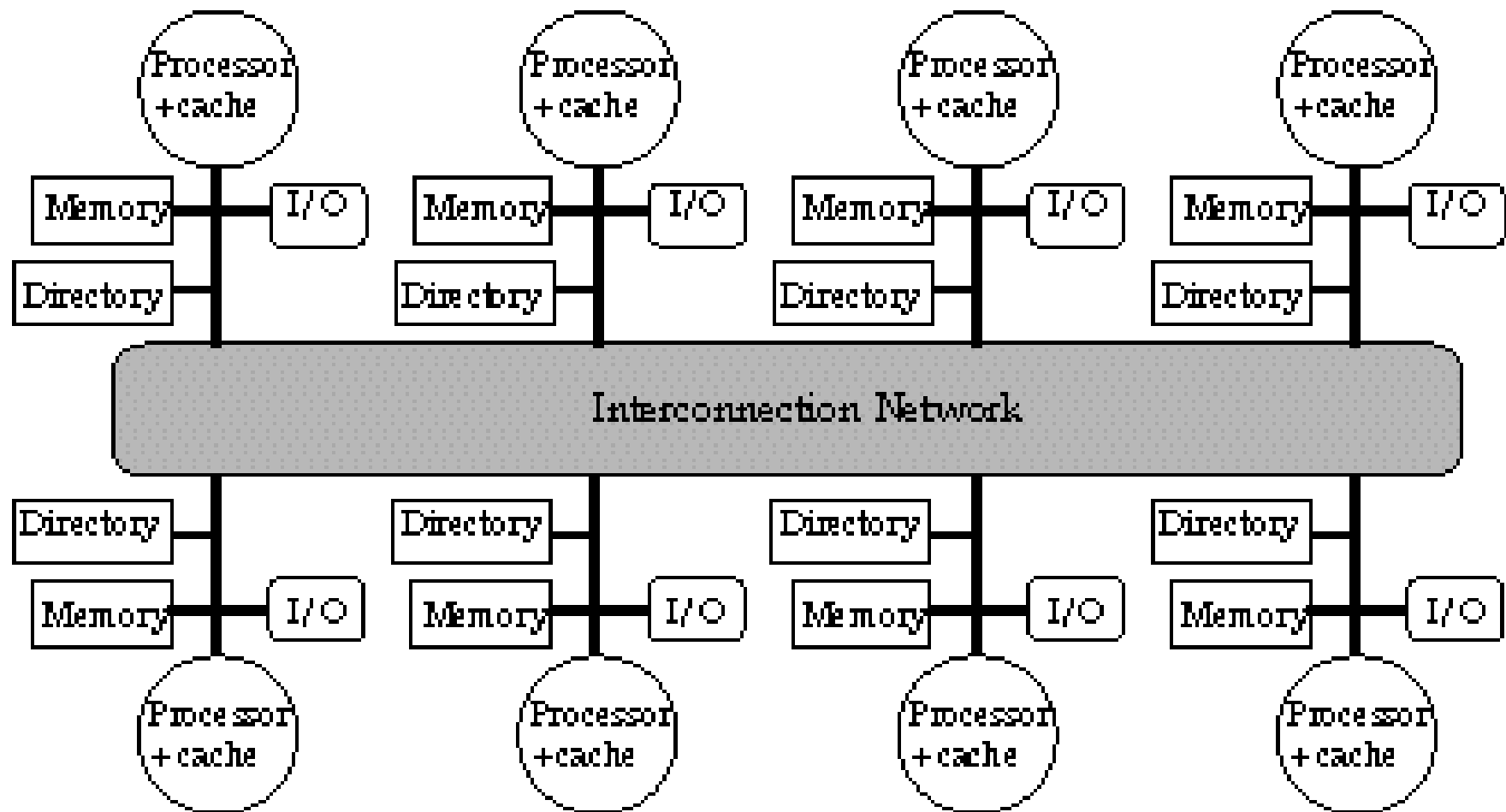


## Larger MPs

---

- Separate Memory per Processor
- Local or Remote access via memory controller
- 1 Cache Coherency solution: non-cached pages
- Alternative: directory per cache that tracks state of every block in every cache
  - ◆ Which caches have a copies of block, dirty vs. clean, ...
- Info per memory block vs. per cache block?
  - ◆ PLUS: In memory => simpler protocol (centralized/one location)
  - ◆ MINUS: In memory => directory is  $f(\text{memory size})$  vs.  $f(\text{cache size})$
- Prevent directory as bottleneck?  
distribute directory entries with memory, each keeping track of which Procs have copies of their blocks

## Distributed Directory MPs



## Network Examples

---

- Bi-directional Ring – EX: HP V Class
- 2-D Mesh and Hypercube – SGI Origin and Cray T3E
- Crossbar and Omega Network – SMPs, IBM SP3, and IP Routers
- Clusters using ethernet, Gigabit ethernet, Myrinet, etc.

Properties of various networks will be discussed later

## CC-NUMA Multiprocessor: Directory Protocol

---

- What is Cache Coherent Non-Uniform Memory Access (CC-NUMA)?
- Similar to Snoopy Protocol: Three states
  - ◆ Shared:  $\geq 1$  processors have data, memory up-to-date
  - ◆ Uncached (no processor has it; not valid in any cache)
  - ◆ Exclusive: 1 processor (owner) has data; memory out-of-date
- In addition to cache state, must track which processors have data when in the shared state (usually bit vector, 1 if processor has copy)
- Directory Size: Big  $\Rightarrow$  Limited Directory Schemes (Not to be discussed)

# Directory Protocol

---

- No bus and don't want to broadcast:
  - ◆ interconnect no longer single arbitration point
  - ◆ all messages have explicit responses
- Terms: typically 3 processors involved
  - ◆ **Local node** where a request originates
  - ◆ **Home node** where the memory location of an address resides
  - ◆ **Remote node** has a copy of a cache block, whether exclusive or shared
- Example messages on next slide:  
P = processor number, A = address

## Example Directory Protocol

---

- Message sent to directory causes two actions:
  - ◆ Update the directory
  - ◆ More messages to satisfy request
- Block is in **Uncached** state: the copy in memory is the current value; only possible requests for that block are:
  - ◆ **Read miss**: requesting processor sent data from memory & requestor made only sharing node; state of block made Shared.
  - ◆ **Write miss**: requesting processor is sent the value & becomes the Sharing node. The block is made Exclusive to indicate that the only valid copy is cached. Sharers indicates the identity of the owner.
- Block is **Shared** => the memory value is up-to-date:
  - ◆ **Read miss**: requesting processor is sent back the data from memory & requesting processor is added to the sharing set.
  - ◆ **Write miss**: requesting processor is sent the value. All processors in the set Sharers are sent invalidate messages, & Sharers is set to identity of requesting processor. The state of the block is made Exclusive.

## Example Directory Protocol

---

- Block is **Exclusive**: current value of the block is held in the cache of the processor identified by the set Sharers (the owner) => three possible directory requests:
  - ◆ **Read miss**: owner processor sent data fetch message, causing state of block in owner's cache to transition to Shared and causes owner to send data to directory, where it is written to memory & sent back to requesting processor. Identity of requesting processor is added to set Sharers, which still contains the identity of the processor that was the owner (since it still has a readable copy). State is shared.
  - ◆ **Data write-back**: owner processor is replacing the block and hence must write it back, making memory copy up-to-date (the home directory essentially becomes the owner), the block is now Uncached, and the Sharer set is empty.
  - ◆ **Write miss**: block has a new owner. A message is sent to old owner causing the cache to send the value of the block to the directory from which it is sent to the requesting processor, which becomes the new owner. Sharers is set to identity of new owner, and state of block is made Exclusive.

## Rough Timing Analysis

The previous Example looked at the bandwidth demands. The other key issue for a parallel program is remote memory access time, or latency. To get insight into this, we use a simple example of a directory-based machine. Figure 8.30 shows the parameters we assume for our simple machine. It assumes that the time to first word for a local memory access is 25 cycles and that the path to local memory is 8 bytes wide, while the network interconnect is 2 bytes wide. This model ignores the effects of contention, which are probably not too serious in the parallel benchmarks we examine, with the possible exception of FFT, which uses all-to-all communication. Contention could have a serious performance impact in other work loads.

Characteristic	Number of processor clock cycles
Cache hit	1
Cache miss to local memory	$25 + \frac{\text{block size in bytes}}{8}$
Cache miss to remote home directory	$75 + \frac{\text{block size in bytes}}{2}$
Cache miss to remotely cached data (3-hop miss)	$100 + \frac{\text{block size in bytes}}{2}$

**FIGURE 8.30** Characteristics of the example directory-based machine. Misses can be serviced locally (including from the local directory), at a remote home node, or using the services of both the home node and another remote node that is caching an exclusive copy. This last case is called a 3-hop miss and has a higher cost because it requires interrogating both the home directory and a remote cache. Note that this simple model does not account for invalidation time. These network latencies are typical of what can be achieved in 1995–96 in an MPP-style network interfaced in hardware to each node and assuming moderately fast processors (150–200 MHz).