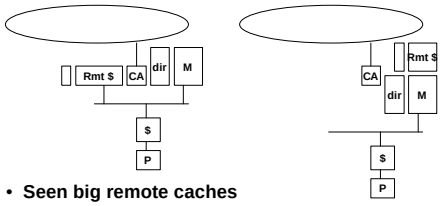
Design Alternatives for SAS: The Beauty of Mobile Homes

CS 258, Spring 99
David E. Culler
Computer Science Division
U.C. Berkeley

Some Questions you might ask

- Can all unnecessary communication be eliminated?
  - capacity-related communication?
  - false-sharing?
- How much hardware support can be eliminated?
- Can weak consistency models be exploited to reduce communication?
- Can we simplify hardware coherence mechanisms while avoiding capacity-related communication?

Overcoming Capacity Limitations



- Seen big remote caches
  - 32 MB on NUMA-Q
- What about using region of local mem as remote cache?
  - basic operation is to access mem. and check tag state
  - dispatch to specific protocol action

Well?

- Does it eliminate communication on capacity misses?
- How much of memory is wasted?
- Do we need a home memory at all?

Cache-Only Memory Arch (COMA)

- View entire local memory as 'attraction memory'
- Associate tag with every memory block
- Whenever a block as access and brought into cache, also bring it into local AMem.
  - keep cache consistent with AMem
  - keep Amem's consistent
- Location decoupled from physical address
- What new issues arise?
- Recall: basic components
  - find state information
  - find copies
  - communication with copies

Finding a block on a miss?

- Hierarchical Snooping COMA?
- Hierarchical Directory COMA?
- Flat Directory COMA?

Last Copy Replacement Problem

- How to avoid discarding the only copy of a block?
- Hierarchical?
- Flat?

Hardware/Software Tradeoffs?

- What aspects of SW are simplified?
- What extra HW support is required?
  - memory tags and comparators
  - memory for replication
- Why is HW support more complicated?
  - discover location on miss
  - no space preallocated

Performance Trade-Offs

- Artrifactual Communiation?
- Remote access latency?
- Local memory access latency?

Which access patterns win/lose

- Miss rate low?
- Miss rate ~low
  - mostly coherence misses
    - » true sharing
    - » false sharing
  - mostly capacity misses or poor initial data placement
    - » course grained
      - blocking
    - » fine grained
      - unpredictable

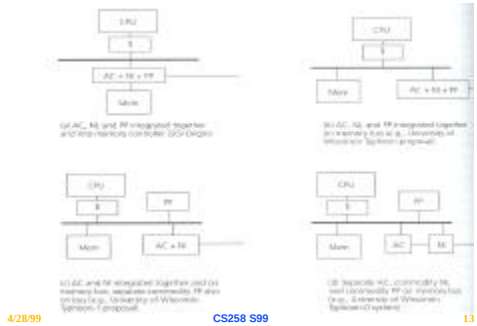
Plenty of replication memory

- Why is it important for performance?

Hardware support

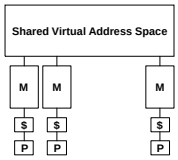
- Four fundamental components of the comm assist
  - access control checks
  - per-block tags and state used in the check
  - protocol processing
  - network interface
- Case for tight integration
  - access control check must see every load/store miss to shared data
  - requires checking per mem. block tag
  - protocol processing involves update cache/dir state
  - move small data items to/from NI
- Case against
  - L1 and L2 cache controller's on chip, closest you can get is the memory controller

Decoupled Assist Options



SAS w/o hardware support?

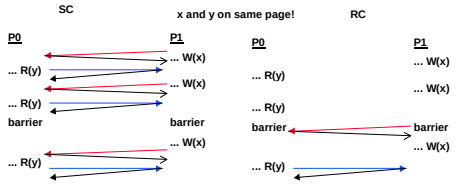
- Treat memory as fully-associative cache for global shared virtual address space
- Unit of coherence: page
- Basic components
  - Access control?
  - Tag and state check?
  - Protocol processing?
  - Communication?
- Problems?



Same virtual address represented at different physical addresses on each processor! - what needs to be invalidated?
Inclusion??

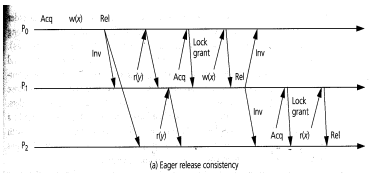
Exploiting Weak Consistency

- So far in HW approaches
  - changes when invalidations must be processed
  - avoid stalling processor while invalidations processed
  - still propagate invalidations ASAP
- Can invalidations be avoided?



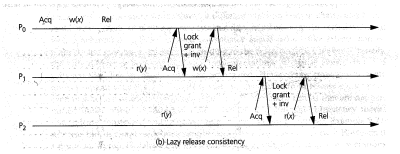
When should inv be propagated?

- Eager Release Consistency
  - At release propagate invalidations due to writes and wait for acks before proceding past the release point
  - conservative, actually!
    - » need to propagate before another processor does an acquire

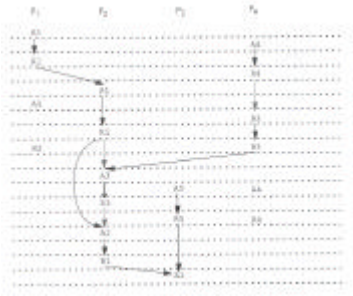


Lazy Release Consistency

- Associate invalidations with release
- On acquire, process invalidations for logically preceeding releases
  - apply them to relavant pages
- Respect program order and causal order



Causal Order



Relationship to HW coherence

- Is LRC coherent?
  - writes not propagated unless synchronization occurs
  - different processes may see writes through different synchronization chains
- LRC and RC are different consistency models!

```
P1                                  P2
lock L1;
ptr = non_null_ptr_val;
unlock L1;
                                    while (ptr == null) {};
                                    lock L1;
                                    a = ptr;
                                    unlock L1;
```


Multiple Writer Protocols

- How do we avoid false sharing traffic when two processors write to different locations in a page?

Options

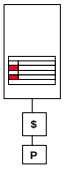
- Treadmarks:
  - protect page till first write
  - make twin at first write from processor
  - at release, compute diff.
    - » propagate at release, when demanded at acquire, ...
  - on page fault, collect and merge diffs from all copies
  - storage reclamation for diffs?
- Home based protocol
  - propagate diffs into home at release
  - on fault, get full page
- write-thru to other pages (Shrimp, Memory channel)

Further Weakening

- Entry consistency
  - associated regions with each synch variable
  - only propagate invalidations for associated region
- Jade
  - similar, but language construct
- Scope consistency

Middle Ground: Simple-COMA, Stache

- automatic migration at page level controlled in software
- fine grain access control in hardware
- page fault:
  - allocate page in local memory, but leave all blocks invalid
- page hit, cache miss:
  - access tag in parallel with memory access
    - » can be separate memory
  - physical address valid (not uniform)
  - on protocol transactions, reverse translate to shared virtual address
- No HW tag comparison. (just state)
- No local/remote check!



Conclusions

- Memory is a binding of names to values
- Modern shared address space designs deeply separate NAMES from LOCATIONS
- Share virtual addresses
- COMA has uniform virtual->'physical', but physical can migrate
- SVM use distinct virtual->physical to achieve page-level sharing
  - must exploit weak consistency models so artifactual communication doesn't dominate
- Simple-COMA uses mapping to simplify HW while providing migration!