Caching Strategies for Data-Intensive Web Sites: A Critique

K. Yagoub, D.Florescu, V. Issarny, P. Valdez

Presented by:

Aziz Kara

University of Waterloo

Outline

- Overview of problems with the paper
- Some suggested improvements
- General comments about paper
- Distributed Discussion

Assumptions

- Everyone has read the paper…

Problems with the paper

- Authors don't address bandwidth used within site infrastructure, or more importantly, bandwidth used to send page to user
- All content resides in a database. Many sites are not of this format, but are still data intensive
- Authors concentrate on response-time and give little discussion about system throughput

Problems with the paper (cont'd)

- Lack of discussion about read v.s. write data
  - Authors only treat the case where a site has read-only data interactions
- No use of statistical information to automatically enhance run-time policies based on access patterns
  - Although authors do mention this as an ultimate goal of the exercise

Problems with the paper (cont'd)

- No distribution in their solution
  - Scheduler in their solution is not distributed therefore can become a bottleneck and a single point of failure in system
    - Especially if scheduler has to repeatedly process run-time policies
  - Cache managers are single-point of failure as well as potential bottlenecks
    - Only link to the repositories, therefore if managers fail, access to repositories is gone.

Problems with the paper (cont'd)

- How flexible to change are the declarative specification and run-time policies?
  - With respect to update mechanisms
    - Partially covered
  - Migration details
    - Not covered
- Does the declarative site specification paradigm restrict the use of dynamic-layouts based sites?
    - Not covered

Problems with the paper (cont'd)

- DB caching is done in DB
  - What are we really saving here?
    - Query processing time?
    - Query optimization time?
  - Still required to fetch results from DB - means crossing client/server boundary
    - Messaging overheads if distributed setup

Suggested Improvements

- Caching updates are either push or pull, use lazy replacement strategy
  - Authors' experiments show that active update mechanisms contribute non-trivial costs to response times
  - Means first request for object takes a little longer to process, but subsequent requests are processed faster
  - Refrain from caching objects that are requested infrequently

Suggested Improvements (cont'd)

- DB caching can sometimes be done in memory
  - e.g. Delayed stock quotes are good for 5 min. Store query results in application server memory so that we don't have to keep going to DB
- Caching static pages with dynamic references - can be thought of as holes to be filled in on HTML page
  - Only have to evaluate non-cached dynamic components and assemble page at the HTML generator

Suggested Improvements (cont'd)

- Process requests for static pages in a different manner that doesn't require front-end processing tools described in solution
  - Useful if scheduler repeatedly must process run-time policy
  - Useful for pages close to root of site
  - Increased load handling capabilities
    - e.g. Sept. 11th, 2001

General Comments

- Authors don't stress importance of site infrastructure design
  - If HTTP requests are served from same machine as XML/HTML generation, then could have a bottleneck problem
- Authors don't survey sites to see how many are of the form they discuss
  - I.e. lack of motivation for problem
  - Whats the point of all this caching if your site falls in this category, but you don't get high traffic levels?

General Comments (cont'd)

- Authors lack discussion of implementation details
  - No discussion of precisely what happens when an HTTP request is received
    - Does scheduler have data structure representation of the run-time policy?
  - Should XML/HTML repositories be physical storage or main-memory storage? Advantages? Disadvantages?

Distributed Discussion Time

Thank you for listening patiently.