Efficient Cooperative Caching using Hints

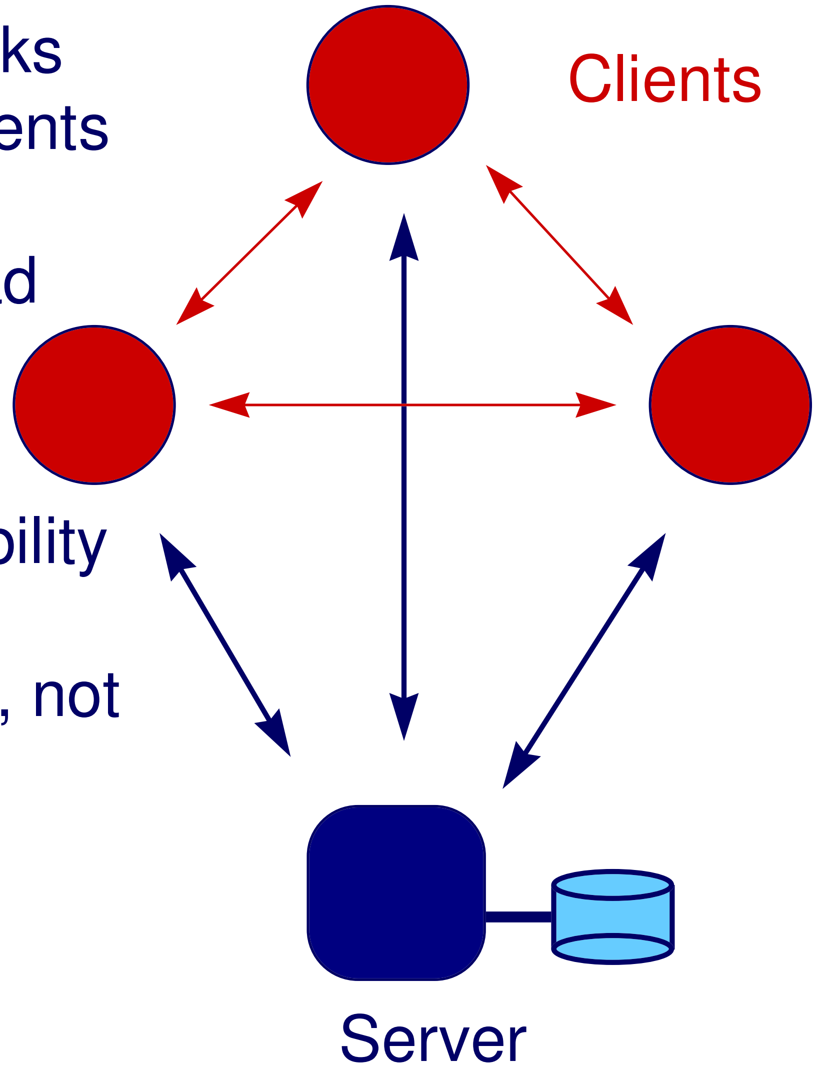
Prasenjit Sarkar

John H. Hartman

Computer Science Department
University of Arizona
{jhh,psarkar}@cs.arizona.edu

Cooperative Caching

- Clients access blocks cached on other clients
- Reduces server load
  - Improves performance
  - Increases scalability
- Cooperative clients, not coordinated
  - No high-level management

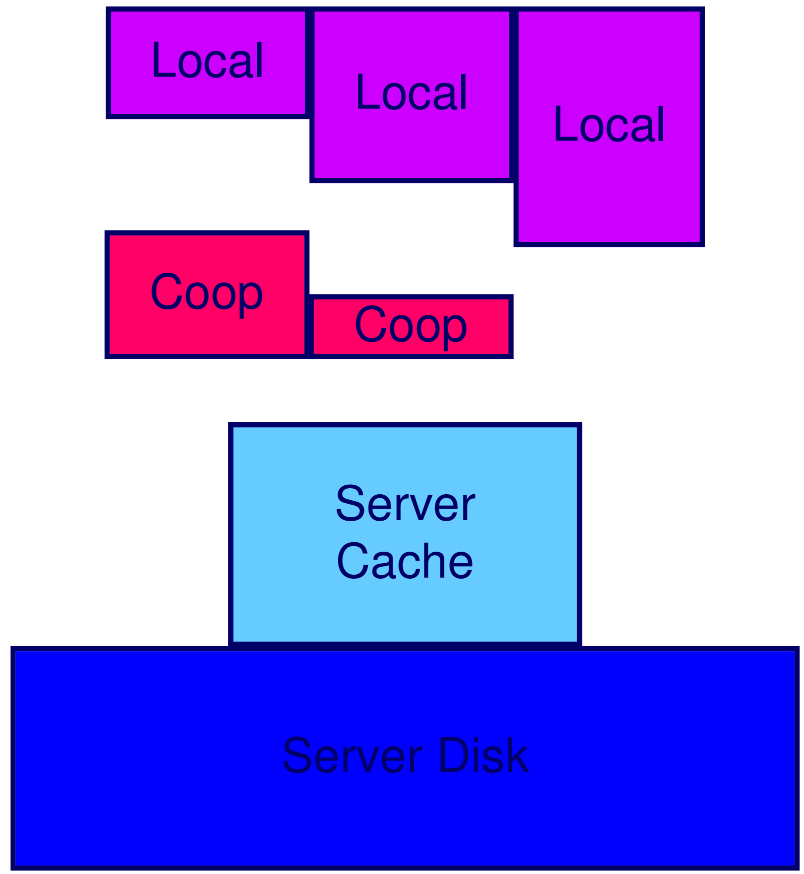


Outline

- Local vs. cooperative cache
- Using hints to manage the cache
- Simulation results
- Conclusions

Local vs. Cooperative Cache

- *Local cache*: blocks cached for local applications
- *Cooperative cache*: blocks cached for other clients
- Share client memory



Difficulties

- Finding blocks on other clients
  - Where to look?
- Cooperative cache contents
  - Which blocks are cached?
  - Where are they cached?
  - Which blocks are replaced?
- Size of cooperative cache
  - Affects local hit rates

Goals

- Good performance, low overhead
- Minimize average block access time
- Clients make local decisions when possible
    - Avoids bottlenecks
    - Reduces overhead
    - Improves scalability
- Simplicity

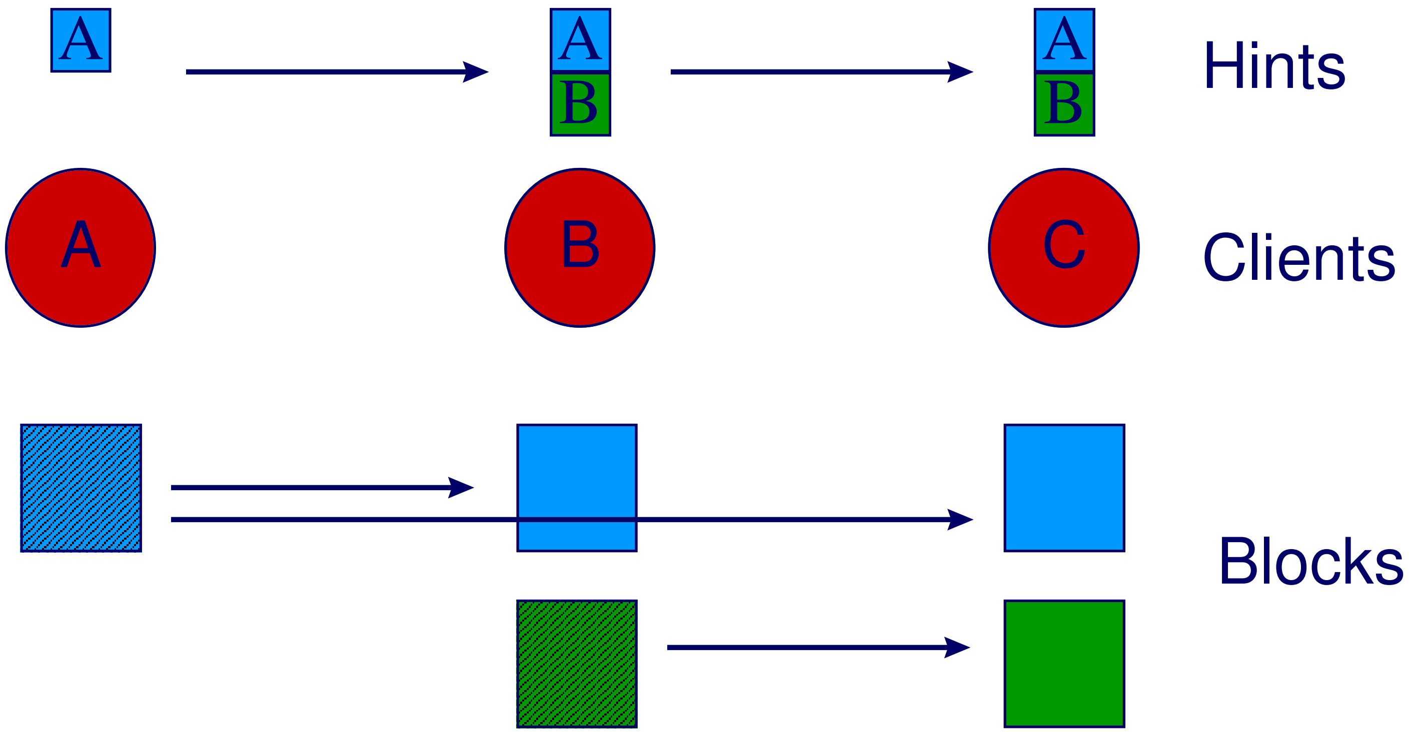
Hints

- Imprecise information about global state
  - Enable local decisions
  - Avoid centralized control
- Trade-offs:
  - \+ Hints can be easy to maintain
  - \+ Local decisions are easy to make
  - \- Poor hints can cause bad decisions and poor performance

Block Lookup

- Clients use block location hints to request blocks
- Out-of-date hints cause misdirected requests
  - Forwarded to another client if possible, to server otherwise
- File's hints obtained from last client to obtain them
  - Assumed to be most accurate
  - *Managers* keep track of last client to obtain hints

Simple Lookup Example



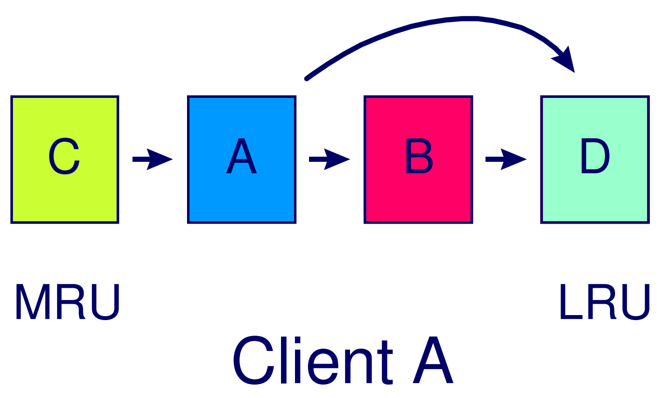
- Hints point to copy read from server

Updating Location Hints

- A client's location hint for a block is updated when:
  - using the hint causes a misdirected request
  - client reads the block from the server
  - the hint points to the client itself and the block is replaced
  - cache consistency invalidates the block

Best-Guess Replacement

- Approximate global LRU
  - Store LRU blocks in cooperative cache
- Each client has list of LRU block on every client
- Client replaces global LRU block with its own
  - Both clients involved update lists



Duplicate Avoidance

- Duplicate copies pollute cooperative cache
- *Master copy*: pre-designated copy to cache
  - Cooperative cache contains only master copies
  - Master copy obtained from server
  - Only a hint:
    - Multiple master copies of same block may occur if location hints are incorrect

Simulations

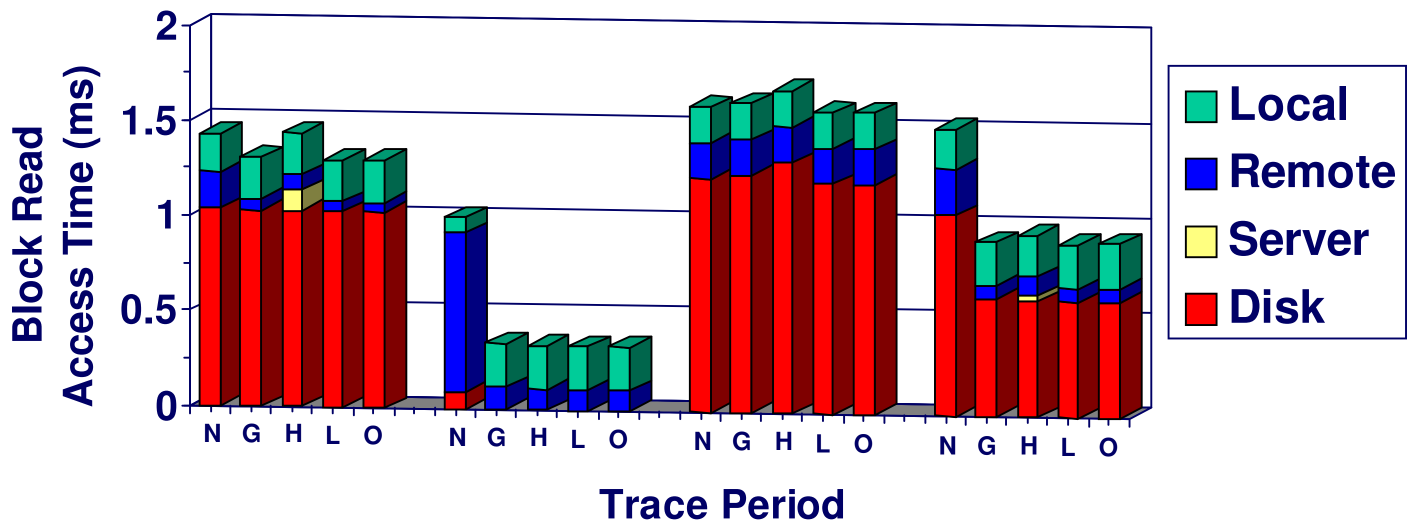
- Trace-driven simulation
  - Sprite file system (four two-day periods)
  - Unix workload

Clients	42	Block Size	8 KB
Servers	1	Message Latency	0.2 ms
Managers	1	Local Memory Latency	0.25 ms
Client Cache	16 MB	Remote Memory Latency	1.25 ms
Server Cache	128 MB	Disk Latency	15.85 ms

Other Algorithms

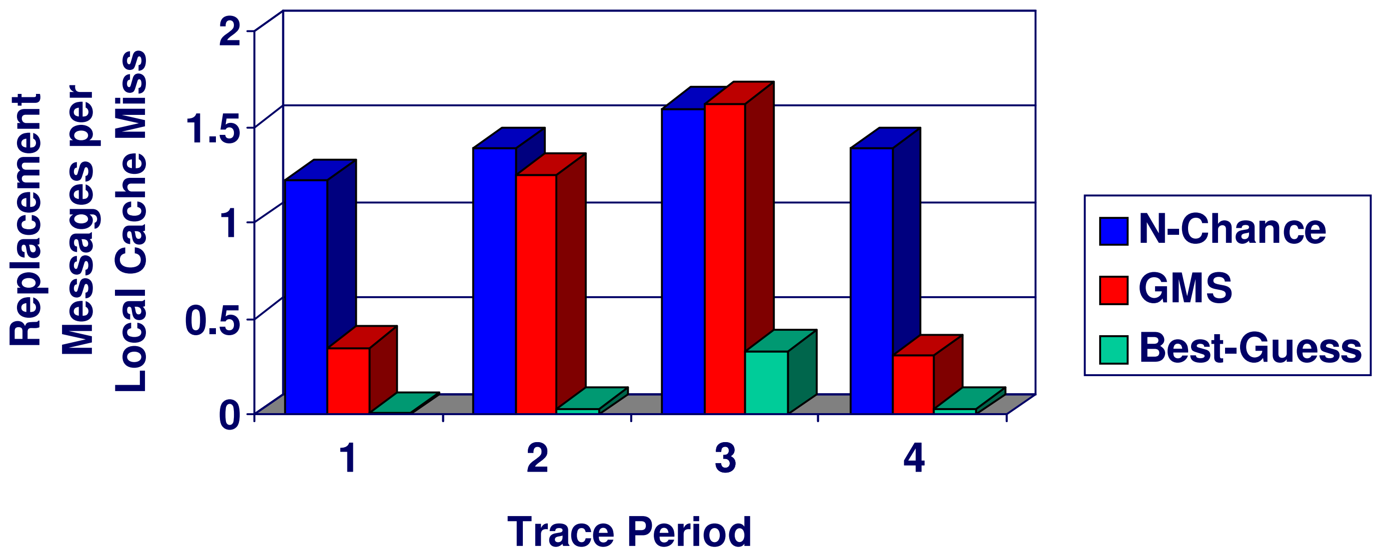
- N-Chance -- [Dahlin94]
  - Centralized control of lookup and avoidance
  - Random replacement
- **G**lobal **M**emory **S**ervice [Feeley95]
  - Centralized control of lookup, avoidance, and replacement
- Optimal
  - Replace block used farthest in the future
- Global LRU
  - Approximates optimal

Block Access Time



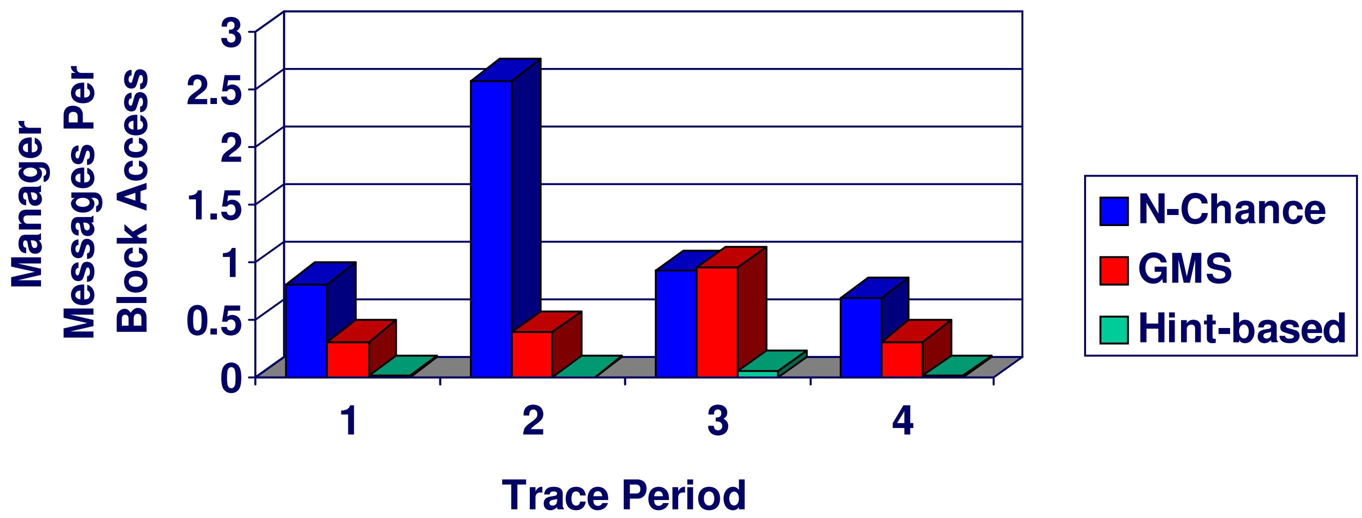
- Hint performance close to optimal
  - Hint accuracy > 99%

Replacement Traffic



- Hints reduce communication
  - Clients make local replacement decisions

Manager Load



- Hints reduce overall manager load
  - Clients make local decisions

Conclusions

- Hints provide good performance, low overhead
  - Near-optimal block access times
  - Replacement traffic reduced by a factor of 5
  - Manager load reduced by a factor of 15
- Future work
  - More workloads, long-term study
  - Asynchronous replacements
  - Forwarding storms
  - Security