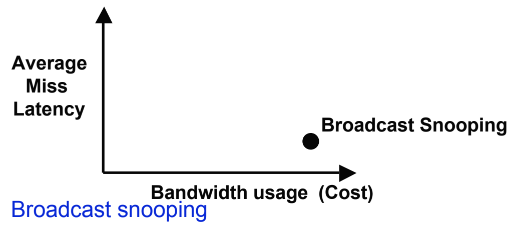
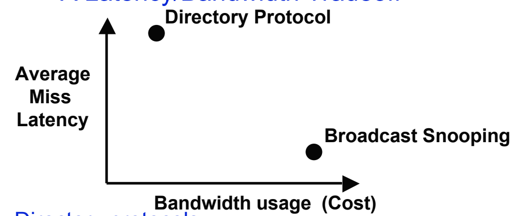
Cache-Coherence: A Latency/Bandwidth Tradeoff



- Broadcast snooping
  - Evolved from bus to switched interconnect
  - + **Direct data response (no indirection)**
  - – **Extra traffic (due to broadcast)**

Burns bandwidth for low latency

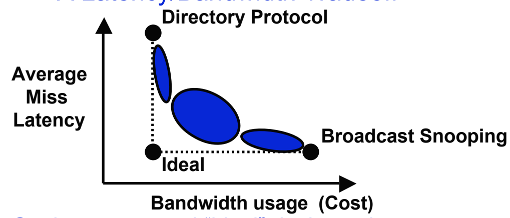
Cache-Coherence: A Latency/Bandwidth Tradeoff



- Directory protocols
  - Send to directory which forwards as needed
  - + **Avoids broadcast**
  - – **Adds latency for cache-to-cache misses**

Sacrifices latency for scalability

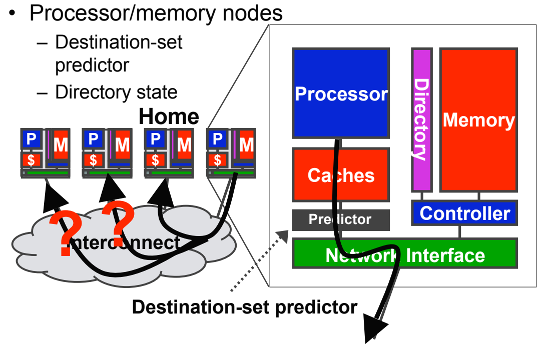
Cache-Coherence: A Latency/Bandwidth Tradeoff



- Goal: move toward "ideal" design point
- Approach: ***Destination-set Prediction***
  - Learn from past coherence behavior
  - Predict destinations for each coherence request
  - Correct prediction avoid indirection & broadcast

System Model

- Processor/memory nodes
  - Destination-set predictor
  - Directory state



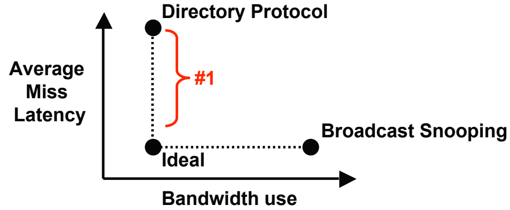
Contributions

- Destination-set predictors are:
  - **Simple:** single-level, cache-like structures
  - **Low-cost:** 64kBs (size similar to L2 tags)
  - **Effective:** significantly closer to "ideal"
- Exploit spatial predictability
  - Aggregate spatially-related information
  - Capture "spatial predictability" ▯ better accuracy
  - Reduces predictor sizes
- Workload characterization for predictor design
  - Commercial and technical workloads
  - See paper

Outline

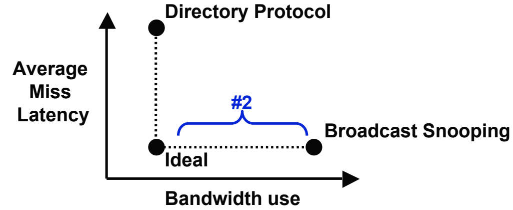
- Introduction
- **Quantifying potential benefit**
- Protocols
- Predictors
- Conclusions

Potential Benefit: Question 1 of 2



1. Significant performance difference? **Yes!**
   - Frequent cache-to-cache L2 misses (35%-96%)
   - Large difference in latency (2x or 100+ ns)
   - **Median of ~20% runtime reduction** (up to 50%)

Potential Benefit: Question 2 of 2

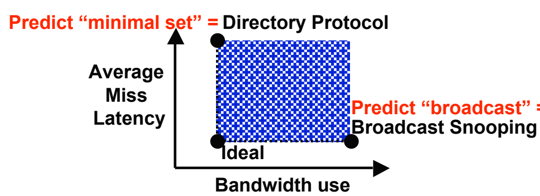


2. Significant bandwidth difference? **Yes!**
   - Only ~10% requests contact > 1 other processor
   - **Broadcast is overkill** (see paper for histogram)
   - The gap will grow with more processors

Outline

- Introduction
- Quantifying potential benefit
- **Protocols**
- Predictors
- Conclusions

Protocols for Destination-Set Prediction



- Protocol goals
  - Allow direct responses for "correct" predictions
  - No worse than base protocols
  - A continuum: erase snooping/directory duality

Protocols for Destination-Set Prediction

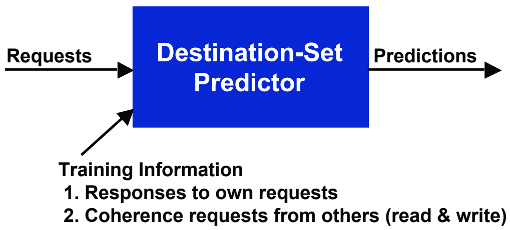
- Many possible protocols for implementation
  - Multicast snooping [Bilir *et al.*] & [Sorin *et al.*]
  - Predictive directory protocols [Acacio *et al.*]
  - Token Coherence [Martin *et al.*]
- Requestor predicts recipients
  - Always include directory + self ("minimal set")
- Directory at home memory ***audits*** predictions
  - Tracks sharers/owner (just like directory protocol)
  - **"sufficient"** □ acts as snooping (direct response)
  - **"insufficient"** □ acts as directory (forward request)

Protocol not the main focus of this work

Outline

- Introduction
- Quantifying potential benefit
- Protocols
- **Predictors**
- Conclusions

Predictor Design Space



- Basic organization
  - One predictor at each processor's L2 cache
  - Accessed in parallel with L2 tags
  - No modifications to processor core

Our Destination-Set Predictors

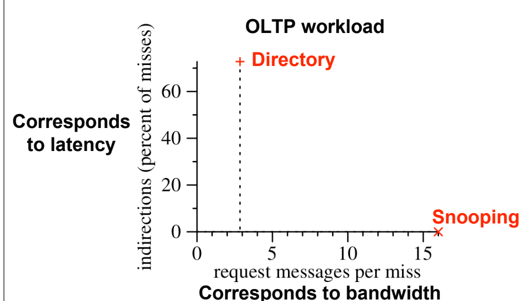
- All simple cache-like (tagged) predictors
  - Index with data block address
  - Single-level predictor
- Prediction
  - On tag miss, send to minimal set (directory & self)
  - Otherwise, generate prediction (as described next)
- Evaluation intermingled
  - Three predictors (more in paper)
  - Exploit spatial predictability
  - Limit predictor size
  - Runtime result

Evaluation Methods

- Six multiprocessor workloads
  - **Online transaction processing (OLTP)**
  - Java middleware (SPECjbb)
  - Static and dynamic web serving (Apache & Slash)
  - Scientific applications (Barnes & Ocean)
- Simulation environment
  - Full-system simulation using Simics
  - 16-processor SPARC MOSI multiprocessor
  - Many parameters (see paper)
  - **Traces** (for exploration) & **timing simulation** (for runtime results)

See "Simulating a $2M Server on $2K PC" [IEEE Computer, Feb 2003]

Trace-based Predictor Evaluation

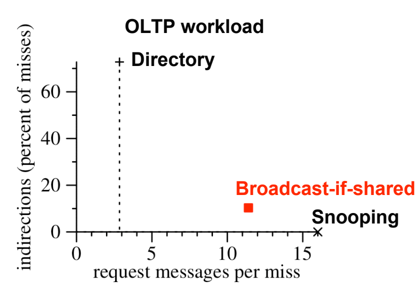


Quickly explore design space

Predictor #1: Broadcast-if-shared

- Performance of snooping, fewer broadcasts
  - Broadcast for "shared" data
  - Minimal set for "private" data
- Each entry: valid bit, 2-bit counter
  - Decrement on data from memory
  - Increment on data from a processor
  - Increment other processor's request
- Prediction
  - If *counter* > 1 then broadcast
  - Otherwise, send only to minimal set

Predictor #1: Broadcast-if-shared



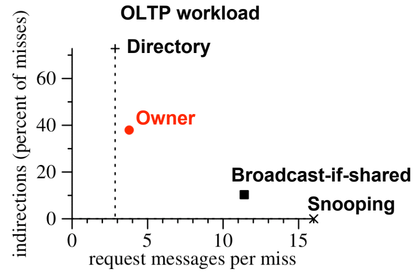
Unbounded predictor, indexed with datablock (64B)

Performance of snooping with less traffic

Predictor #2: Owner

- Traffic similar to directory, fewer indirections
  - **Predict one extra processor** (the "owner")
  - Pairwise sharing, write part of migratory sharing
- Each entry: valid bit, predicted *owner ID*
  - Set "owner" on data from other processor
  - Set "owner" on other's request to write
  - Unset "owner" on response from memory
- Prediction
  - If "*valid"* then predict "owner*"* + minimal set
  - Otherwise, send only to minimal set

Predictor #2: Owner

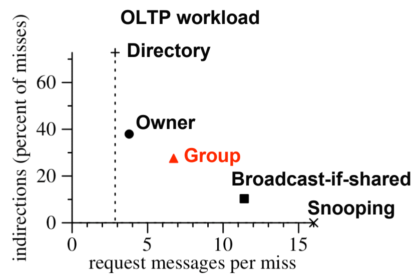


Traffic of directory with higher performance

Predictor #3: Group

- Try to achieve ideal bandwidth/latency
  - **Detect groups of sharers**
  - Temporary groups or logical partitions (LPAR)
- Each entry: N 2-bit counters
  - Response or request from another processor □ Increment corresponding counter
  - Train down by occasionally decrement all counters (every 2N increments)
- Prediction
  - Begin with minimal set
  - For each processor, if the corresponding counter > 1, add it in the predicted set

Predictor #3: Group

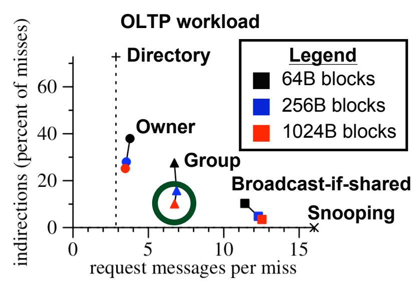


A design point between directory and snooping protocols

Indexing Design Space

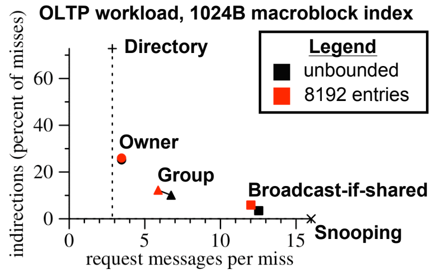
- Index by **cache block** (64B)
  - Works well (as shown)
- Index by **program counter** (PC)
  - Simple schemes not as effective with PCs
  - See paper
- Index by **macroblock** (256B or 1024B)
  - Exploit spatial predictability of sharing misses
  - Aggregate information for spatially-related blocks
  - E.g., reading a shared buffer, process migration

Macroblock Indexing



Macroblock indexing is an improvement
Group improves substantially (30% □ 10%)

Finite Size Predictors

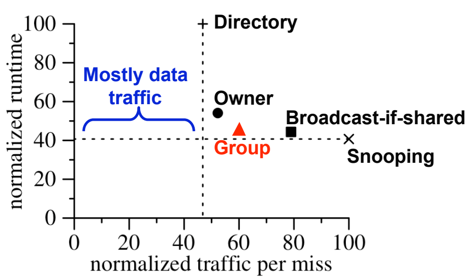


- **8192 entries → 32kB to 64kB predictor**
- **2-4% of L2 cache size (smaller than L2 tags)**

Runtime Results

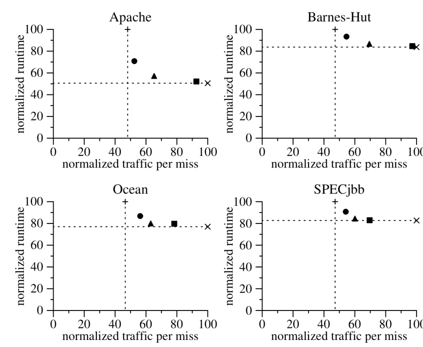
- What point in the design space to simulate?
  - As available bandwidth → infinite
    snooping performs best (no indirections)
  - As available bandwidth → 0,
    directory performs best (bandwidth efficient)
- Bandwidth/latency → cost/performance tradeoff
  - Cost is difficult to quantify (cost of chip bandwidth)
  - Other associated costs (snoop b/w, power use)
  - Bandwidth under-design will reduce performance
- Our evaluation: **measure runtime & traffic**
  - Simulate plentiful (but limited) bandwidth

Runtime Results: OLTP



- **1/2 runtime of directory, 2/3 traffic of snooping**

More Runtime Results



Conclusions

- Destination-set prediction is effective
  - Provides better bandwidth/latency tradeoffs
    (Not just the extremes of snooping and directory)
  - Significant benefit from macroblock indexing
  - Result summary: 90% the performance of snooping,
    only 15% more bandwidth than directory
- Simple, low-cost predictors
  - Many further improvements possible
- One current disadvantage: protocol complexity
  - Solution: use Token Coherence [ISCA 2003]