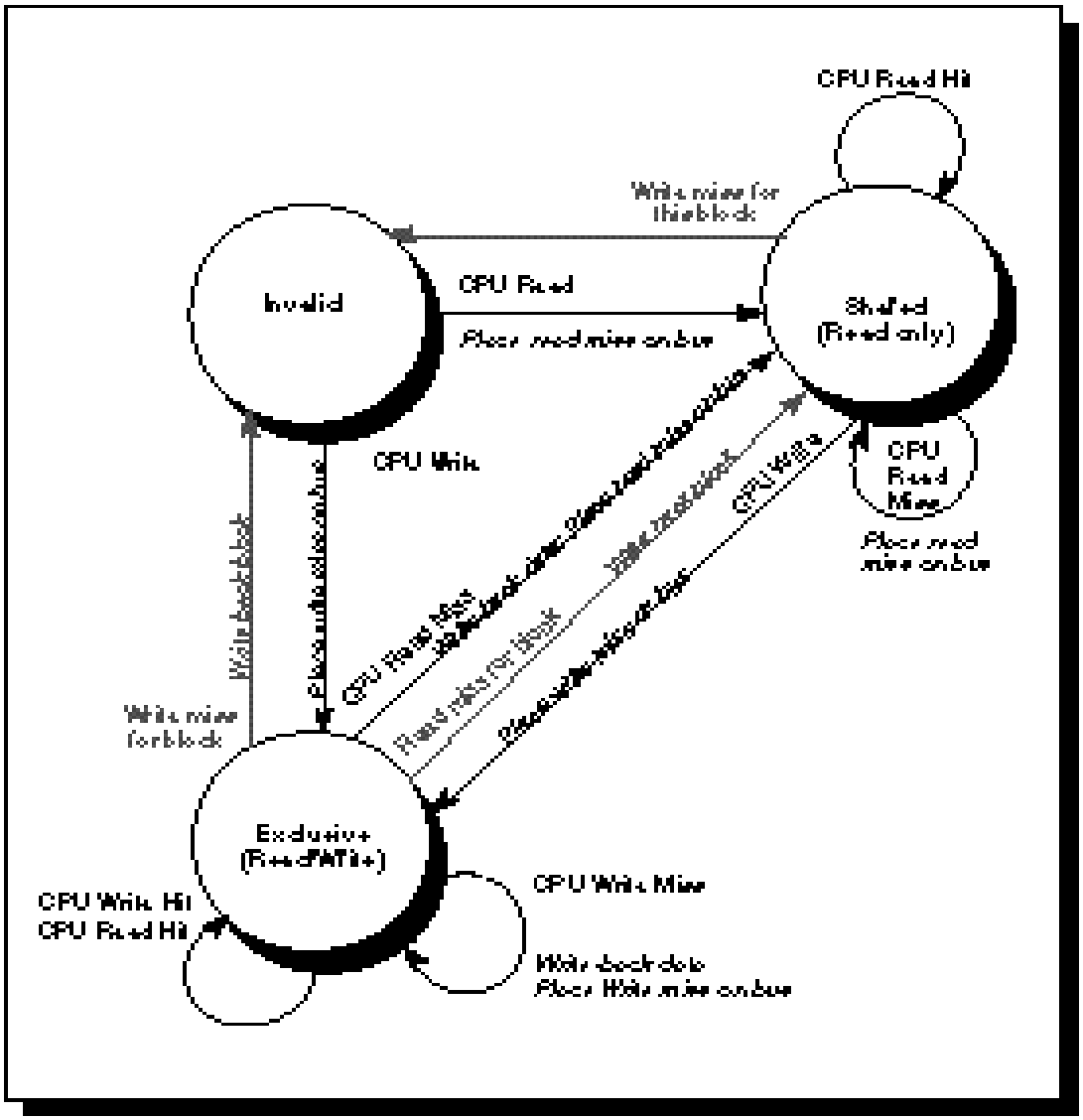
Lecture 32: Multiprocessors—Performance of Snoopy Caches vs. Directories

Professor Randy H. Katz

Computer Science 252

Spring 1996

Snoop Cache: State Machine



Extensions:

- Fourth State
- Clean-> dirty, need invalidate only (upgrade request) Berkeley Protocol
- Clean exclusive state (no miss for private data on write) Illinois Protocol

Snoop Cache Variations

Berkeley Protocol	Basic Protocol	Illinois Protocol
Owned Exclusive	**Exclusive**	**Private Dirty**
Owned Shared	**Shared**	**Private Clean**
Shared	**Invalid**	**Shared**
Invalid		**Invalid**

Owner can update via bus invalidate operation
Owner must write back when replaced in cache

If read sourced from memory, then Private Clean
if read sourced from other cache, then Shared
Can write in cache if held private clean or dirty

Review: Larger MPs

- Separate Memory per Processor
- Local or Remote access via memory controller
- Cache Coherency solution: non cached pages
- Alternative: directory/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?
  - In memory => simpler protocol (centralized/one location)
  - In memory => directory is *f*(memory size) vs. *f*(cache size)
- Prevent directory as bottleneck: distribute directory entries with memory, each keeping track of which Procs have copies of their blocks

Review: Directory Protocol

- **Similar to Snoopy Protocol: 3 states**
  - **Shared: ≥ 1 processors have data, memory up to date**
  - **Uncached**
  - **Exclusive: 1 processor(owner) has data; memory out of date**
- **In addition to cache state, must track which processors have data when shared state**
- **Terms:**
  - **Local node is the node where a request originates**
  - **Home node is the node where the memory location of an address resides**
  - **Remote node is the node that has a copy of a cache block, whether exclusive or shared.**

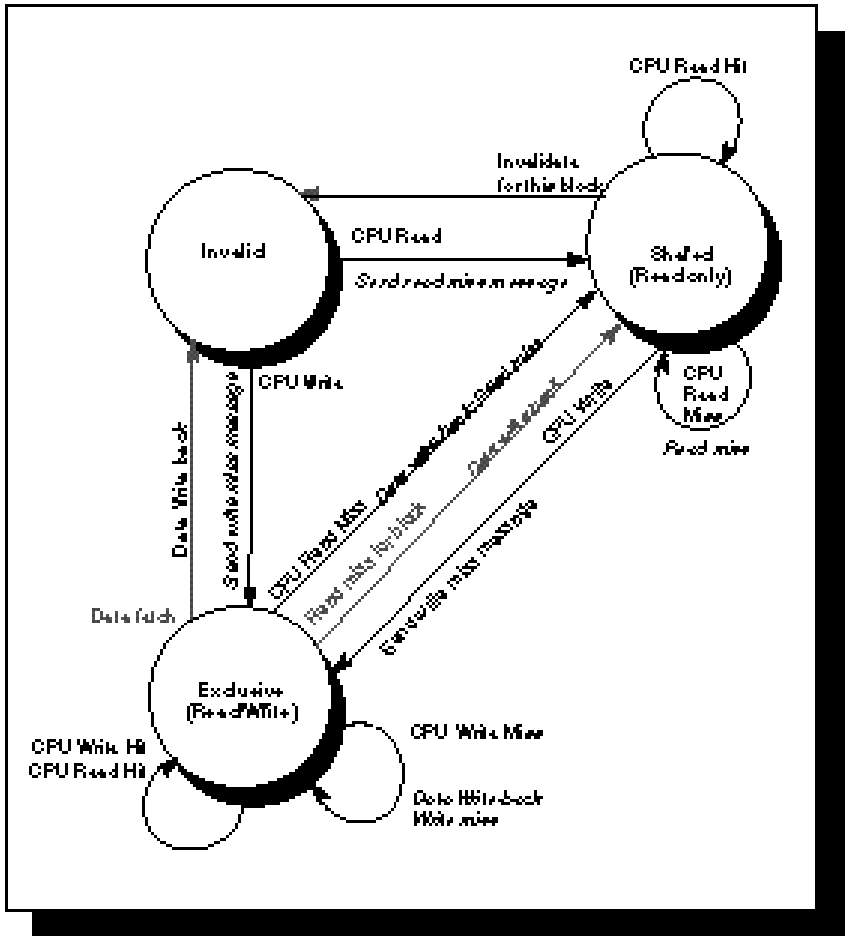
Review: Example Directory Protocol

- **Message sent to directory causes 2 acitons:**
  - **update the directory**
  - **more messages to satifty 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 is sent back the data from memory and the requestor is the only sharing node. The state of the block is made Shared.**
  - **Write miss: requesting processor is sent the value and 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 set is to identity of requesting processor. The state of the block is made Exclusive.**

Review: Example Directory Protocol

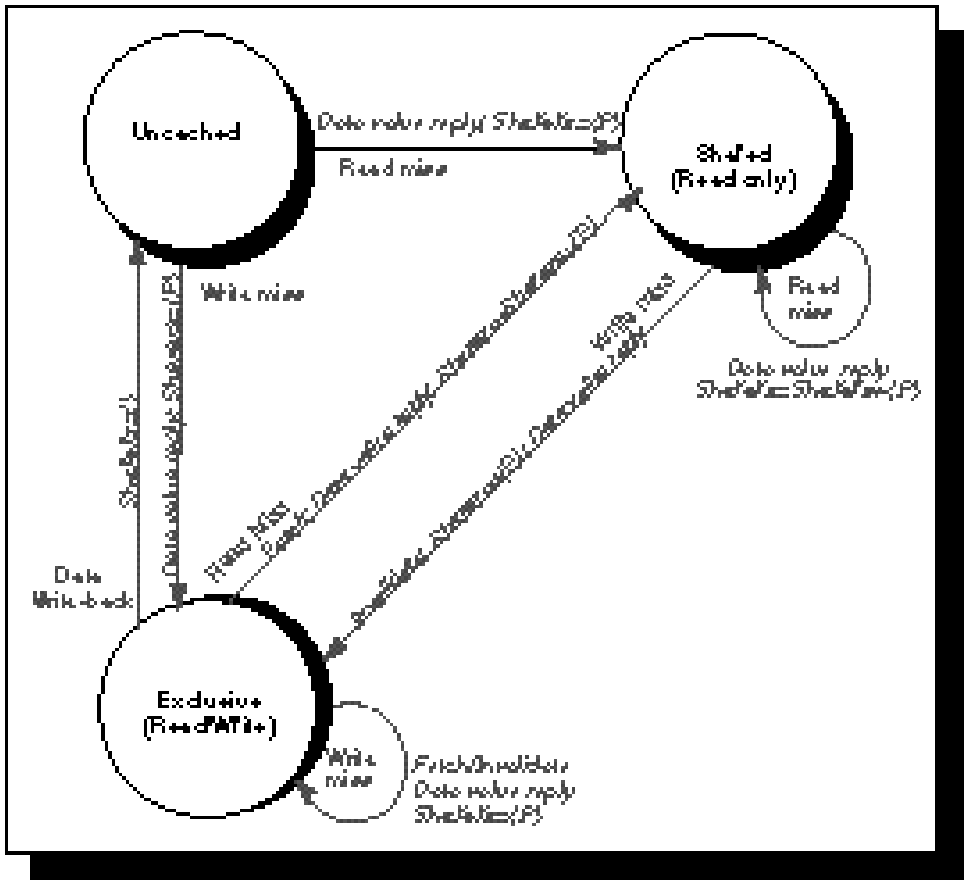
- **Block is Exclusive: current value of the block is held in the cache of the processor identified by the set Sharers (the owner), & 3 possible directory requests:**
  - **Read miss: owner processor is sent a data fetch message, which causes state of block in owner's cache to transition to Shared and causes owner to send data to directory, where it is written to memory and sent back to the 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).**
  - **Data write-back: owner processor is replacing the block and hence must write it back. This makes the 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 send to the requesting processor, which becomes the new owner. Sharers is set to identity of new owner, and state of block is made Exclusive.**

Review: State Transition Diagram for an Individual Cache Block in a Directory Based System



- **The states are identical to those in the snoopy case, and the transactions are very similar with explicit invalidate and write-back requests replacing the write misses that were formerly broadcast on the bus.**

Review: State Transition Diagram for the Directory

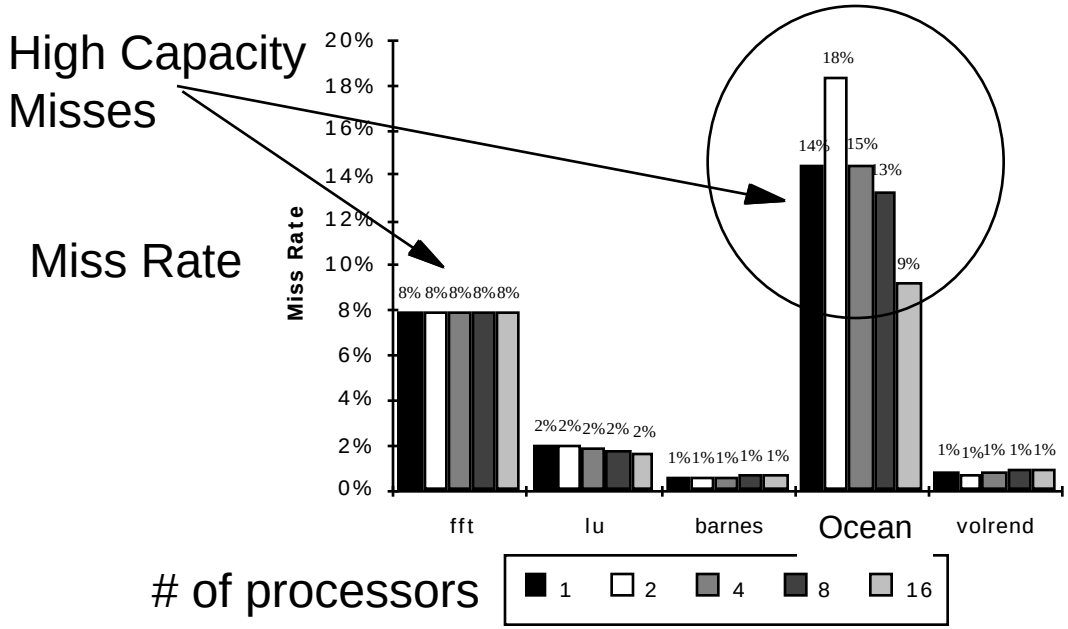


- **The same states and structure as the transition diagram for an individual cache**
  - **All actions are in color since they all are externally caused. Italics indicates the action taken the directory in response to the request. Bold italics indicate an action that updates the sharing set, Sharers, as opposed to sending a message.**

Miss Rates for Snooping Protocol

- 4th C: Conflict, Capacity, Compulsory and Coherency Misses
- More processors: increase coherency misses while decreasing capacity misses (for fixed problem size)
- Cache behavior of Five Parallel Programs:
  - FFT Fast Fourier Transform: Matrix transposition + computation
  - LU factorization of dense 2D matrix (linear algebra)
  - Barnes-Hut n-body algorithm solving galaxy evolution probem
  - Ocean simluates influence of eddy & boundary currents on large-scale flow in ocean: dynamic arrays per grid
  - VolRend is parallel volume rendering: scientific visualization

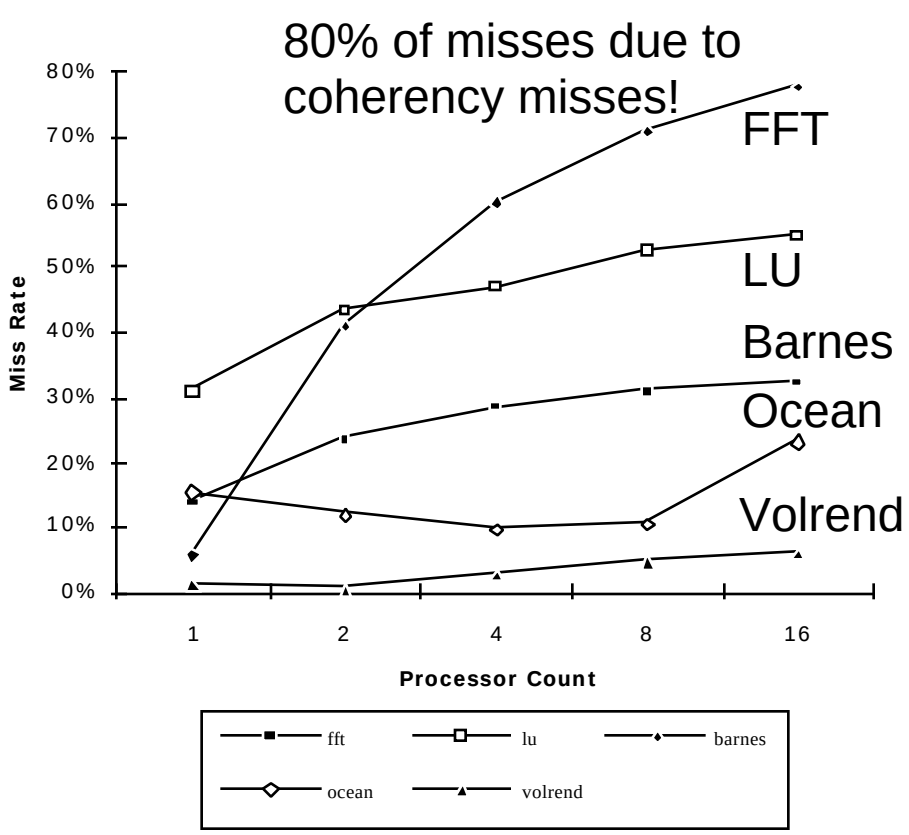
Miss Rates for Snooping Protocol



Big differences in miss rates among the programs

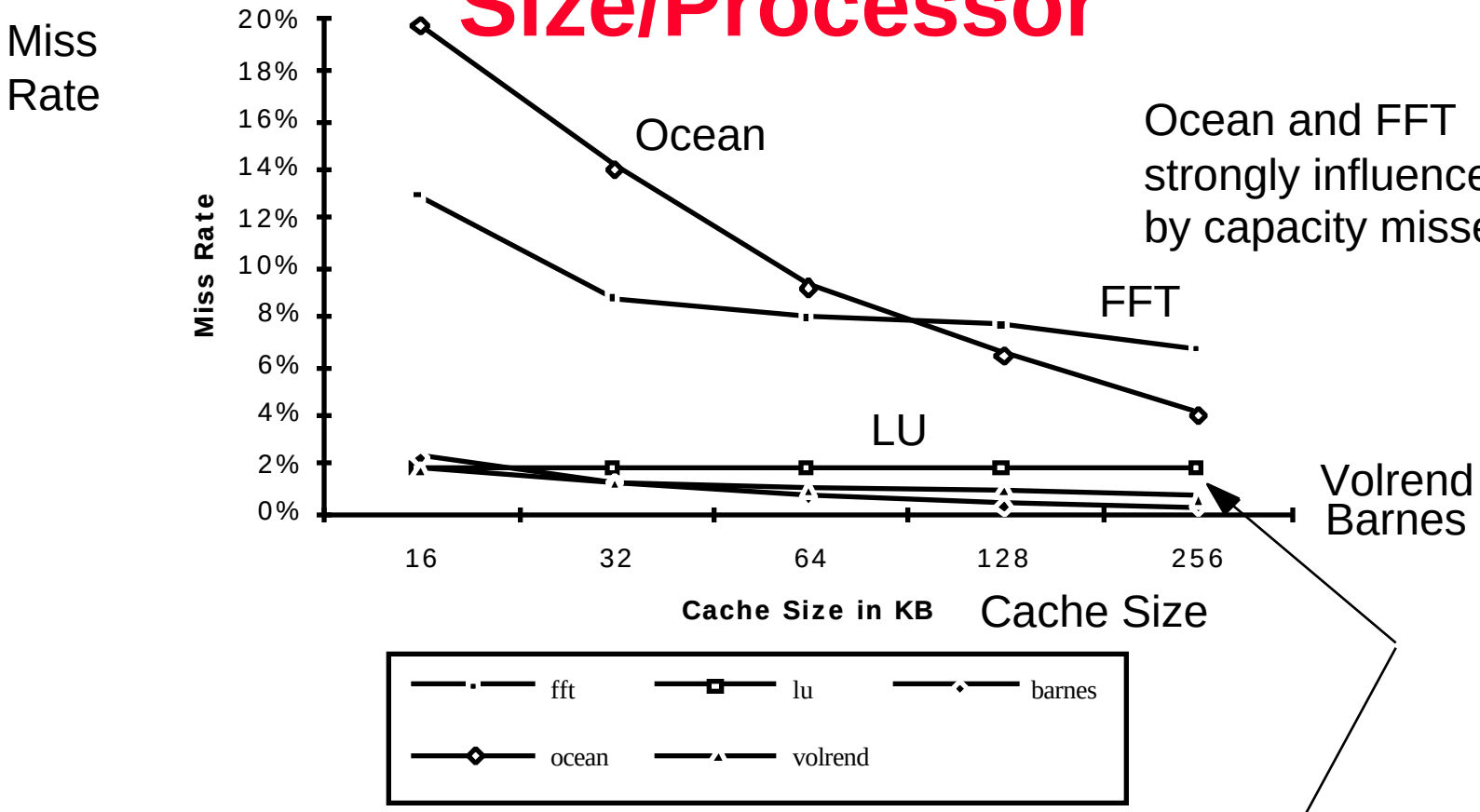
- **Cache size is 64KB, 2-way set associative, with 32B blocks.**
- **With the exception of Volrend, the misses in these applications are generated by accesses to data that is potentially shared.**
- **Except for Ocean, data is heavily shared; in Ocean only the boundaries of the subgrids are shared, though the entire grid is treated as a shared data object. Since the boundaries change as we increase the processor count (for a fixed size problem), different amounts of the grid become shared. The anamolous increase in miss rate for Ocean in moving from 1 to 2 processors arises because of conflict misses in accessing the subgrids.**

% Misses Caused by Coherency Traffic vs. # of Processors



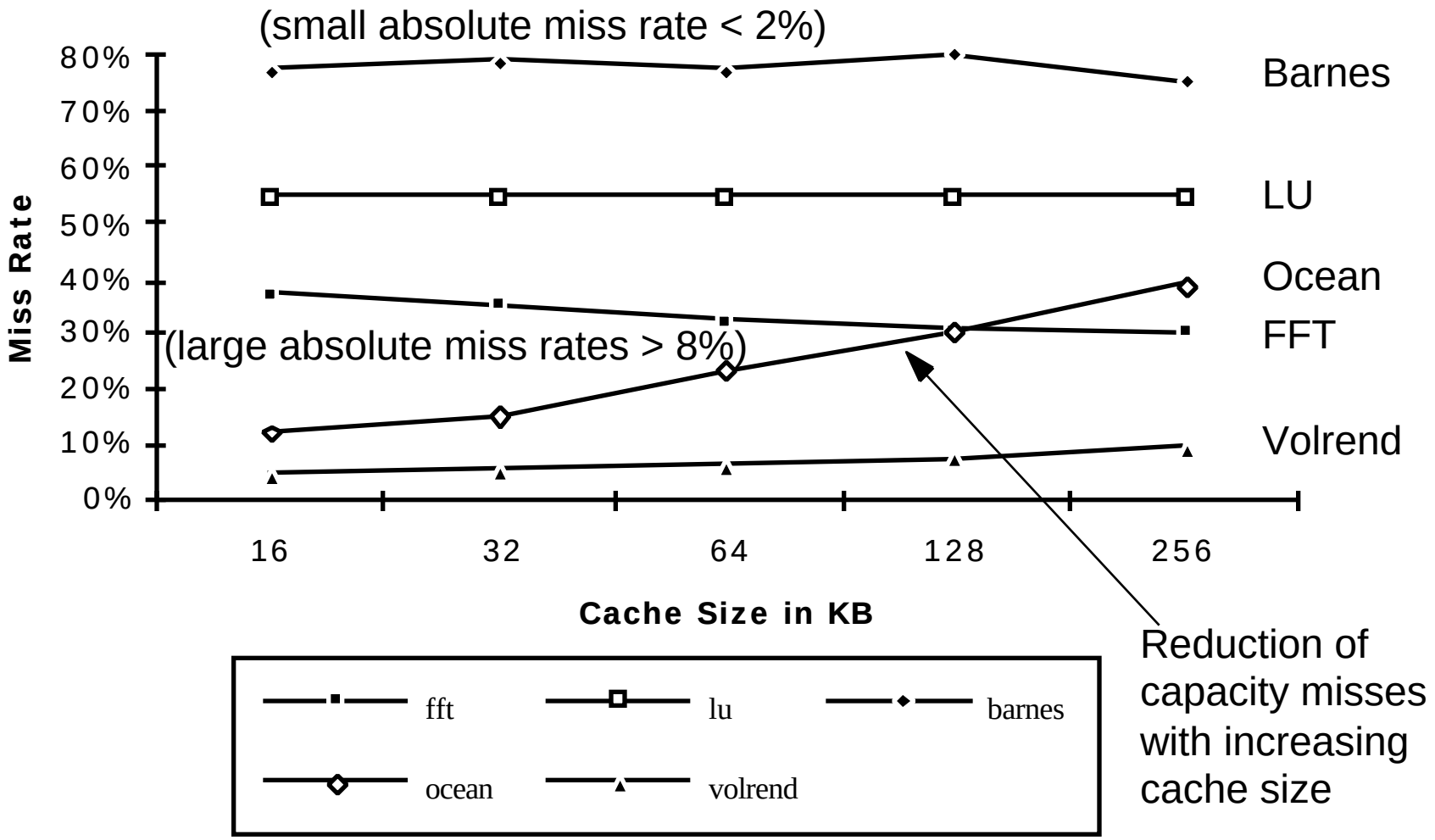
- **% cache misses caused by coherency transactions typically rises when a fixed size problem is run on more processors.**
- **The absolute number of coherency misses is increasing in all these benchmarks, including Ocean. In Ocean, however, it is difficult to separate out these misses from others, since the amount of sharing of the grid varies with processor count.**
- **Invalidations increases significantly; In FFT, the miss rate arising from coherency misses increases from nothing to almost 7%.**

Miss Rates as Increase Cache Size/Processor



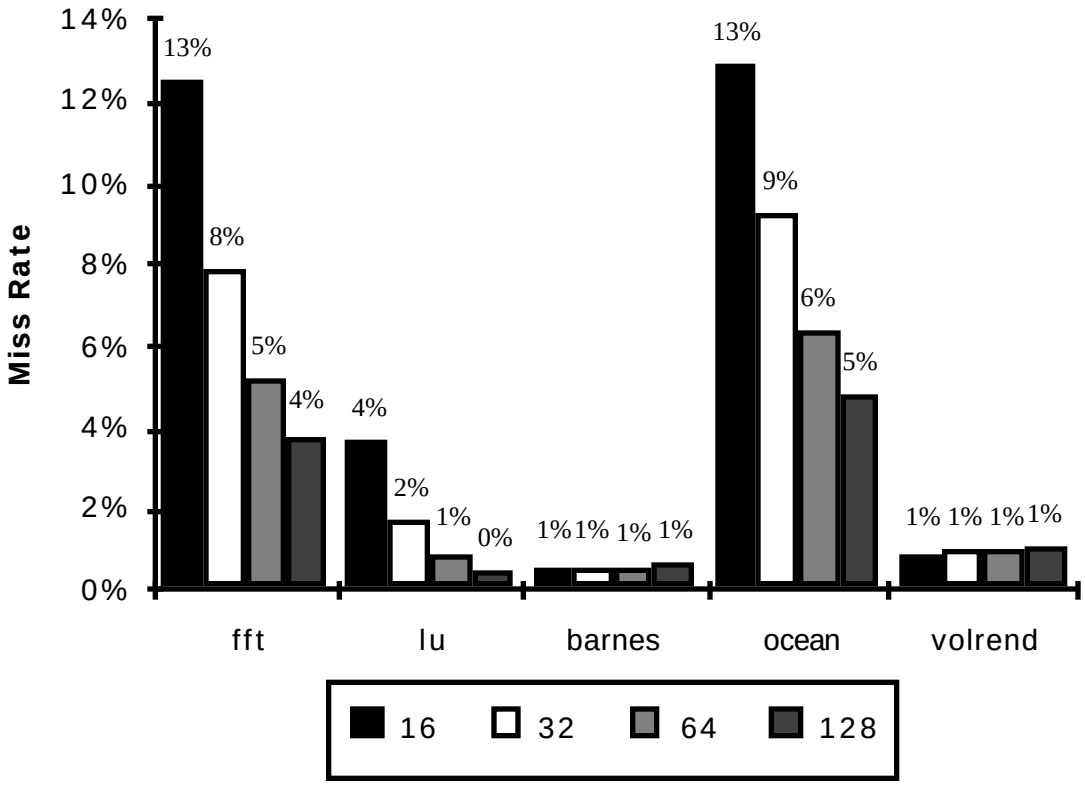
- **Miss rate drops as the cache size is increased, unless the miss rate is dominated by coherency misses.**
- **The block size is 32B & the cache is 2-way set-associative. The processor count is fixed at 16 processors.**

% Misses Caused by Coherency Traffic vs. Cache Size



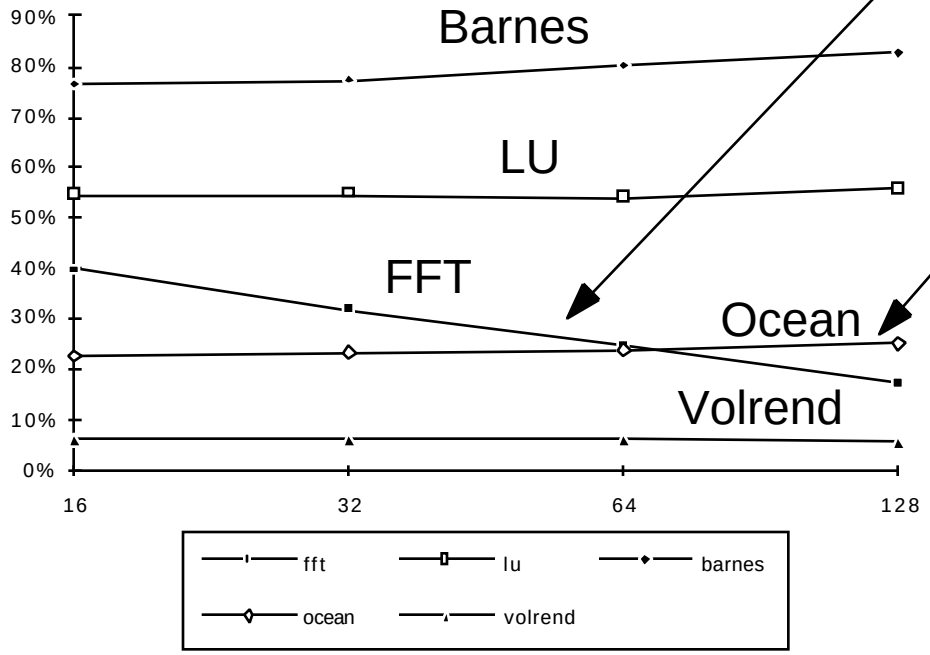
Miss Rate vs. Block Size

- **Since cache block hold multiple words, may get coherency traffic for unrelated variables in same block**
- **False sharing arises from the use of an invalidation-based coherency algorithm. It occurs when a block is invalidated (and a subsequent reference causes a miss) because some word in the block, other than the one being read, is written into.**



miss rates mostly fall with increasing block size

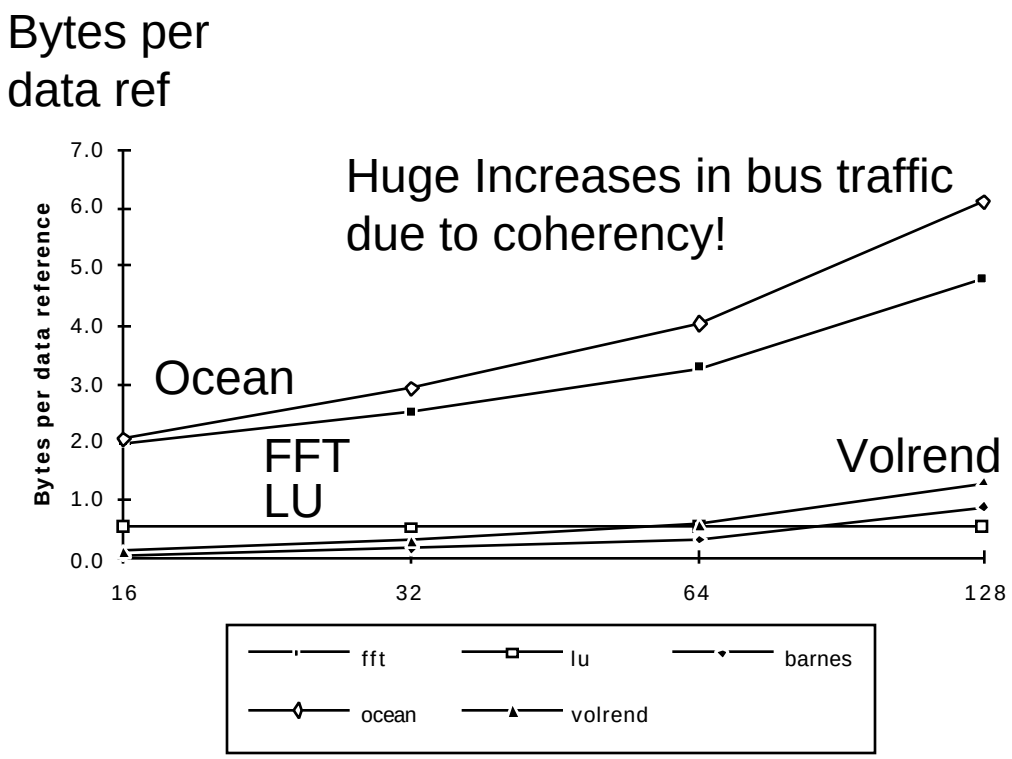
% Misses Caused by Coherency Traffic vs. Block Size



Behavior tracks cache size behavior
FFT: Coherence misses reduced faster than capacity misses!

- **FFT communicates data in large blocks & communication adapts to the block size (it is a parameter to the code); makes effective use of large blocks.**
- **Ocean competing effects that favor different block size**
  - **Accesses to the boundary of each subgrid, in one direction the accesses match the array layout, taking advantage of large blocks, while in the other dimension, they do not match. These two effects largely cancel each other out leading to an overall decrease in the coherency misses as well as the capacity misses.**

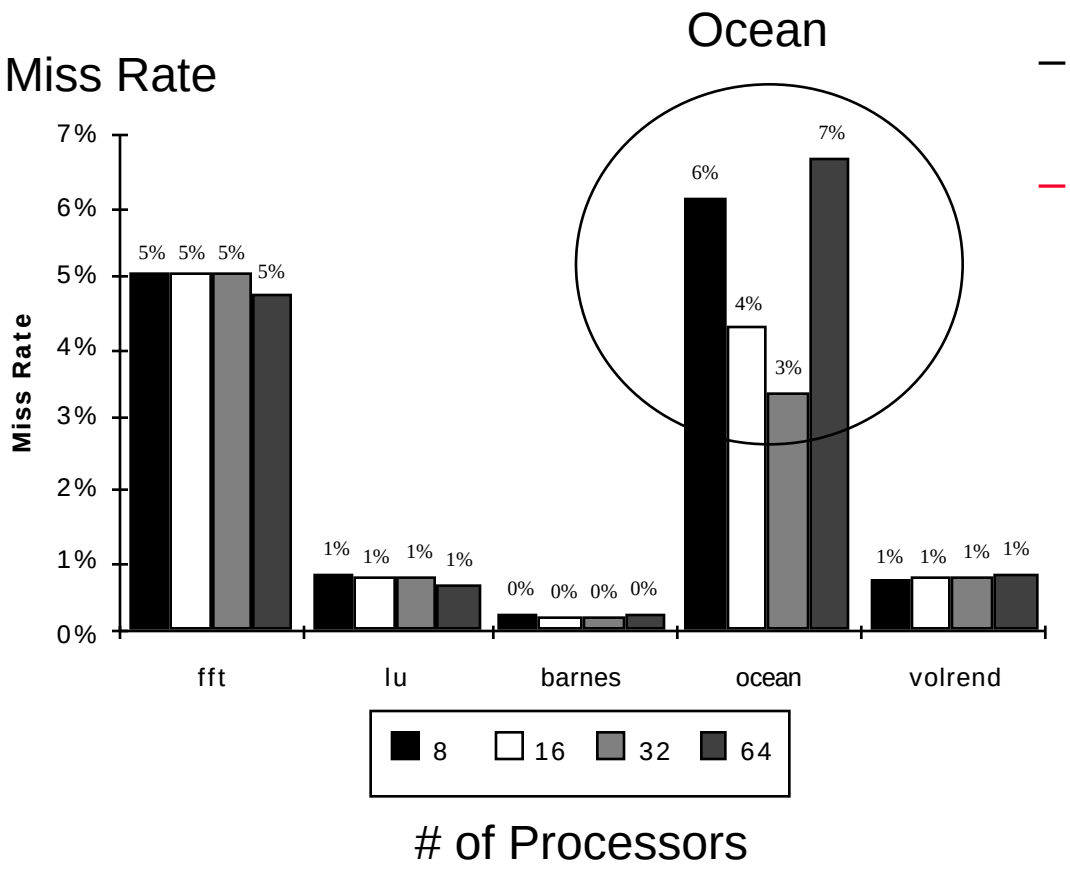
Bus Traffic as Increase Block Size



- **Bus traffic climbs steadily as the block size is increased.**
- **Volrend: the increase is more than a factor of 10, although the low miss rate keeps the absolute traffic small.**
- **The factor of 3 increase in traffic for Ocean is the best argument against larger block sizes.**
- **Remember that our protocol treats ownership misses the same as other misses, slightly increasing the penalty for large cache blocks: in both Ocean and FFT this effect accounts for less than 10% of the traffic.**

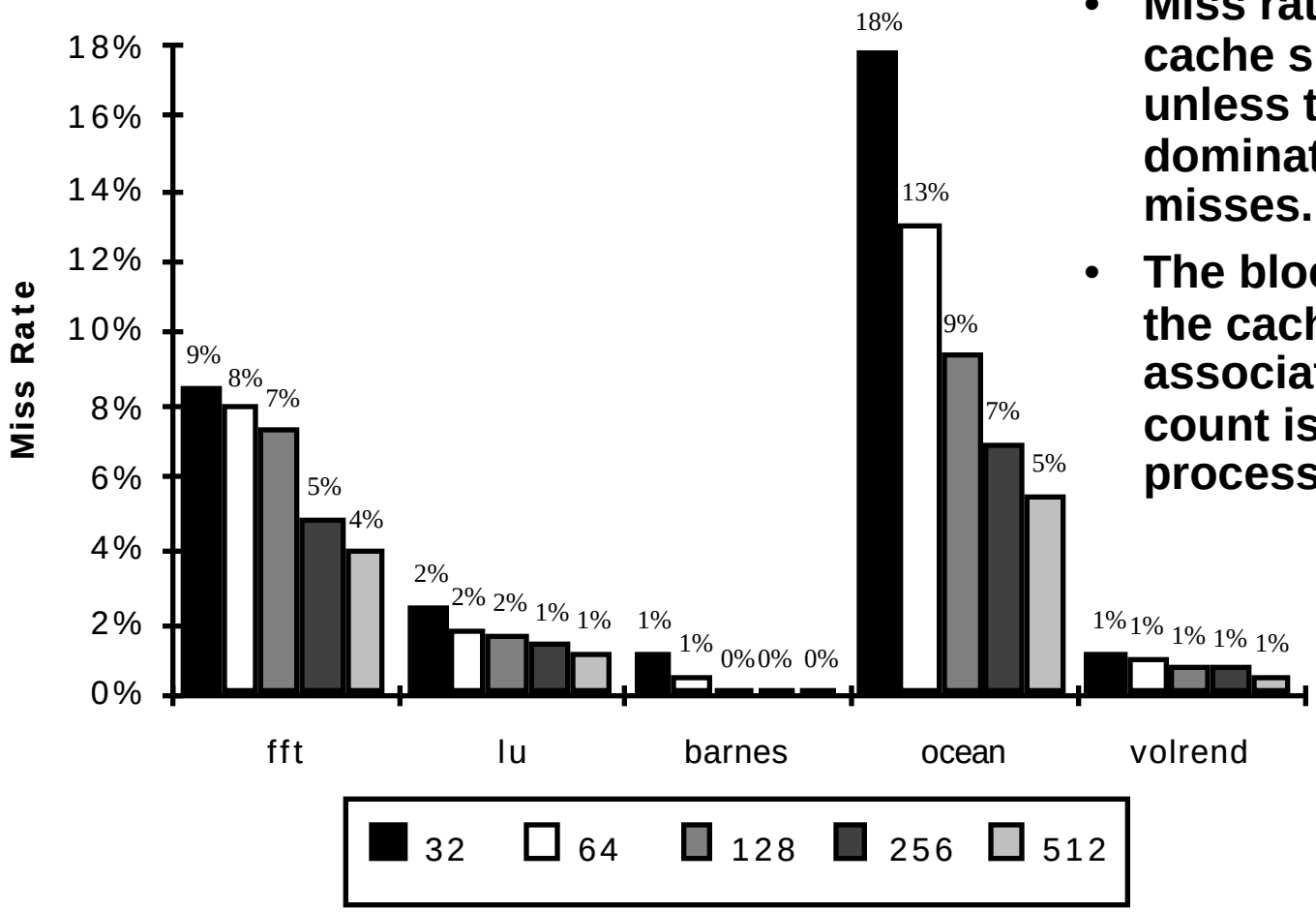
Miss Rates for Directory

Use larger cache to circumvent longer latencies to directories



- **Cache size is 128 KB, 2-way set associative, with 64B blocks.**
- **Ocean: only the boundaries of the subgrids are shared, though the entire grid is treated as a shared data object. Since the boundaries change as we increase the processor count (for a fixed size problem), different amounts of the grid become shared. The increase in miss rate for Ocean in moving from 32 to 64 processors arises because of conflict misses in accessing small subgrids & for coherency misses for 64 processors.**

Miss Rates as Increase Cache Size/Processor for Directory



- **Miss rate drops as the cache size is increased, unless the miss rate is dominated by coherency misses.**
- **The block size is 64B and the cache is 2-way set-associative. The processor count is fixed at 16 processors.**

Block Size for Directory

- **Assumes 128 KB cache & 64 processors**
  - **Large cache size to combat higher memory latencies than snoop caches**