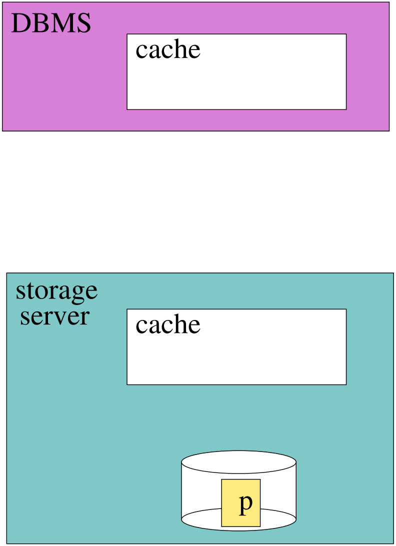
CLIC
CLient-Informed Caching for Storage Servers

Xin Liu Ashraf Aboulnaga Ken Salem Xuhui Li

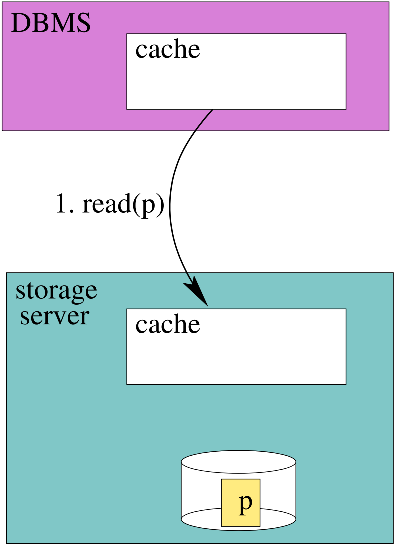
David R. Cheriton School of Computer Science
University of Waterloo

February 2009

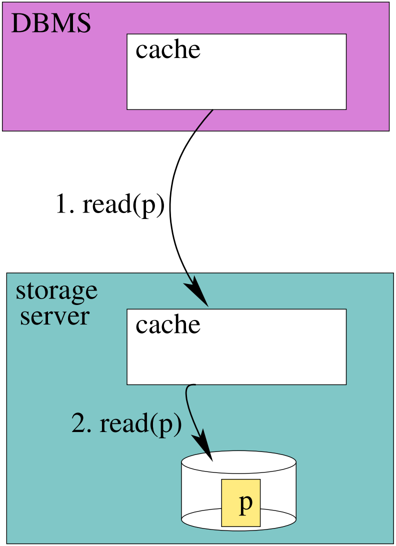
Two-Tier Caching



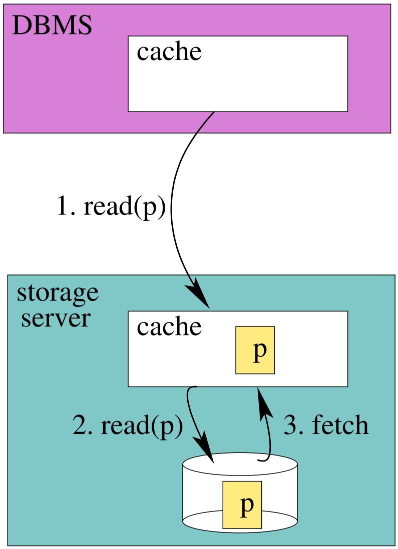
Two-Tier Caching



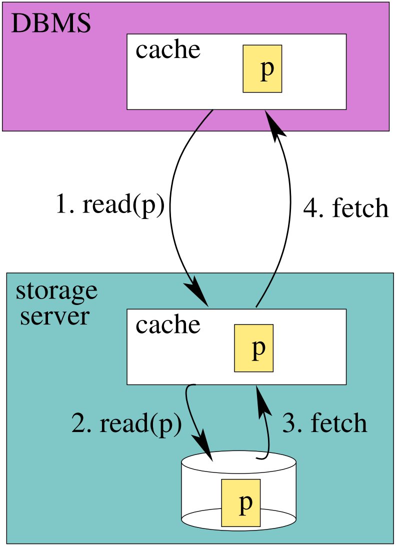
Two-Tier Caching



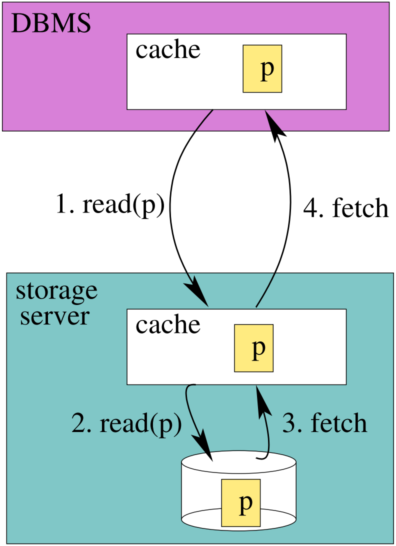
Two-Tier Caching



Two-Tier Caching



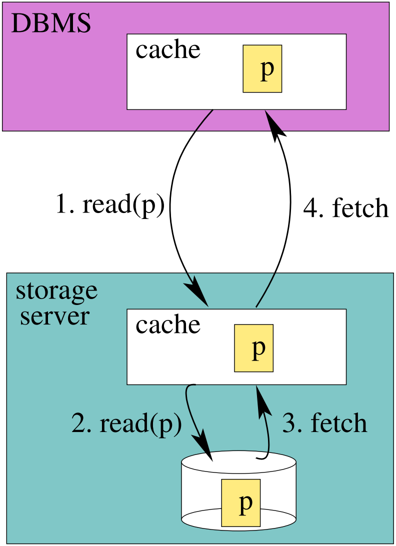
Two-Tier Caching



Problems:

- cache inclusion

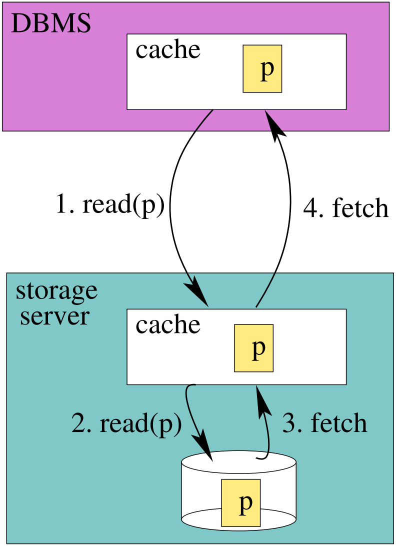
Two-Tier Caching



Problems:

- cache inclusion
- poor temporal locality

Two-Tier Caching



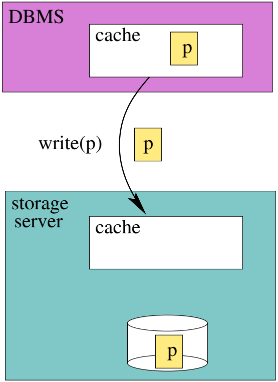
Problems:

- cache inclusion
- poor temporal locality

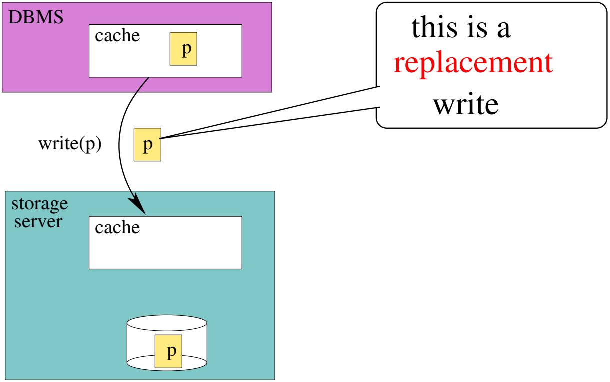
One Solution:

- hinting

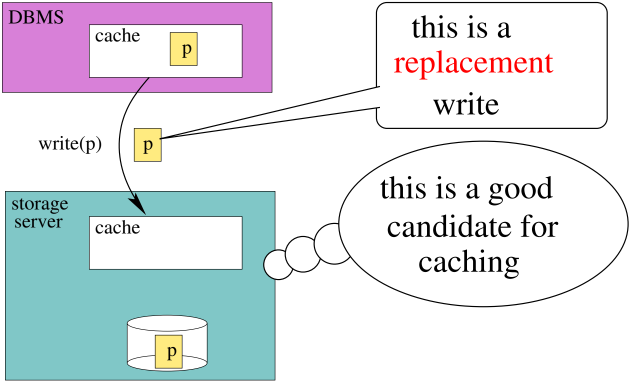
Example: Write Hints



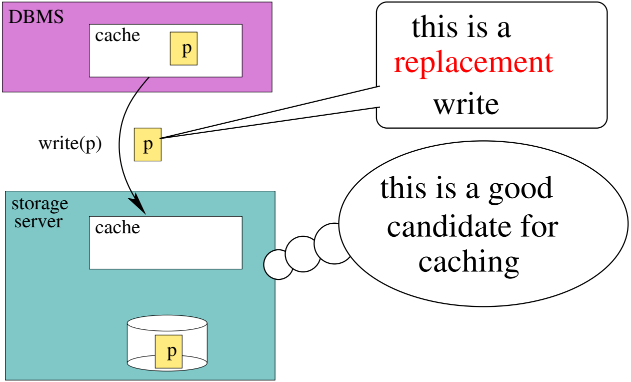
Example: Write Hints



Example: Write Hints



Example: Write Hints



The storage server can use TQ, an ad hoc hint-aware replacement policy, to exploit write hints.

Problems with Ad Hoc Hint-Aware Policies

narrowness: new hints? multiple hints?

Problems with Ad Hoc Hint-Aware Policies

narrowness: new hints? multiple hints?

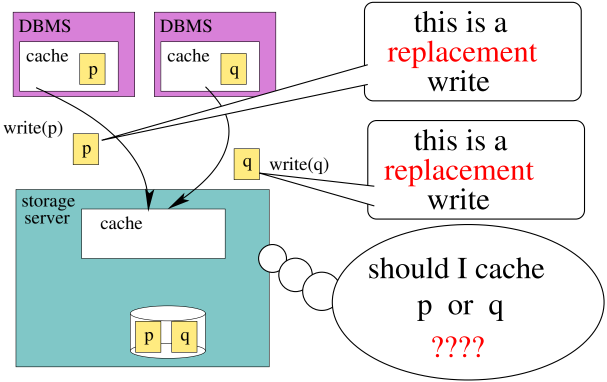
brittleness: correct response to hints?

Problems with Ad Hoc Hint-Aware Policies

narrowness: new hints? multiple hints?

brittleness: correct response to hints?

single source: multiple hint generators?



The CLIC Approach

- a hint-aware caching policy for 2nd-tier caches

The CLIC Approach

- a hint-aware caching policy for 2nd-tier caches
- no hard coded response to specific hints

The CLIC Approach

- a hint-aware caching policy for 2nd-tier caches
- no hard coded response to specific hints
- instead, learn which hints signal good caching opportunities

The CLIC Approach

- a hint-aware caching policy for 2nd-tier caches
- no hard coded response to specific hints
- instead, learn which hints signal good caching opportunities
- benefits:
  - handles multiple hint types
  - handles new hint types
  - handles hints from multiple clients by treating each client's hints as distinct

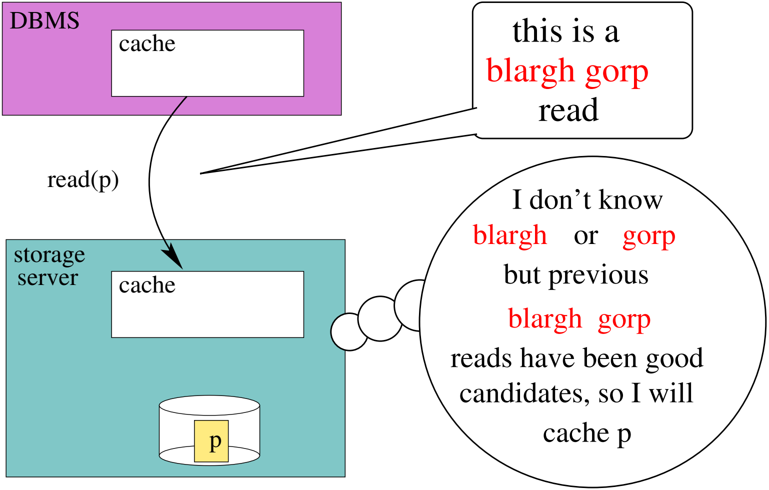
The CLIC Approach

- a hint-aware caching policy for 2nd-tier caches
- no hard coded response to specific hints
- instead, learn which hints signal good caching opportunities
- benefits:
  - handles multiple hint types
  - handles new hint types
  - handles hints from multiple clients by treating each client's hints as distinct

CLIC Hints

CLIC separates the generation of hints (done by the storage clients) from the interpretation of those hints for caching purposes (done by the storage server).

CLIC Illustrated



Generating Hints

- Storage client must be modified to generate one or more types of hints.
- Storage clients attach a hint set to each read or write request. A hint set includes one hint of each type generated by the client.
- A storage client may choose to generate any types of hints that might be of use to the storage server.

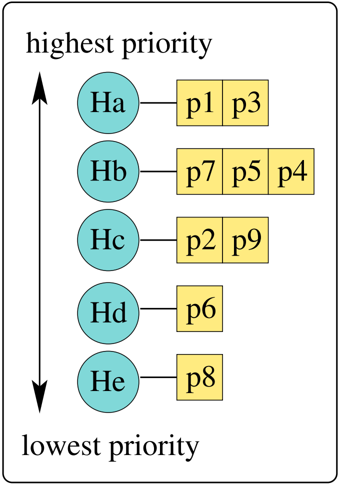
Generating Hints

- Storage client must be modified to generate one or more types of hints.
- Storage clients attach a hint set to each read or write request. A hint set includes one hint of each type generated by the client.
- A storage client may choose to generate any types of hints that might be of use to the storage server.

Example: Hints from DB2

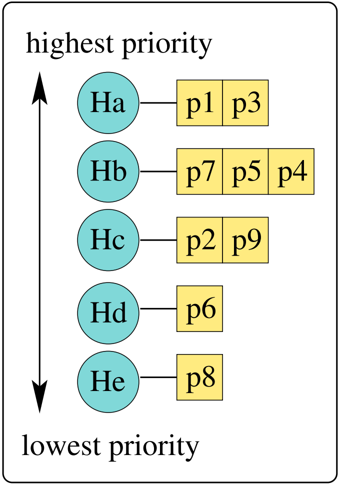
- buffer pool ID
- object ID: identifies a group of related DB objects
- object type ID: distinguishes table from index
- request type: read, replacement/recovery write
- DB2 buffer priority

A CLIC-Managed Cache



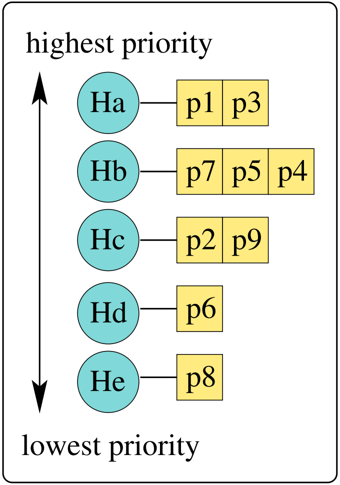
- each page is associated with the hint set which it was most-recently read or written

A CLIC-Managed Cache



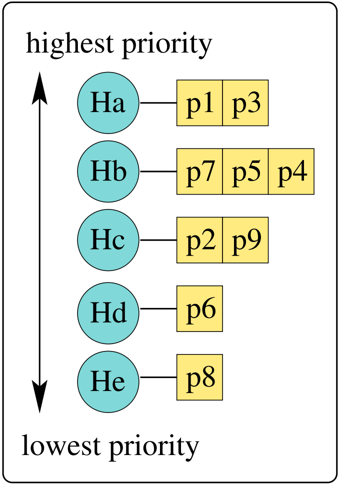
- each page is associated with the hint set which it was most-recently read or written
- each hint set has a priority

A CLIC-Managed Cache



- each page is associated with the hint set which it was most-recently read or written
- each hint set has a priority
- CLIC evicts pages associated with the lowest-priority hint sets

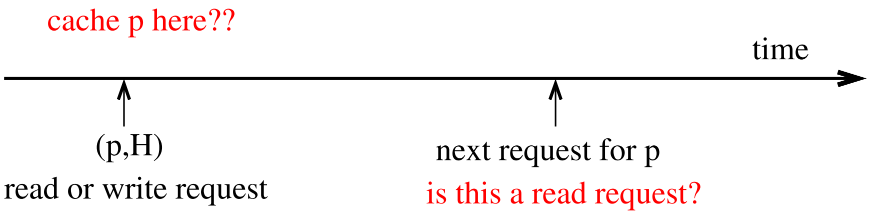
A CLIC-Managed Cache



- each page is associated with the hint set which it was most-recently read or written
- each hint set has a priority
- CLIC evicts pages associated with the lowest-priority hint sets
- CLIC chooses hint set priorities using a simple cost/benefit analysis

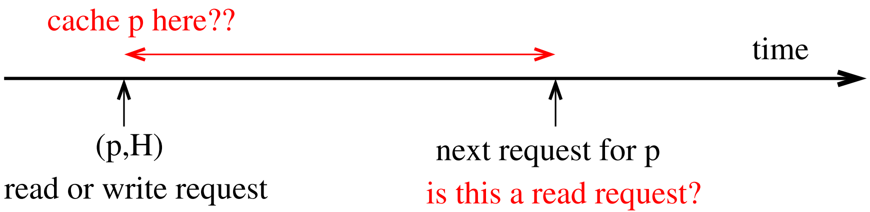
Cost/Benefit Analysis

Cost/Benefit Analysis



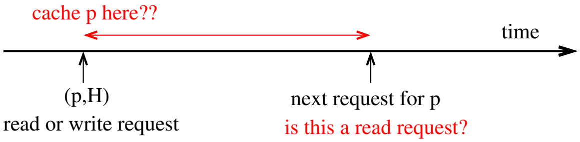
- there is a benefit to caching if the next request for *p* is a read request

Cost/Benefit Analysis



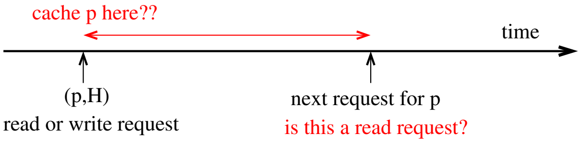
- there is a benefit to caching if the next request for *p* is a read request
- the cost of obtaining this benefit is that *p* must remain cached until the read request

Assigning Priorities to Hint Sets



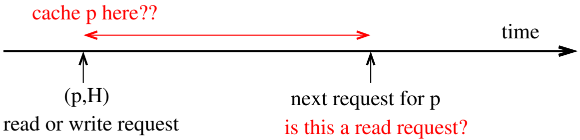
- when request (p, H) occurs, CLIC cannot know the the cost and benefit of caching p

Assigning Priorities to Hint Sets



- when request (p, H) occurs, CLIC cannot know the the cost and benefit of caching p
- instead CLIC estimates the cost and benefit of caching p at (p, H) based on previous requests with hint set H

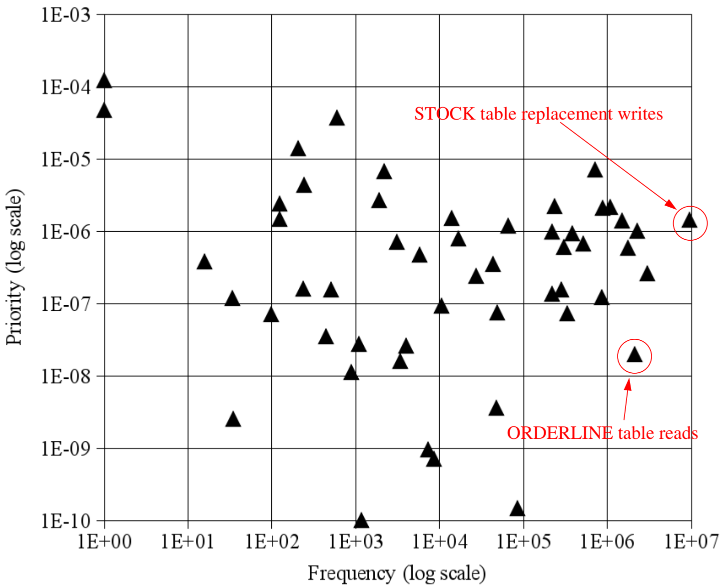
Assigning Priorities to Hint Sets



- when request (p, H) occurs, CLIC cannot know the the cost and benefit of caching p
- instead CLIC estimates the cost and benefit of caching p at (p, H) based on previous requests with hint set H
- CLIC assigns a priority to each hint set based on the cost and benefit of previous requests with hint set H

$$\text{Priority}(H) = \frac{\text{Read\_Hit\_Rate}(H)}{\text{Mean\_Time\_Until\_Read\_Hit}(H)}$$

DB2 Hint Analysis Example



Efficient Hint Analysis

- To analyze the cost and benefit of hint sets, CLIC must
  - track the most recent request and hint set for each page
  - track the mean read hit rate and read hit distance for each hint set

Efficient Hint Analysis

- To analyze the cost and benefit of hint sets, CLIC must
  - track the most recent request and hint set for each page
  - track the mean read hit rate and read hit distance for each hint set
- To reduce space requirements, CLIC
  - tracks the most recent request only for cached pages and a fixed number of additional, uncached paged

Efficient Hint Analysis

- To analyze the cost and benefit of hint sets, CLIC must
  - track the most recent request and hint set for each page
  - track the mean read hit rate and read hit distance for each hint set
- To reduce space requirements, CLIC
  - tracks the most recent request only for cached pages and a fixed number of additional, uncached paged
  - tracks read hit statistics only for frequently occurring hint sets

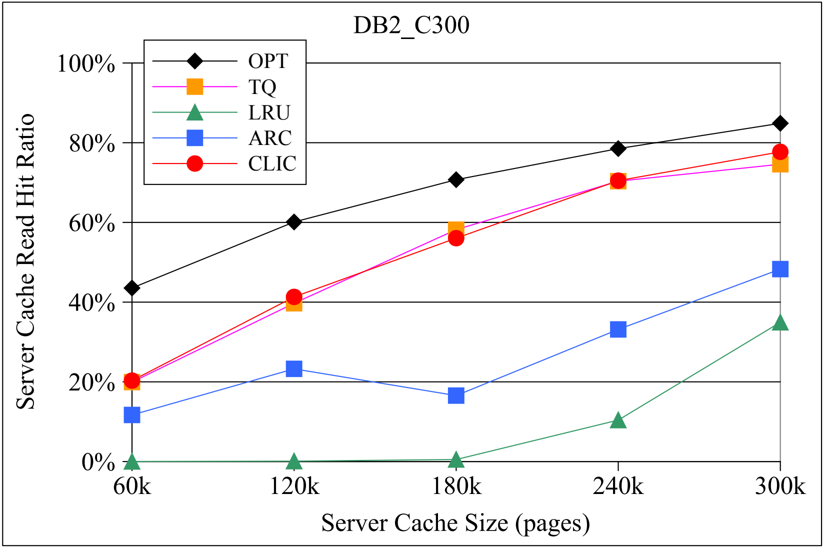
Efficient Hint Analysis

- To analyze the cost and benefit of hint sets, CLIC must
  - track the most recent request and hint set for each page
  - track the mean read hit rate and read hit distance for each hint set
- To reduce space requirements, CLIC
  - tracks the most recent request only for cached pages and a fixed number of additional, uncached paged
  - tracks read hit statistics only for frequently occurring hint sets
- We have also investigated the use of generalization to reduce the number of distinct hint sets.

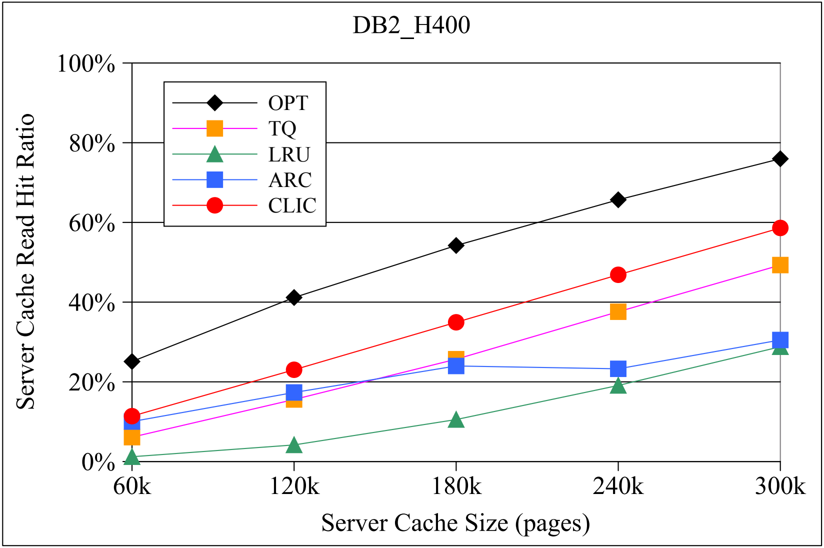
Performance

- we have used trace-driven simulation of the storage server buffer cache to compare CLIC to other replacement policies
- methodology
  1. modify DB2 and MySQL to generate hints and produce I/O traces
  2. run TPC-C (on-line transaction processing) and TPC-H (decision support) workloads on the database systems and collect I/O traces
  3. feed the traces to a simulation of second-tier cache, which implements CLIC, LRU, ARC, TQ and OPT
  4. measure the hit ratio achieved by different policies.

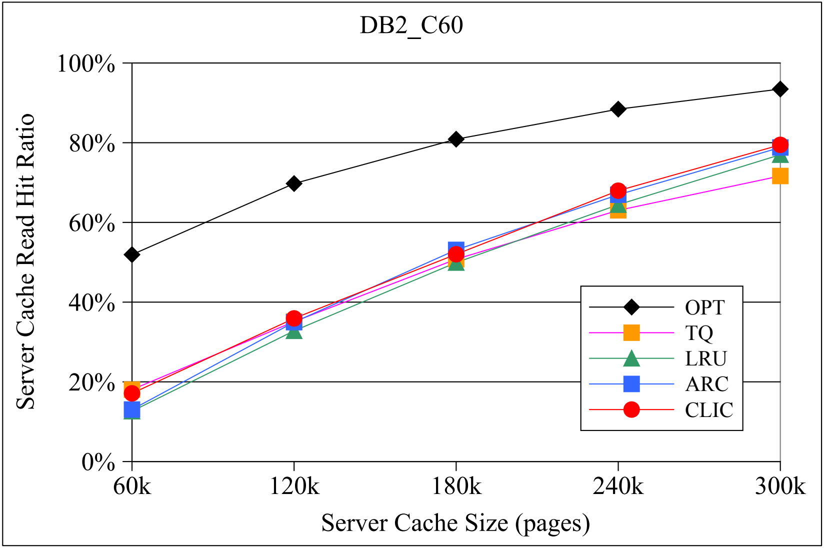
DB2 TPC-C - Medium DB2 Buffer Cache



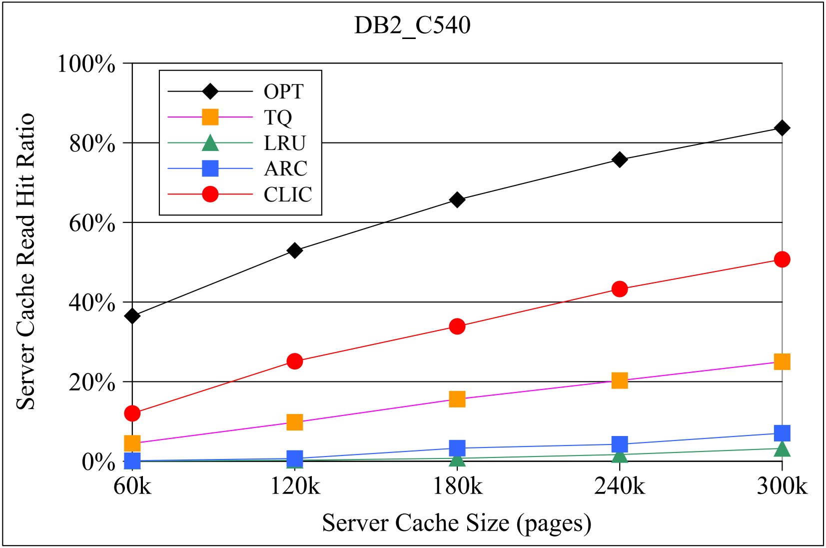
DB2 TPC-H - Medium DB2 Buffer Cache



DB2 TPC-C - Small DB2 Buffer Cache



DB2 TPC-C - Large DB2 Buffer Cache



Summary and Conclusions

- CLIC learns to identify I/O hints that signal good caching opportunities by tracking the request stream observed by the second-tier cache
- Because CLIC's responses to specific hints are not predefined, it naturally accommodates new hint types and hints from multiple storage clients.
- for our traces:
  - CLIC's performance usually dominates ARC's and LRU's, sometimes by a factor of 2 or more.
  - CLIC dominates the ad hoc, hint-aware TQ algorithm
  - CLIC's space overhead can be kept low (1% of storage server cache size in our experiments)