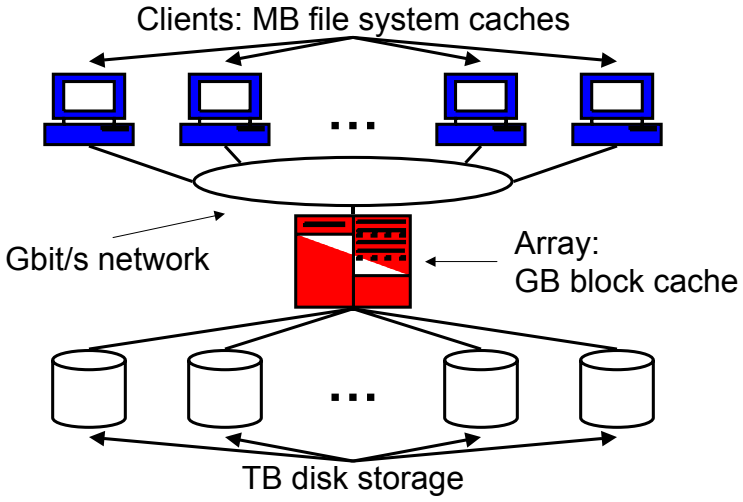
My cache or yours?
Making storage more exclusive

Theodore Wong (Carnegie Mellon University)
John Wilkes (HP Labs)

USENIX 2002

Carnegie Mellon
Parallel Data Laboratory

Typical storage system

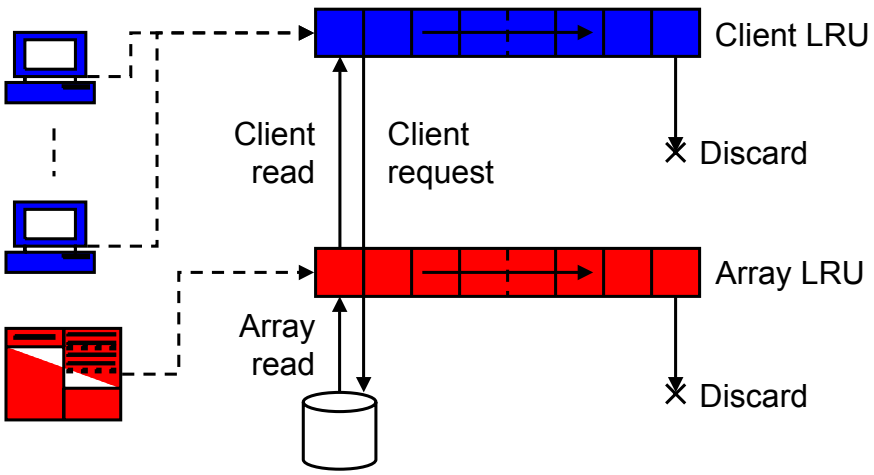


Motivation

> "Your cache ain't nuthin' but trash."
> *–Muntz and Honeyman*

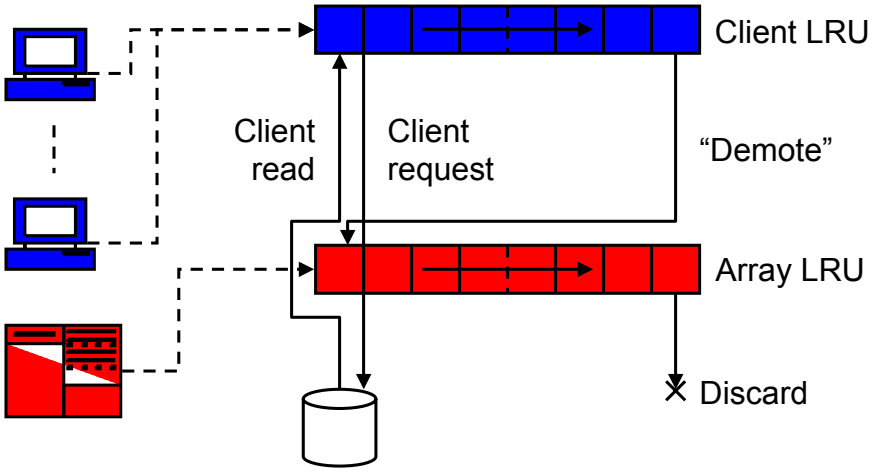
- Array cache is *inclusive*
  - Blocks duplicated in the client and array
- We make the array cache *exclusive*
  - Blocks either at the client or array

Typical inclusive caching



➢ Blocks cached in two places: wasteful!

Exclusive caching



➢Blocks cached in only one place

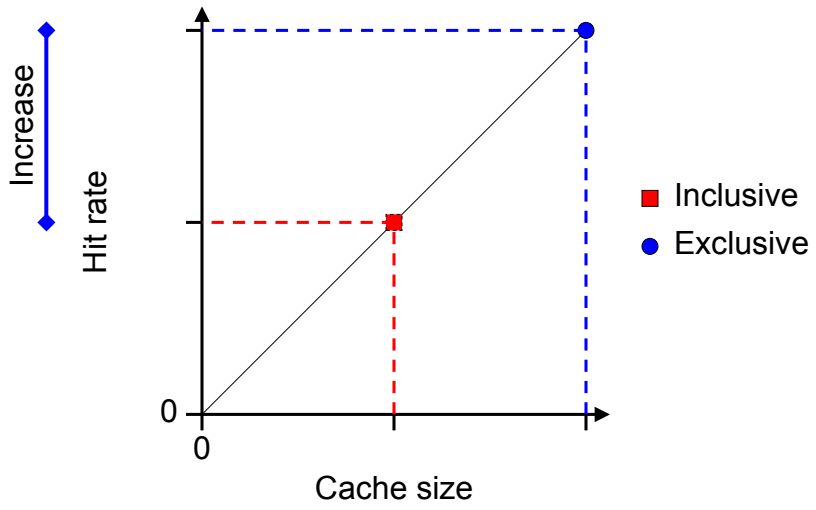
Outline

- Single-client evaluation
- Multi-client evaluation
- Conclusions

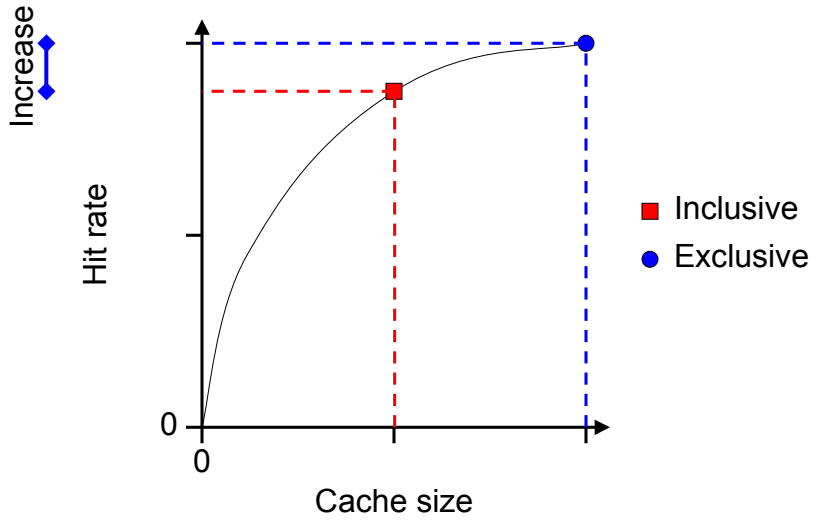
Single-client evaluation

- Verify that exclusive caching works
- Study caching schemes in simple systems:
  - Single client cache, equal in size to array cache
  - Read-only workloads
- Analyze sensitivity to:
  - Reduced bandwidth
  - Larger and smaller array cache sizes

Predicting benefits: Random



Predicting benefits: Zipf-like



Single-client workloads: Synthetic

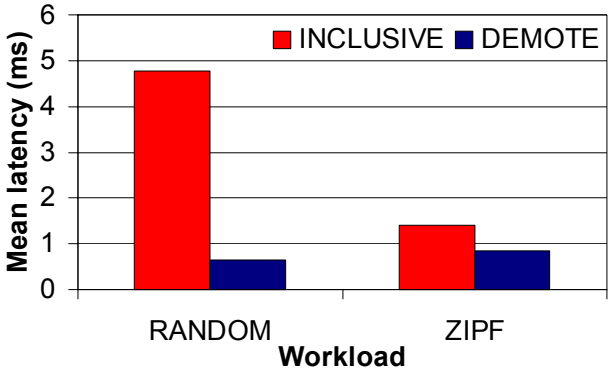
- RANDOM (e.g., transaction processing)
- ZIPF (e.g., web server; file server)

- Simulated in Pantheon

Single-client schemes

- INCLUSIVE
  - Baseline “typical” scheme
- DEMOTE
  - Exclusive caching scheme

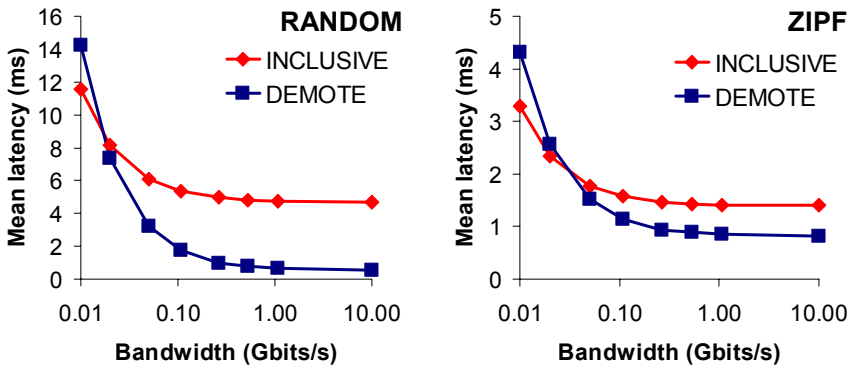
Single-client results: Synthetic



64 MB client cache
64 MB array cache
1 Gbit/s network
10 ms disk latency

➢ Exclusive caching works

Sensitivity evaluation results: Network



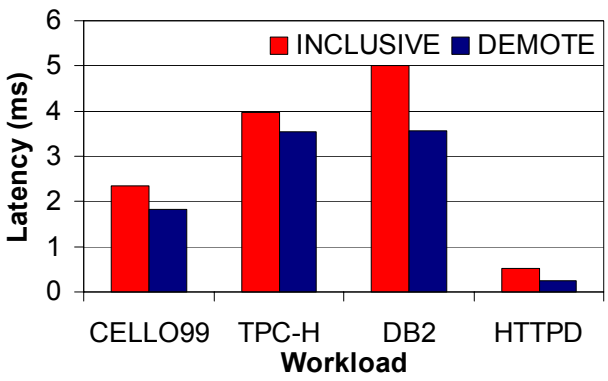
- Resilient to bandwidth variation

Single-client workloads: Real

- CELLO99: File server
- TPC-H: Database server benchmark
- DB2: Multi-client database workload
- HTTPD: Web server farm

- Simulated in fscachesim

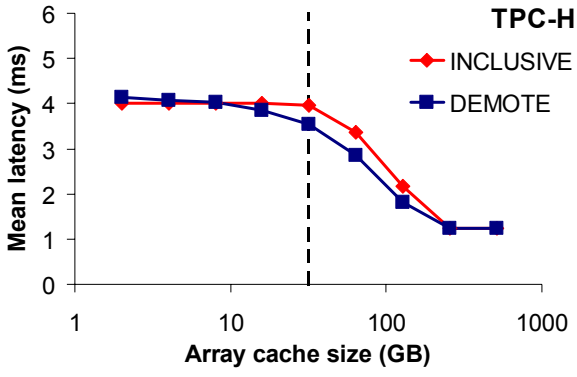
Single-client results: Real



1 Gbit/s network
5 ms disk latency

➢ Exclusive caching works for real loads

Sensitivity evaluation results: Cache size



32 GB client cache
1 Gbit/s network
5 ms disk latency

➢ Resilient to array cache size variation

Single-client summary

Workload	Speedup
RANDOM	7.5
ZIPF	1.7

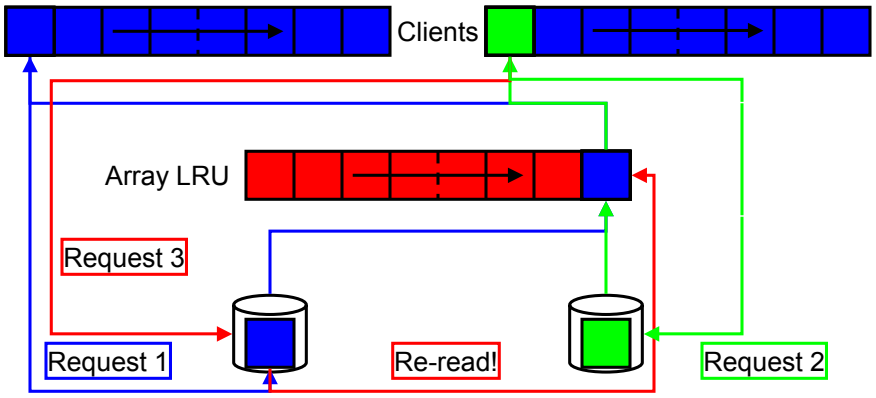
Workload	Speedup
CELLO99	1.3
TPC-H	1.1
DB2	1.4
HTTPD	2.2

- Exclusive caching yields significant speedups
- Resilient to bandwidth, cache size variation

Multi-client evaluation

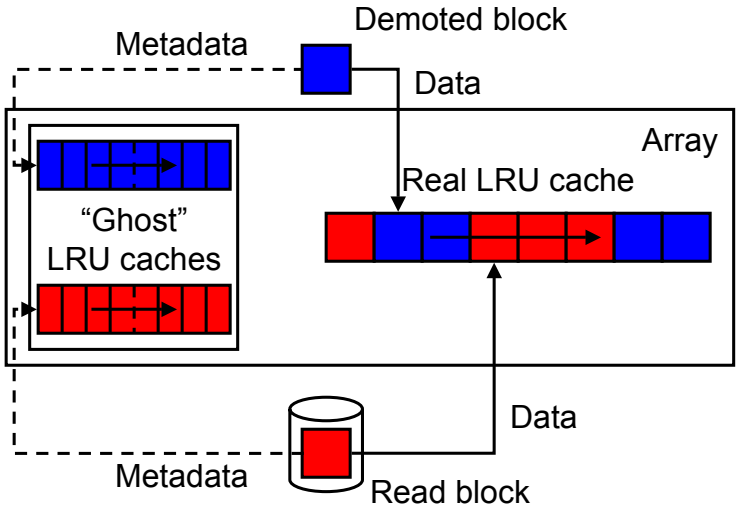
- Predict benefits for real systems
  - Large array cache, smaller client caches
- Consider effects of inter-client sharing
- Define two types of workload:
  - Disjoint workloads: No block sharing
  - Shared workloads: Some block sharing

Shared workloads and DEMOTE

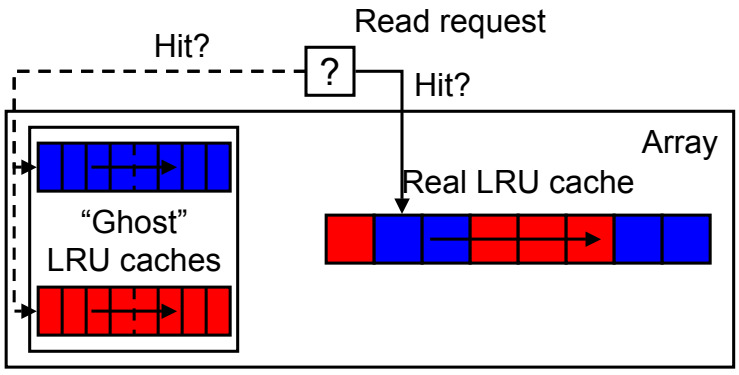


- Need to save "disk-read" blocks at array

Adaptive exclusive caching



Adaptive caching: Receiving requests

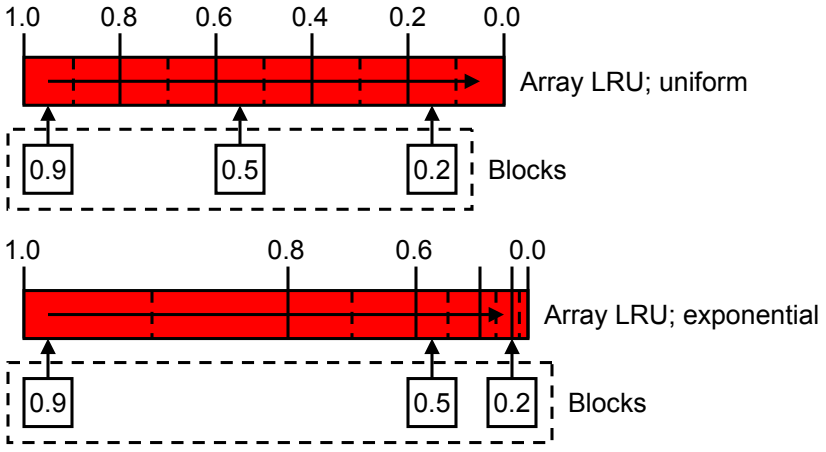


Insertions with scores

- Read block insertion:
  - Score = read ghost hits / sum of ghost hits
  - 0 → real cache head, 1 → real cache tail
- Demoted block insertion: similar to read

➢ Need to make insertions fast

Fast insertions with segments



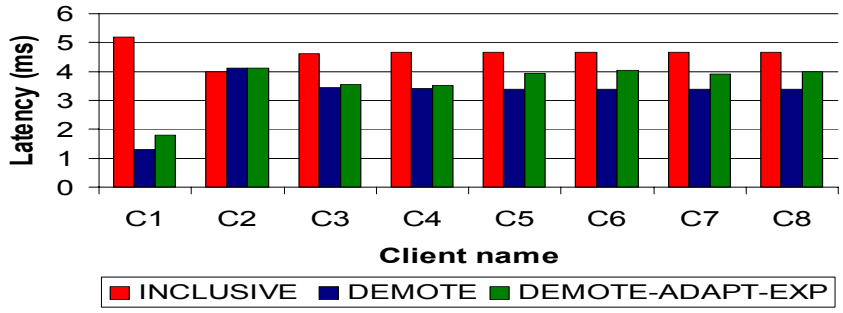
Multi-client workloads

- Disjoint:
  - DB2 (8 hosts): As before
  - OPENMAIL (6 hosts): Mail server farm
- Shared:
  - HTTPD (7 hosts): As before
- Simulated in fscachesim

Multi-client schemes

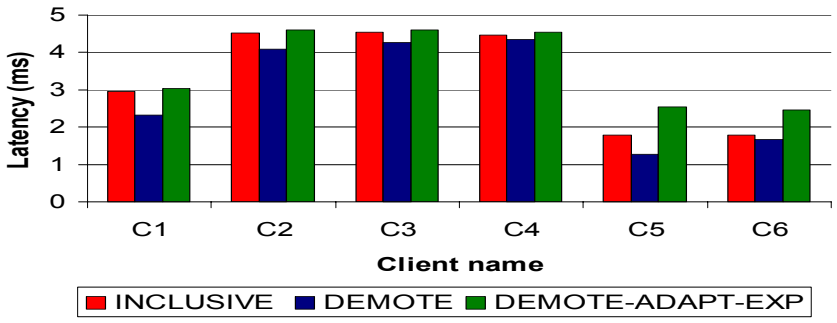
- INCLUSIVE
  - Baseline
- DEMOTE
- DEMOTE-ADAPT-EXP
  - Adaptive caching, exponential segments

Multi-client results: DB2



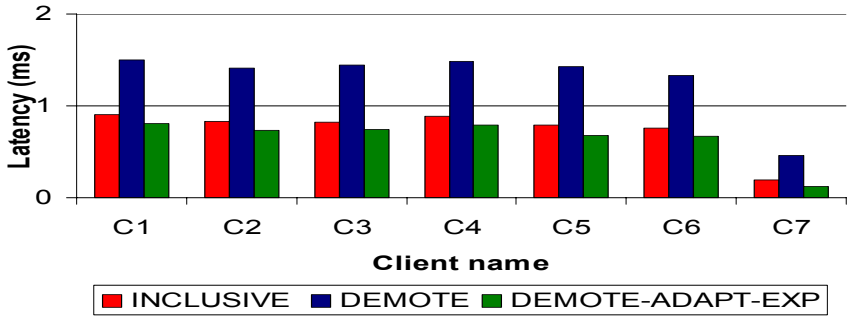
➢ DEMOTE for disjoint workloads:
- Keep demoted blocks
- Eject disk-read blocks

Multi-client results: OPENMAIL



- Again, DEMOTE for disjoint workloads:
  - Keep demoted blocks
  - Eject disk-read blocks

Multi-client results: HTTPD



- DEMOTE-ADAPT-EXP for shared workloads:
  - Keep both demoted and disk-read blocks

Multi-client summary

Workload	Mean per-client speedup	
	DEMOTE	DEM-ADAPT-EXP
DB2	**1.5**	1.3
OPENMAIL	**1.2**	0.9
HTTPD	0.6	**1.2**

- Eject read blocks for disjoint workloads (DB2, OPENMAIL)
- Keep some read blocks for shared workloads (HTTPD)

Related work

- Inclusive caching [Muntz1992, Froese1996]
- Global memory management:
  - Database system cache management [Franklin1992]
  - Peer-to-peer cooperative caching [Dahlin1994, Feeley1995]
- Per-workload cache management policies

Conclusions

> "My cache OR your cache?"
> –*Wong and Wilkes*

- Exclusive caching beats inclusive
- Simple demote op yields big speedups

More information

- My research page:
  - http://www.cs.cmu.edu/~tmwong/research/
- HPL Storage Systems Program:
  - http://www.hpl.hp.com/SSP/