Intelligent Internet Caching and Pre-fetching

Krishna Prasad C
pkrishna@cse.iitb.ernet.in
Ajay Dhawale
ajay@cse.iitb.ernet.in

Outline

- Why Pre-fetch?
- HTTP support for Caching
- Properties of Web Cache
- Prefetching Methods
- Future Work

Why Pre-fetch?

- Demand Driven caching, under utilizes the cache space, network Bandwidth and does not reduce the user latency.
- Expiry of document on cache.
- User access pattern is predictable.
  - Online Prefetching
  - Offline Prefetching
- Semantic Caching.

Properties of Web Cache

- Non homogeneous Size.
- The Cost of downloading a document is neither linear nor uniform.
- The Web Protocols support only whole file transfers.
- Locality of reference.

Cache Coherence

- Traditional Coherence Methods
  - Validation Check with version - as in NFS.
  - Callback - notification on change.
- HTTP Support for Cache coherence
  - **Get** does not guarantee that it will return a fresh page.
  - Conditional get - *If-Modified-Since:date.*
  - *Pragma:no-cache* - Reload!!
  - *Last-Modified:* in response header
  - *Date:* in response - Last known to be good.
  - *Expires:date* expected date of expiry.

Pre-fetching to keep Cache coherent

Claim : We can pre-fetch all the active documents known to change after the expiry date.
Support : Studies on user access patterns show that users tend to return to documents accessed by them in near past at regular interval [multiple of 24 hours] .
Issues

- expiry date does not guarantee that the document has changed.
- The document on expiry should be pre-fetched only if there is a high probability of it being accessed in near future.
- Many document on web do not have *Expiry:Date* associated with them, but do change regularly.

Pre-fetching Issues...

- Advertisements.
- mirror sites.

Solution : *Shingling.*
To compute the *resemblance* and/or the *containment* of two documents it suffices to keep for each document a *sketch* of a few hundred bytes.

Replacement Policy

- LRU is as acceptable replacement policy for Web Caching.
- Basic LRU has to be modified for the web; Non-uniform size and non-uniform replacement cost has to be considered.
- Cache supporting pre-fetching can be assumed to be full to start with.
- *Admission control :* A new document must have some minimum criteria to purge already existing documents in the cache and gain entry.

LRU - Mathematical Model

$$Minimize \sum_{i \in C(k)} c_i . \frac{y_i}{\Delta T_{ik}} \tag{1}$$

subject to:

$$\sum_{i \in C(k)} S_i . y_i >= R \tag{2}$$

$$y_i \in \{0, 1\} \tag{3}$$

This is NP-Hard problem . Knapsack.

LRU - Approx Algorithm

- Sort in non decreasing order based on $\frac{\Delta T_{ik}.S_i}{C_i}$
- Purge highest index objects - greedy selection
- Issue - Difficult to implement as the order depends on the units of the each quantity measured.

Support : Popular documents are small (256-768 bytes).
Small Documents ($\tilde{4}$k) have higher density in cache.

LRU with Pre-fetching

- LRU designed for demand driven caching. Each document on cache must have properties like *time of last reference*.
- Replacing older document with a fresher one - retain the older document's time of last reference.
- Pre-fetching an entirely new document - Have to assign a *last reference time*
  - last reference time $= \infty$ - document does not gain admission.
  - last reference time $= 0$ - document gains preference over other documents.

Pre-Fetching Methods

At any given point of time, given a *condition* and a document d, it can be predicted that the document d would be accessed with probability p and with confidence c.

Methods

- Deterministic - Data Mining - Association Rules, clustering...
- Stochastic - Based on Markov's Chain, Fractal models...

Web Mining - Association Rules

- $R \rightarrow S$, where R and S are ordered sets of documents.
- Each association rule has *support* and *confidence*.
- any user assessing documents in set R, is bound to access documents in S with a probability given by the *confidence* and *support* gives the fraction of all user session in which $R \cup S$ occurs.

1. L1 = large 1_itemsets;
2. for (k = 2; $L_{k-1} \neq \emptyset$; k++) do **begin**
3. c_k = apriori_gen(L_{k-1});
4. get_count(C_k);
5. $L_k = \{c \in C_k$ / c.count $\geq$ minsup$\}$
6. **end**
7. Answer = $\cup L_k$

Stochastic Methods - Issues

State driven. State table has an entry for probability , confidence of accessing a document. Each document accessed changes the state.

- Most of these methods are *online* methods.
- Heavy book keeping needed to maintain the large state tables. Not scalable.
- The set of documents in the system needs to be finite and constant.
- Destination set is every document on web.
- These methods are generally well suited for pre-fetching documents from a slow read only tertiary device like CD-ROM into the secondary / primary memory or a HTTP accelerator.

Work to be Done...

Only offline prefetching is suited for Web.

1. What is the universe of the documents on the web?
   We would need to study what percentage of documents that are referenced are (1) in the cache and (2) one link away from the documents on the cache.

2. How do you keep cache coherent through pre-fetching?
   We need to mine the logs of proxy cache to find out a set of documents that are changing regularly and note when was the change observed (this is because the change in document would be observed only when a user accesses it). This can be done using the size of the document and flag TCP_REF_FAIL_HIT (which means the hit was a stale one) in the logs.

3. What is the *admission control* for objects being pre-fetched into the cache?

4. How to break up the user sessions in the logs into logical transactions?

5. Semantic Caching

   (a) What fraction of user patterns start from a query?

   (b) What fraction of users follow the links in the given document?

   (c) Does *Keywords* in the document give the semantic of the document?