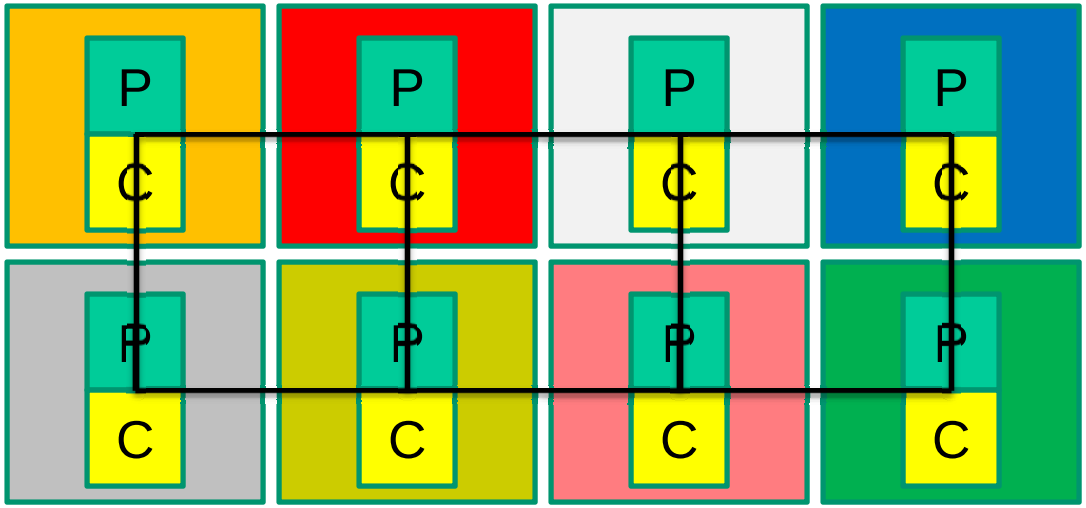
Lecture 17: Large Cache Design

Papers:

- Managing Distributed, Shared L2 Caches through OS-Level Page Allocation, Cho and Jin, MICRO'06
- Co-Operative Caching for Chip Multiprocessors, Chang and Sohi, ISCA'06
- Victim Replication, Zhang and Asanovic, ISCA'05

Page Coloring Example

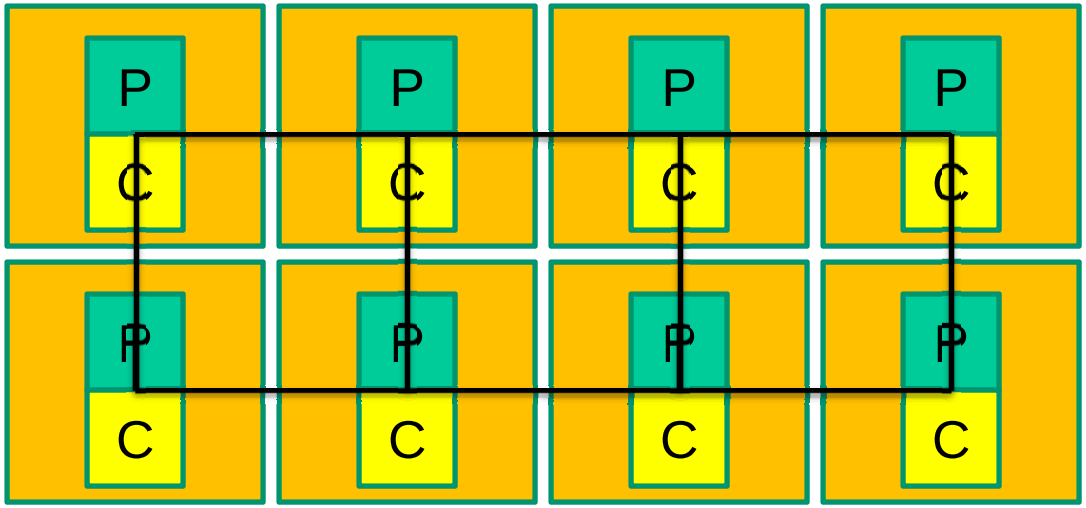


Cho and Jin, MICRO'06

- Page coloring to improve proximity of data and computation
- Flexible software policies
- Has the benefits of S-NUCA (each address has a unique location and no search is required)
- Has the benefits of D-NUCA (page re-mapping can help migrate data, although at a page granularity)
- Easily extends to multi-core and can easily mimic the behavior of private caches

Victim Replication

- Large shared L2 cache (each core has a local slice)
- On an L1 eviction, place the victim in local L2 slice (if there are unused lines)
- The replication does not impact correctness as this core is still in the sharer list and will receive invalidations
- On an L1 miss, the local L2 slice is checked before fwding the request to the correct slice



Static and Dynamic NUCA

- Static NUCA (S-NUCA)
  - The address index bits determine where the block is placed
  - Page coloring can help here as well to improve locality

- Dynamic NUCA (D-NUCA)
  - Blocks are allowed to move between banks
  - The block can be anywhere: need some search mechanism
  - Each core can maintain a partial tag structure so they have an idea of where the data might be (complex!)
  - Every possible bank is looked up and the search propagates (either in series or in parallel) (complex!)

Private L2s

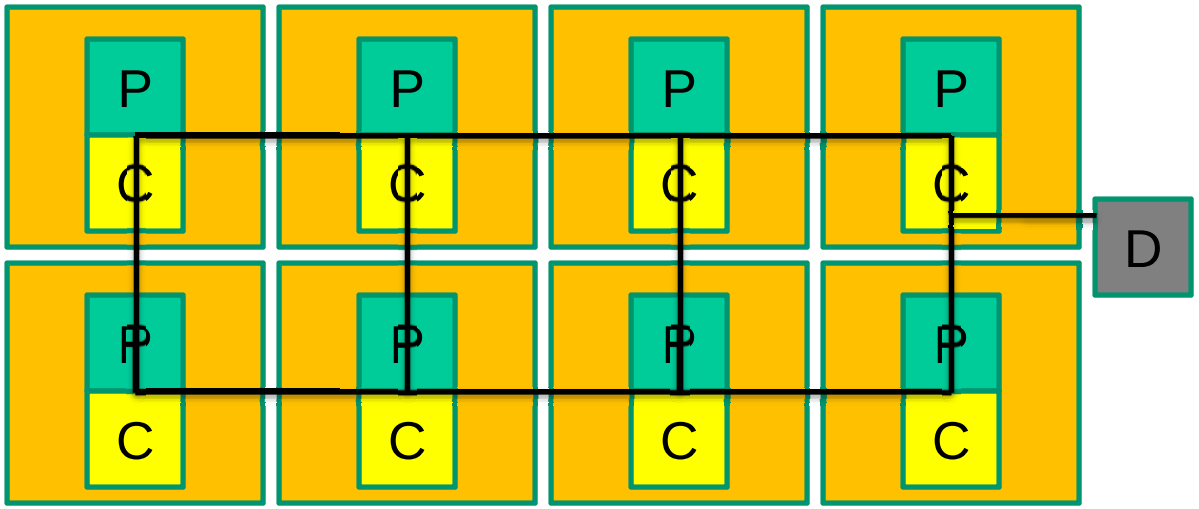
Arguments for private L2s:

- Lower latency for L2 hits
- Fewer ways have to be looked up for L2 hits
- Performance isolation (little interference from other threads)
- Can be turned off easily (since L2 does not have directory info)
- Fewer requests on the on-chip network

Primary disadvantage:

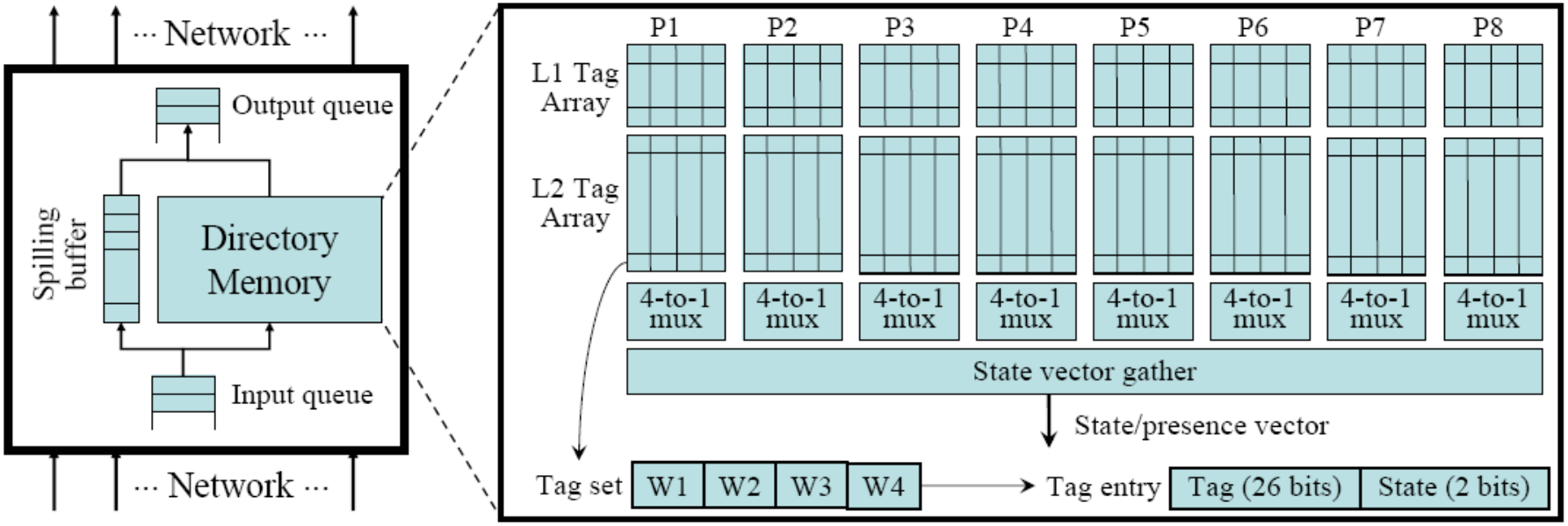
- More off-chip accesses because of higher miss rates

Coherence among L2s



- On an L2 miss, can broadcast request to all L2s and off-chip controller (snooping-based coherence for few cores)
- On an L2 miss, contact a directory that replicates tags for all L2 caches and handles the request appropriately (directory-based coherence for many cores)

The Directory Structure



- For 64-byte blocks, 1 MB L2 caches, overhead ~432 KB
- Note the complexities in maintaining presence vectors, non-inclusion for L1 and L2
- Note that clean evictions must also inform the central directory
- Need not inform directory about L1-L2 swaps (the directory is imprecise about whether the block will be found in L1 or L2)

Co-operation I

- Cache-to-cache sharing
- On an L2 miss, the directory is contacted and the request is forwarded to and serviced by another cache
- If silent evictions were allowed, some of these forwards would fail

Co-operation II

- Every block keeps track of whether it is a *singlet* or *replicate* – this requires notifications from the central directory every time a block changes modes
- While replacing a block, replicates are preferred (with a given probability)
- When a singlet block is evicted, the directory is contacted and the directory then forwards this block to another randomly selected cache (weighted probabilities to prefer nearby caches or no cache at all) (hopefully, the forwarded block will replace another replicate)

Co-operation III

- An evicted block is given a *Recirculation Count* of N and pushed to another cache – this block is placed as the LRU block in its new cache – every eviction decrements the RC before forwarding (this paper uses N=1)

- Essentially, a block has one more chance to linger in the cache – it will stick around if it is reused before the new cache experiences capacity pressure

- This is an attempt to approximate a global LRU policy among all 32 ways of aggregate L2 cache

- Overheads per L2 cache block: one bit to indicate “once spilled and not reused” and one bit for “singlet” info

Results

Table 5. Multithreaded Workload Miss Rate and L1 Miss Breakdown

	Thousand misses per transaction Off-chip (Private / Shared / CC)	L1 Misses breakdown (Private / Shared / CC)		
		Local L2	Remote L2	Off-chip
OLTP	9.75 / 3.10 / 3.80	90% / 15% / 86%	7% / 84% / 13%	3% / 1% / 1%
Apache	1.60 / 0.90 / 0.94	65% / 9% / 51%	15% / 77% / 36%	20% / 14% / 13%
JBB	0.13 / 0.08 / 0.10	72% / 10% / 57%	14% / 80% / 32%	14% / 10% / 11%
Zeus	0.71 / 0.46 / 0.49	67% / 9% / 45%	15% / 78% / 41%	19% / 12% / 13%

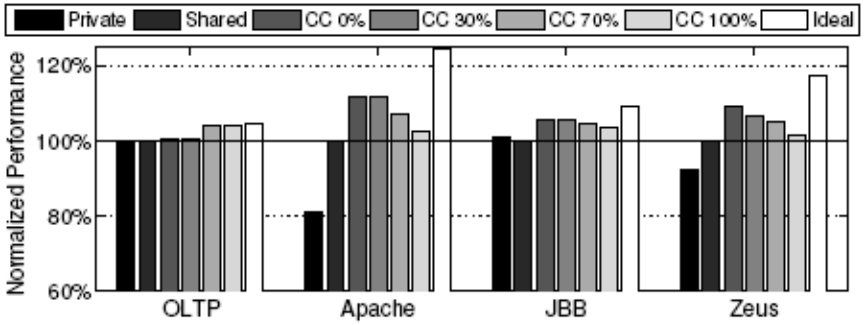


Figure 4. Multithreaded Workload Performance

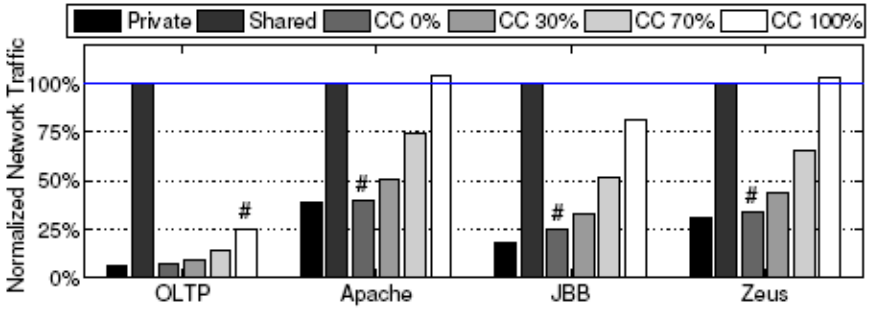


Figure 6. Multithreaded Workload Bandwidth ("#" indicates the best performing CC scheme)

Results

Table 6. Multiprogrammed Workload Miss Rate and L1 Miss Breakdown

	Misses per thousand instructions Off-chip (Private / Shared / CC)	L1 Misses breakdown (Private / Shared / CC)		
		Local L2	Remote L2	Off-chip
Mix1	3.1 / 2.0 / 2.4	78% / 19% / 67%	3% / 73% / 22%	19% / 9% / 11%
Mix2	3.0 / 1.6 / 1.8	64% / 35% / 75%	4% / 55% / 14%	32% / 9% / 11%
Mix3	1.2 / 0.7 / 0.8	91% / 20% / 87%	1% / 77% / 9%	7% / 3% / 4%
Mix4	0.6 / 0.3 / 0.3	95% / 12% / 90%	0% / 86% / 8%	4% / 2% / 2%
Rate1	0.8 / 0.6 / 0.8	90% / 20% / 80%	3% / 76% / 13%	7% / 4% / 6%
Rate2	53 / 51 / 41	31% / 7% / 24%	11% / 47% / 34%	58% / 46% / 42%

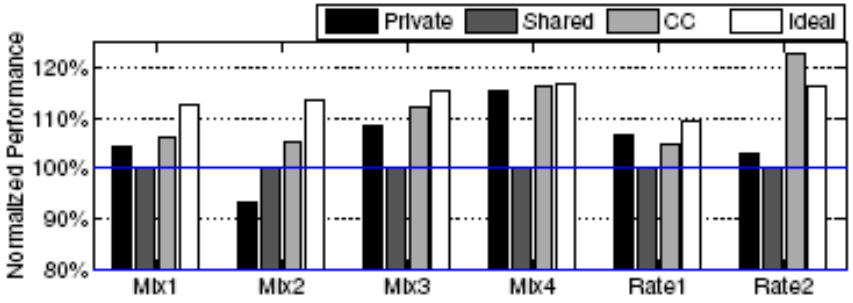


Figure 7. Multiprogrammed Workload Performance

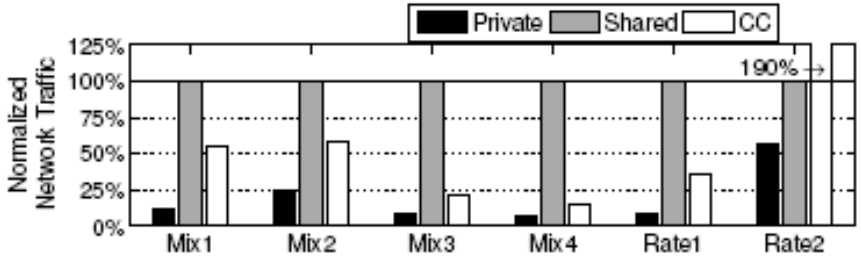


Figure 9. Multiprogrammed Workload Bandwidth

Title

- Bullet