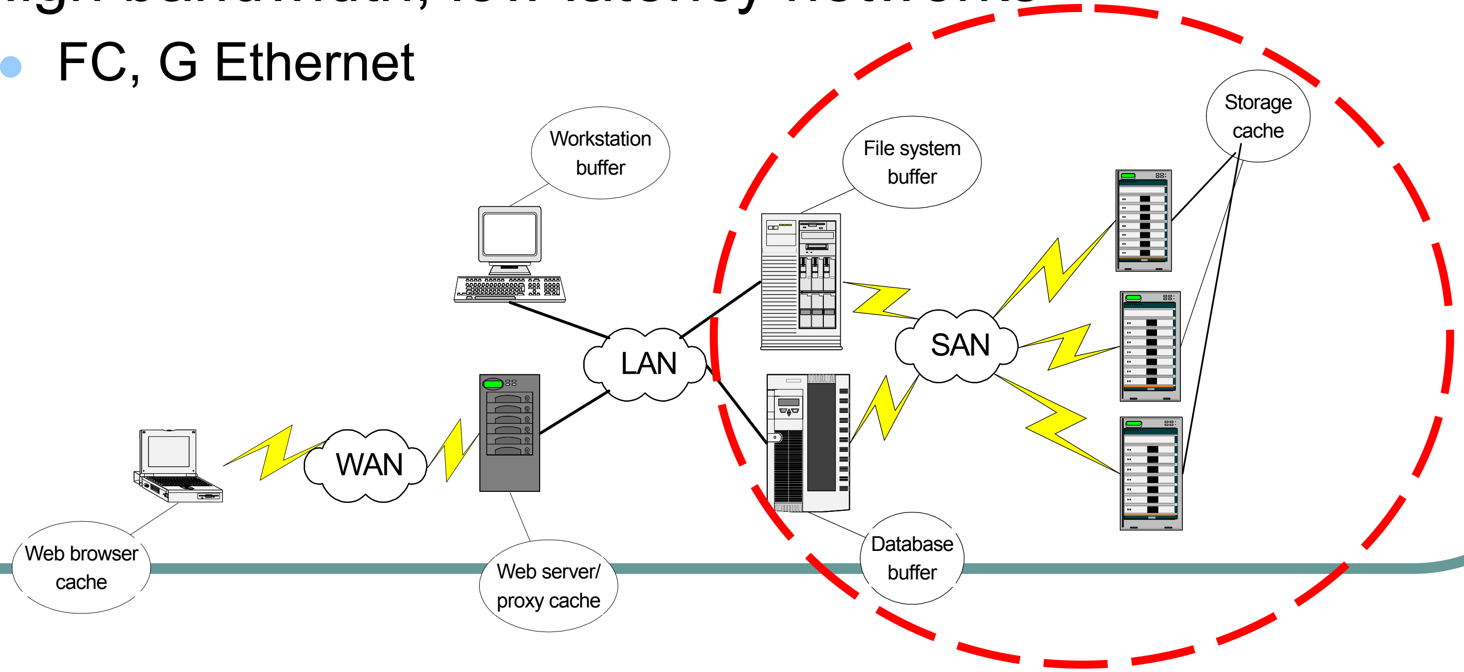
Collaborative buffer caches in data centers

Zhifeng Chen and Yuanyuan Zhou

University of Illinois, Urbana-Champaign

Buffer Cache Hierarchy

- Multiple heterogeneous systems in data centers:
  - Storage, database, filesystem, web service, etc.
- Gigabytes of memory for buffer caching:
  - Reduce out-going requests of individual server.
- High bandwidth, low latency networks
  - FC, G Ethernet



Inefficiency of the hierarchy

- Upper level buffer filtering effect:
  - Lower level buffer suffers high miss ratio.
  - Improved algorithms to achieve exclusive caching. (MQ, etc.)
- Lower level buffer caching effect:
  - Different response time for hits and misses.
  - DB/FS buffer reduces network messages instead of disk I/O.

Content-aware caching

- Basic idea: buffer caches knows the content of the other buffer cache.
  - Generalize exclusive caching.
- Different from cooperative caching [Dahlin94]
  - Client-client vs. client-server.
  - Heterogeneous software vs. homogeneous software.

Tracking buffer content

- Message exchange
  - Updates meta-data periodically with approximation.
  - Piggyback meta-data delta on replies.
  - Reduce space overhead using the bloom filter.
- Estimation
  - One buffer cache emulates other buffer caches.
  - Gray box probing can obtain adequate parameters.
  - *Need internal knowledge of other buffer caches.*

Explore neighbor knowledge

- Eviction based placement [Chen03]
  - Storage cache reloads evicted DB/FS pages
- Preferential caching
  - Replacement prefers the block in both level buffer caches.

Next access: d

Higher level: | ~~a~~ | b

Lower level: | b | c

LRU: evict a

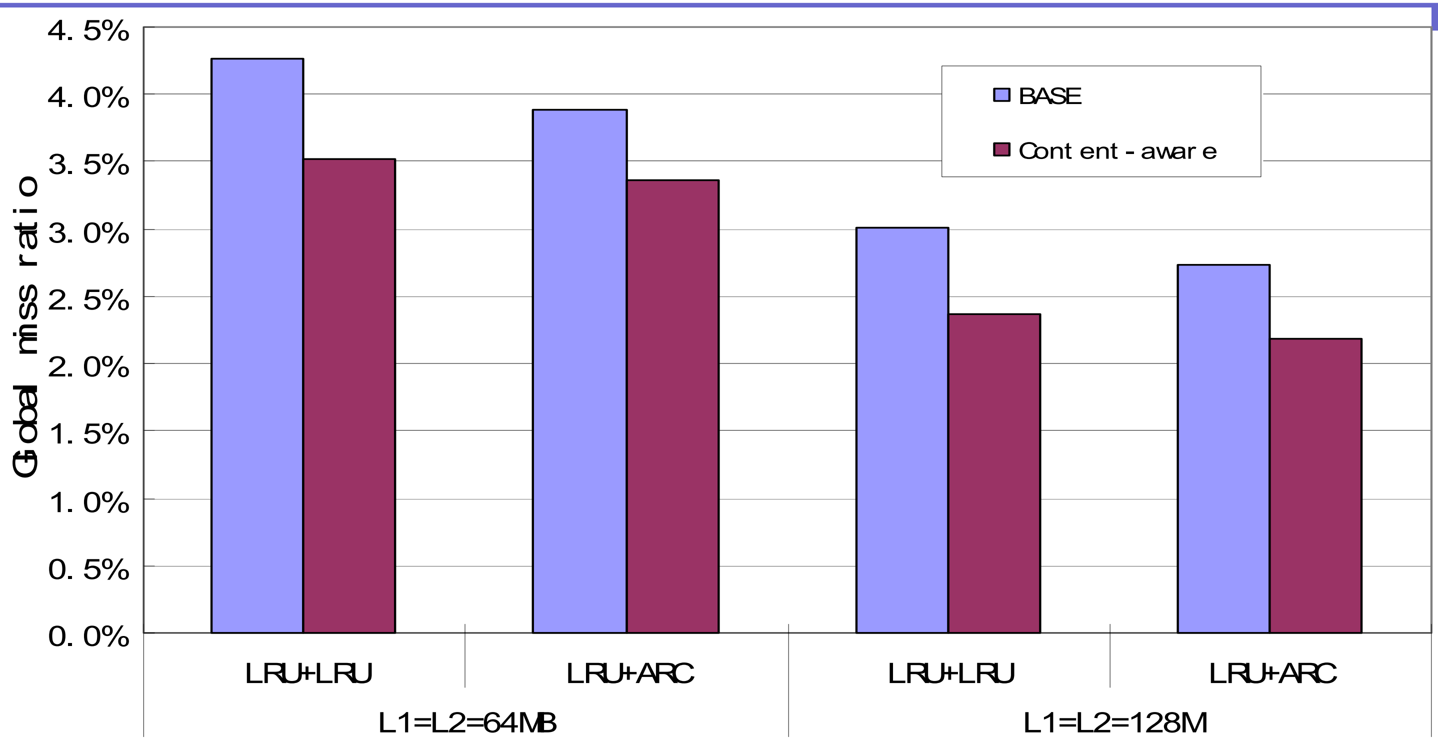
LRUCA: evict b

- Questions:
  - Which is better? Or, do both?
  - What is the optimal scheme with global knowledge?

Transparent deployment

- *CacheLib*: Everyone use the same module
  - A flexible tool to construct various buffer caches.
  - Very small overhead.

Preliminary result



- File system trace simulation: Auspex.
- Database buffer pool trace simulation: DB2 TPC-C, 80 warehouse, 2 hours.
- 10%-20% less disk I/O
- More results in the poster.

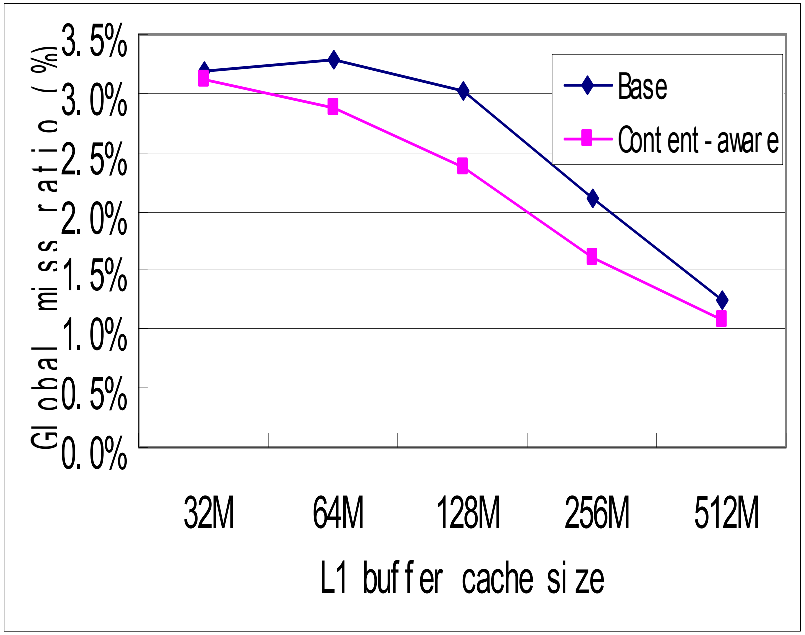
Related work

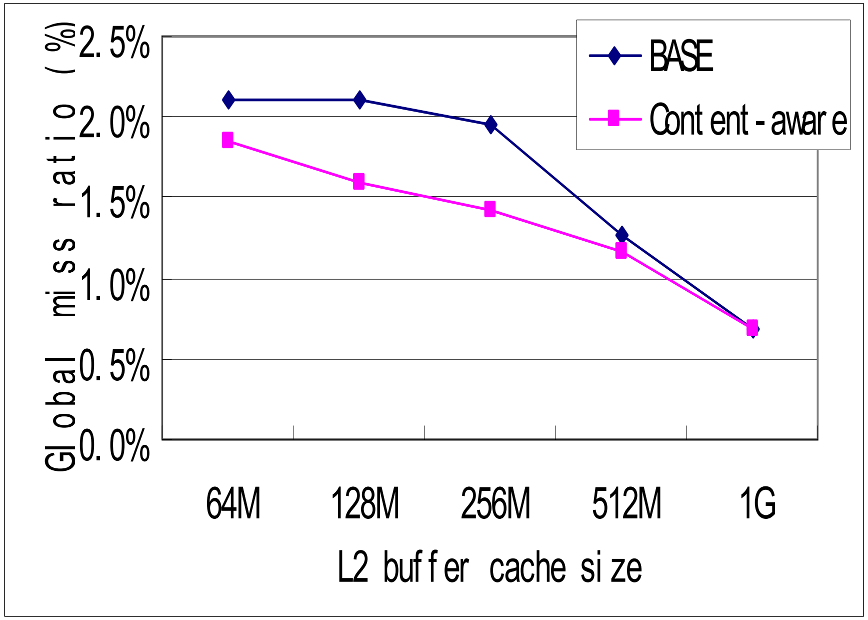
- Network file systems [Dahlin94, Feeley95, Sarkar96]
- Web caching [Karger99,Fan00]
- Database buffer management [Jauhari90]
- Storage cache [Zhou01,Wong02,Chen03]

Conclusion

- Content-aware caching is potential to explore the aggregate large buffer cache in a data center.
- Future work:
  - Application performance effect.
  - Content aware CLOCK.
  - Automatic detection of remote buffer for less tuning effort.

Disk I/O reduction: File system



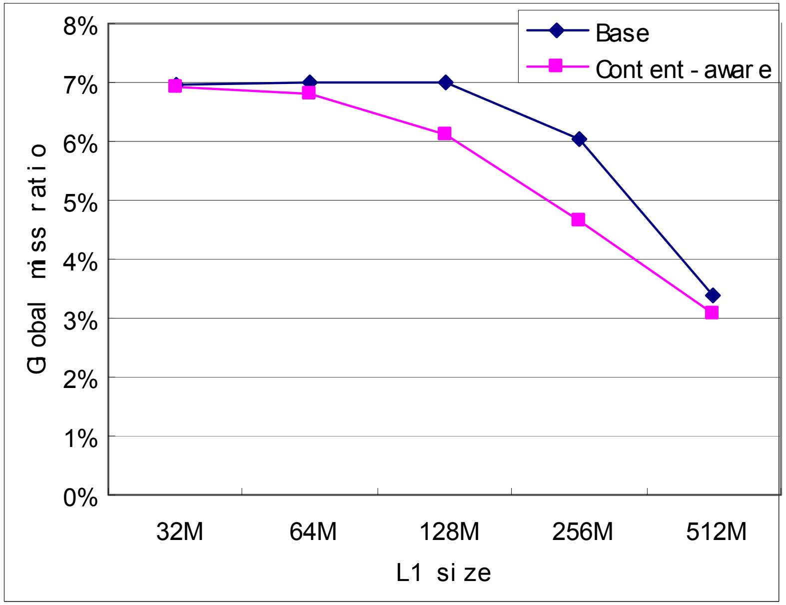


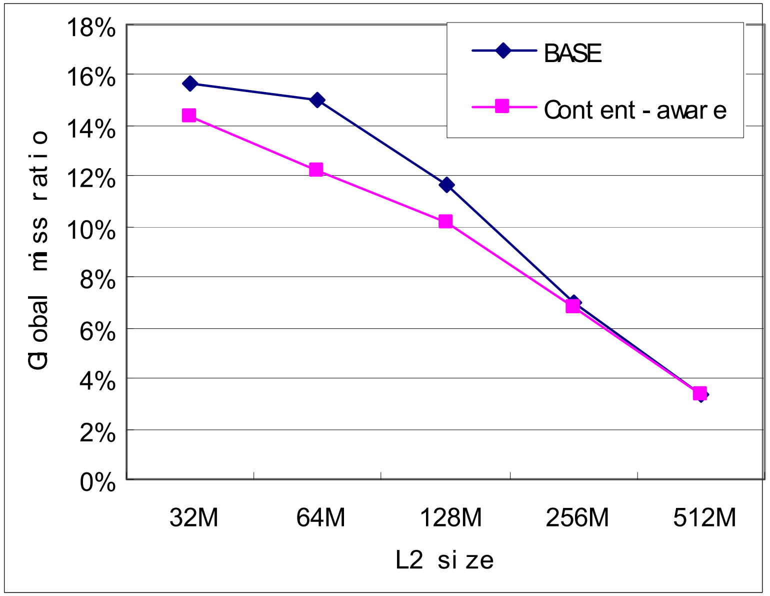
- Fixing storage cache (128MB) with various file system buffer size.
- LRU+LRU vs. LRUCA+LRU

- Fixing file system buffer size (256MB) with various storage cache size.
- LRU+LRU vs. LRUCA+LRU

Auspex filesystem trace.

Disk I/O reduction: TPC-C



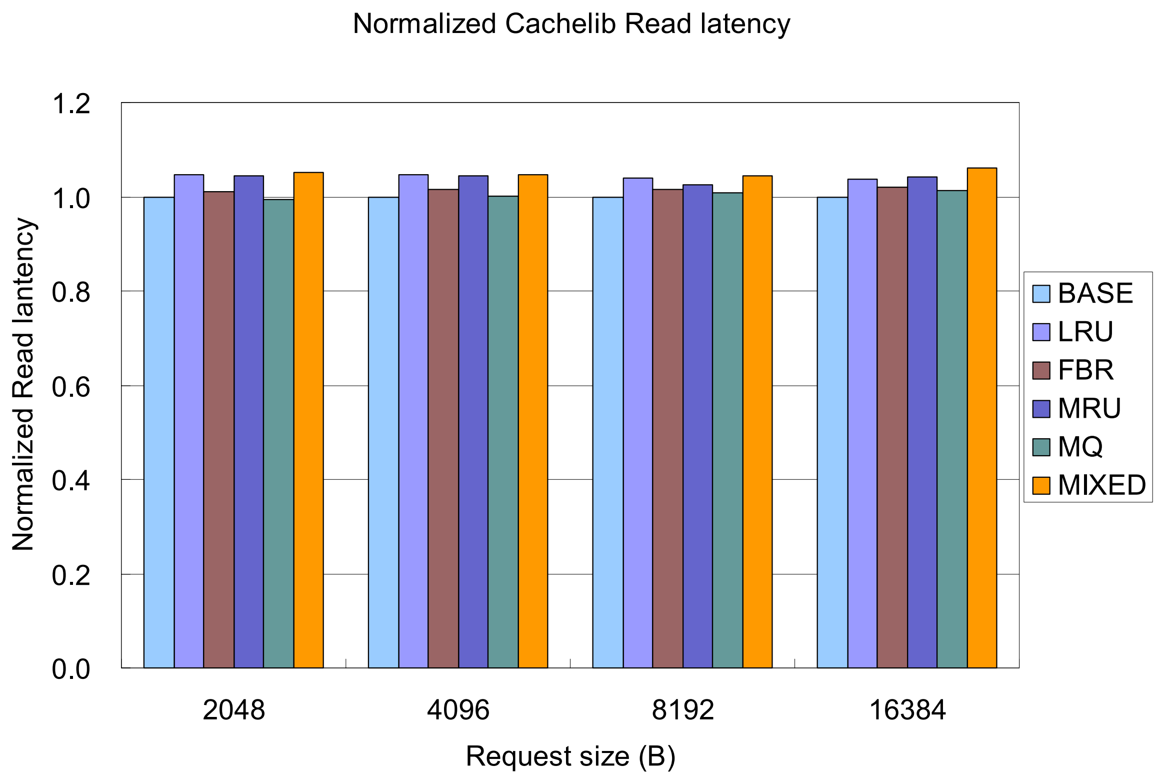


- Fixing storage cache (256MB) with various file system buffer size.
- LRU+LRU vs. LRUCA+LRU

- Fixing DB2 buffer size (64MB) with various storage cache size.
- LRU+LRU vs. LRUCA+LRU

DB2 TP-C buffer pool access trace simulation. 80 Warehouse, 2 hours.

Cachelib overhead



- *Cachelib* in a storage buffer cache: overhead < 5%

Content-aware caching

- Basic idea: buffer caches knows the content of the other buffer cache.
  - Generalize exclusive caching.

- Example: content-aware LRU (LRUCA)
  - Replacement prefers the block in both level buffer caches.
  - Applicable to other replacement algorithm (CLOCK).

Next access: d

Higher level: | ~~a~~ | **b**

Lower level: | **b** | **c**

LRU: evict a

LRUCA: evict b