An Active Architecture for User Profile based Dynamic Web Caching

Static caching is a well-researched area while dynamic caching has proved to be a hard nut to crack. Personalization of content has become the way of the web today with almost all major service portals like MSN, CNN, Lycos, Yahoo, Amazon etc implementing this to a certain extent. For example I can create a user profile on MSN subscribing to news, magazines and other information of my interest. For example I maybe interested in Sports, Business and the International Headlines. Also the ads that I see on these web pages are tailored towards my user profile. Caching this kind of content is hard and the server has to basically execute the script for every user query placing a great deal of burden on it. But, dynamic personalized content, which is critical to the personalized infrastructure of today, does not yet fit into the web-caching model. We think that at least a certain proportion of this personalized user content can be both generated and cached on the proxy itself thus reducing the burden on the web server itself to a certain extent.

We would like to propose a caching architecture to support caching and generation of certain course gained personalized user content on the proxy itself and supporting protocols between proxy caches and cache-server to enable this. We hope to through this architecture propose a solution to reduce to certain extent the large penalty for personalized processing of dynamic content on the server side.

We through our architecture hope to integrate two new areas in web caching namely Active Caching and Push Caching and invalidation protocols to provide caching of coarse grained dynamically generated content. The following would be the ideas used in this approach

- Most of the web servers dynamically add ads to the requested web page based on the user's profile. It uses some type of application logic to map between a user profile and the set of ads that are appropriate to him. In terms of dynamic caching what can be done is that both the most common ads on the server as well the code, which maps user profiles to these ads, can be cached on the proxy.

  So now a user's request for that site hits the web cache first. The cache can then request the user profile for that particular user from the web server, which stores the database containing the user profiles.

  The web cache activates the application that generates the dynamic web pages giving it as an argument the user profile retrieved from the web server. Then the application with the user profile and the set of the ads that have been cached can map between them and create part of the dynamic web

page containing the ads while the rest is retrieved from the server. It can then integrate these both and send the web page back to the client.

Since ads are usually large files like images and animations and these are cached on the proxy this would decrease the utilization of the bandwidth on the network as well as the delay. Also since the server only has to perform part of the processing, which cannot be done on the proxy, this decreases the load on it.

- The server in this case can actively push the ads to the web cache. If any of the ads on the server change then the server also pushes this new content across to the web cache. If the code that maps the user profiles to the ads changes for some reason, though this should not be the common case, the server actively pushes the new code across to the cache, which in turn starts using this new code in place of the old one.

- An optimization to this would be that the proxy could also store a set of pre-generated popular user profiles. So on a user's request it can return the closest matching cached profile instead of going through the process of building it dynamically from the ads again thus conserving CPU cycles. In case it does not find a similar page then it builds the page as described in the previous paragraph.

  The above approach makes a interesting tradeoff between higher memory usage, due to the storing of the pre-generated pages, and lower computation requirements on proxy. If the proxy is memory sufficient then maybe we could cache the most common user profile pages without having to generate them every time.

  If on the other hand it is memory constrained then we could build again and again even the most commonly asked for dynamic pages that we could not cache due to the memory constraint. Hence this could provide an interesting tradeoff between these resources, which is worth analyzing, and studying.

Issues involved:

- Mobile Dynamic Content Generation code that would be sent by the web server to be executed on the proxies.
- Active Push Caching of content from the web server to the proxies.
- Consistency between the content on the proxy and the web server.
- Maintaining some kind of mapping between the underlying data and the constructed web pages. So if some content changes then the proxy has to know which of the pre-built web pages depend on this now stale data and rebuild them once again.