Bandwidth Adaptive Snooping

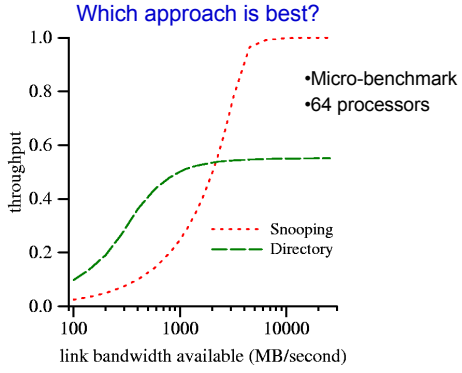
Milo M.K. Martin, Daniel J. Sorin
Mark D. Hill, and David A. Wood

Wisconsin Multifacet Project
Computer Sciences Department
University of Wisconsin—Madison

Two classes of multiprocessors

- Snooping (SMP) multiprocessors
  - – Broadcast-based → use more interconnect bandwidth
  - \+ Directly locate owner → low latency cache-to-cache transfers
    (36% - 91% of misses are cache-to-cache transfers in our commercial workloads)
- Directory-based multiprocessors
  - \+ Indirection → bandwidth-efficient & scalable
  - – Indirection → higher latency cache-to-cache transfers
- Problem: higher performing approach varies with:
  - – Configuration (e.g., number of processors)
  - – Workload (e.g., cache miss rate)

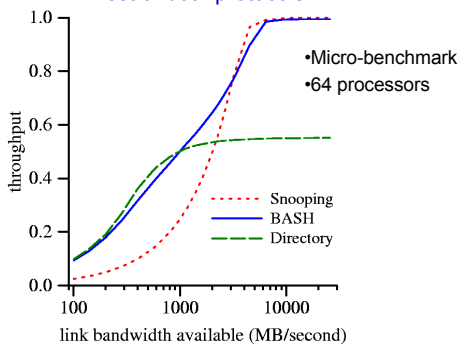
Which approach is best?



Bandwidth Adaptive Snooping Hybrid (BASH)

- Goals
  - – Best performance aspects of both approaches
    - High performance for many configurations & workloads
    - Future workload properties unknown at design time
  - – Single design
    - Coherence logic integrated with processors
    - One part for many systems
- Hybrid protocol
  - – Snooping-like broadcast requests
  - – Directory-like “unicast” requests
- Bandwidth adaptive
  - – Estimate available bandwidth
  - – Adjust rate of broadcast based on estimate

Best of both protocols

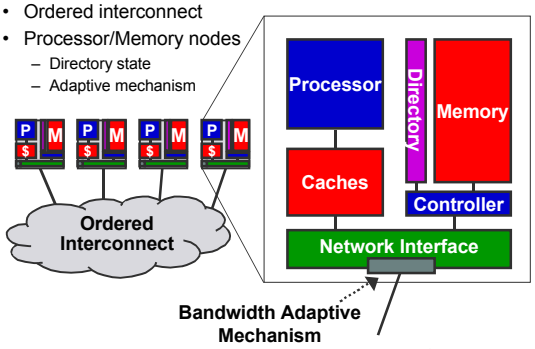


Outline

- Overview
- Bandwidth adaptive mechanism
- Hybrid protocol
- Evaluation
- Conclusions

System model

- Ordered interconnect
- Processor/Memory nodes
  - Directory state
  - Adaptive mechanism



Bandwidth adaptive mechanism

- Choose broadcast or unicast for each miss
- Goal: minimize latency - avoid extreme queuing delay
- Approach: limit average interconnect utilization
  - Contention dominates miss latency at high utilizations
  - Interconnect utilization goal (e.g., 75%)
  - Adjust rate of broadcast
  - Feedback control system

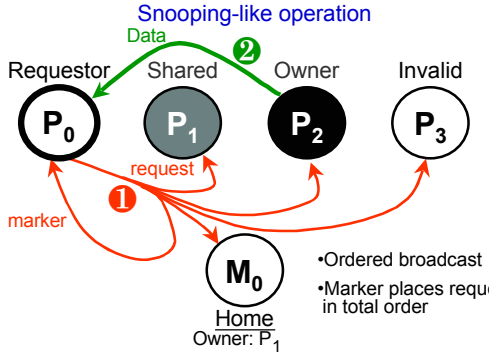
Implementation

- Two counters at each processor
  - Utilization counter *(Above or below utilization threshold?)*
  - Policy counter *(Probability of broadcast?)*
- At each processor
  - Each cycle: Monitor local link & adjust utilization counter
  - Each sampling interval: Adjust policy counter based on utilization counter
  - Each miss: Compare policy counter with a random number
- Why random?
  - Steady state of mixed broadcasts and unicasts
  - Enables us to avoid oscillation

Outline

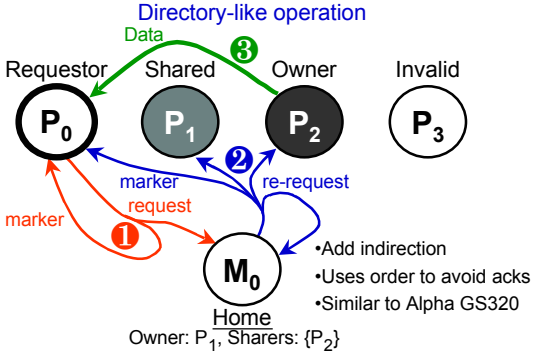
- Overview
- Bandwidth adaptive mechanism
- Hybrid protocol
  - Snooping-like operation
  - Directory-like operation
  - Complexity & Scalability
- Evaluation
- Conclusions

Snooping-like operation



Low latency cache-to-cache, but requires broadcast

Directory-like operation

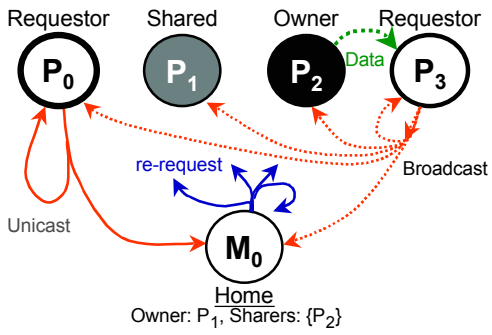


Avoids broadcast, but frequently adds indirection

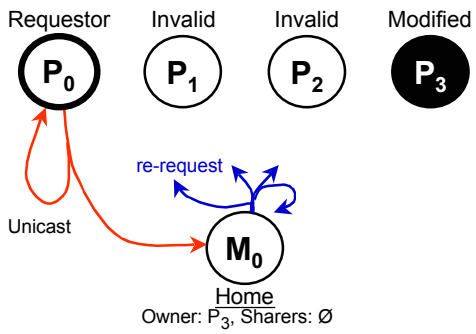
Protocol races

- Choose broadcast or unicast for each miss
- Protocol simultaneously allows
  - Broadcast requests
  - Unicast requests
  - Forwarded requests
  - Writebacks
- Like all protocols, BASH has protocol races

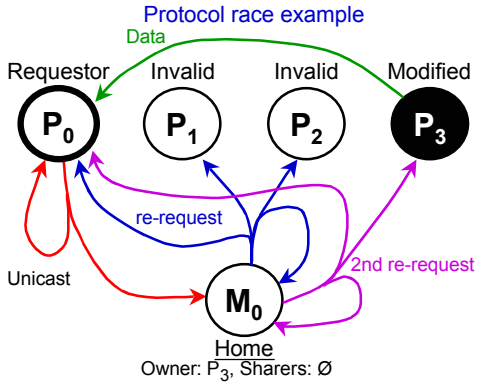
Protocol race example



Protocol race example



Protocol race example



Protocol races

- Race detection: directory audits all requests
  - Observes all requests
  - Compares request destination set with current sharers
  - Occasionally needs to re-issue a request
- Requests are processed uniformly
  - Processors - respond with data or invalidate
  - Directory - audit request, may forward data or request

See paper for more information

Complexity

- One “cost” of implementing BASH
- Quantifying complexity is difficult…
  - Protocol controllers are finite state machines
  - Similar number of *states*
  - BASH has twice as many *events* and *transitions*
- Moderate complexity
  - Additive, not multiplicative
- Similar to Multicast Snooping
  - Original proposal [Bilir et al., ISCA 1999]
  - Enhanced, specified & verified [Sorin et al., TPDS 2002]

Scalability

- Limited by ordered interconnect
  - BASH eliminates broadcast-only nature of snooping
- Recent systems with an ordered interconnect
  - Compaq AlphaServer GS320 (32 processor) - directory
  - Sun UE15000 (106 processors) - snooping
  - Fujitsu PrimePower 2000 (128 processors) - snooping
- Potential alternative
  - Timestamp Snooping network [Martin et al., ASPLOS 2000]

Outline

- Overview
- Bandwidth adaptive mechanism
- Hybrid protocol
- Evaluation
- Conclusions

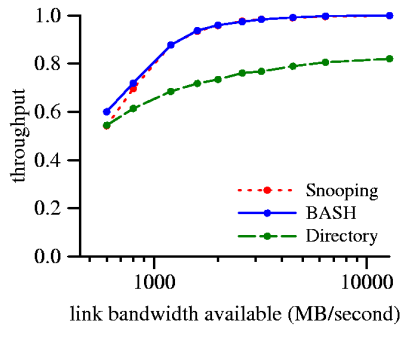
Workloads & methods

- Workloads [CAECW '02]
  - OLTP: IBM's DB2 & TPCC-like (1GB database)
  - Static web: Apache
  - Dynamic web: SlashCode
  - Java middleware: SpecJBB
  - Scientific workload: Barnes-Hut
- Setup and tuned for 16 processors
- Full system simulation
  - Virtutech's Simics
  - Solaris 8 on SPARC V9
  - Blocking processor model
- Memory system simulator
  - Captures timing, races, and all transient states

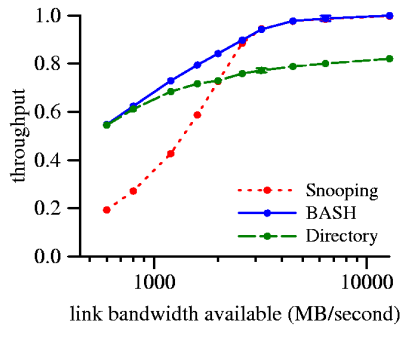
Three Questions

1) Is our adaptive mechanism effective?
2) Does BASH adapt to multiple workloads?
3) Does BASH adapt to multiple configurations?

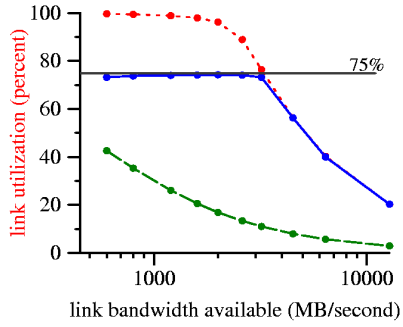
(1) SpecJBB on 16 processors



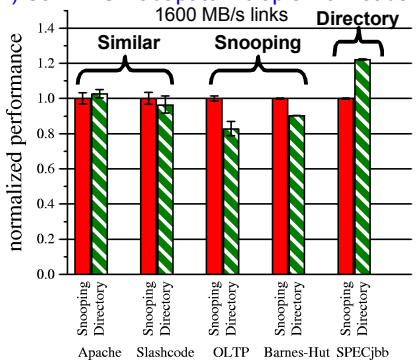
(1) SpecJBB on 16 processors, 4x broadcast cost



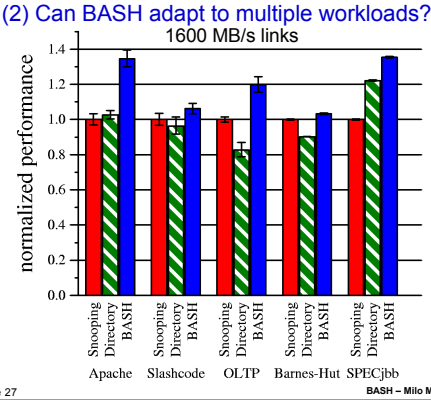
(1) SpecJBB on 16 processors, 4x broadcast cost



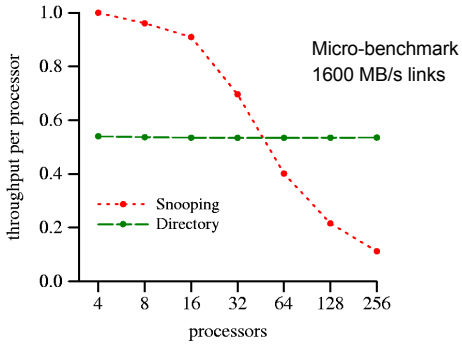
(2) Can BASH adapt to multiple workloads?



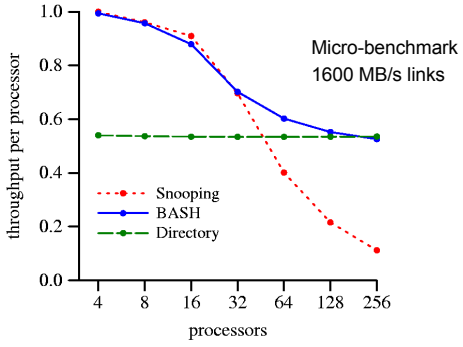
(2) Can BASH adapt to multiple workloads?



(3) Can BASH adapt to multiple configurations?



(3) Can BASH adapt to multiple configurations?



Results Summary

1) Is our adaptive mechanism effective?
   - Yes
2) Does BASH adapt to multiple workloads?
   - Yes
3) Does BASH adapt to multiple configurations?
   - Yes

Conclusions

- Bandwidth Adaptive Snooping Hybrid (BASH)
  - Hybrid of snooping and directories
  - Simple bandwidth adaptive mechanism
- Adapts to various workloads & system configurations
  - Robust performance
  - Outperforms base protocols in some cases
- Future directions
  - Focus bandwidth on likely cache-to-cache transfers
  - Explore multicasts
  - Power-adaptive coherence

Queuing model motivation

- A multiprocessor as a simple queuing model
  - Exponential service & think time distributions