Keyword-based Fragment Detection for Dynamic Web Content Delivery

Daniel Brodie, Amrish Gupta, and Weisong Shi
Wayne State University

ABSTRACT

Fragment-based caching has been proposed as a promising technique for dynamic Web content delivery and caching [2, 3, 4]. Most of these approaches either assume the fragment-based content is served by Web server automatically, or look at server-side caching only. There is no method of extracting fragments from an existing dynamic Web content, which is of great importance to the success of fragment-based caching. Also, current technologies for supporting dynamic fragments do not allow to take into account changes in fragment spatiality, which is a popular technique in dynamic and personalized Web site design. This paper describes our effort to address these shortcomings. The first, DyCA, a Dynamic Content Adapter, is a tool for creating fragment-based content from original dynamic content. Our second proposal is an augmentation to the ESI standard that will allow it to support looking up fragment locations in a mapping table that comes attached with the template. This allows the fragments to move across the document without needing to reserve the template.

Categories and Subject Descriptors

H.3.0 [**Information Systems**]: Information Storage and Retrieval

General Terms

Algorithms,Performance

Keywords

Dynamic Web Content Delivery, Fragment Detection

1. DYCA: DYNAMIC CONTENT ADAPTER

Existing Problem With the ESI The ESI [5] standard currently only allows for fragments to be linked directly with a url in the template. This means that fragment-based caching would not be able to take advantage of spatial changes, when a fragment of a document only moves in position on the template, but contains the same data. Currently, with the ESI, this can only be achieved by updating the fragments as if their content changed, but this is inefficient.

Design As mentioned earlier, DyCA has been designed to augment the existing servers to serve dynamic content efficiently. This is achieved by the DyCA design that generates dynamic content that adheres to an augmented ESI specification and also separates between the content generation and content delivery. DyCA is split up into 2 separate but very important parts, the Dynamic Content Generator module, and the Dynamic Content Delivery module.

The content generation module generates the fragments and template from the original dynamic Web content. It uses a keyword based approach to split up the static content into the different dynamic fragments. The keyword based approach works by building an XML of the website and searching this XML tree for specific tags that can signify a different fragment. These tags are set separately for different website based on the structure of the HTML and the content. Differences, such as a different font, a table, or a frame, can all be used to separate the fragments from the document. By looking at popular websites, it is fairly simple to see the implicit fragments contained in the document. Objects, such as sidebars on the website, can be easily distinguished, and a keyword based extraction method can then be easily created. Once the tags in the XML tree are identified, the children XML tags and the rest of the XML content contained in the identified tags is extracted to create the fragments. A special include tag, that has a special fragment id for each fragment, will be placed in the position where the fragment was extracted from the main document. At this point we have the fragment files, and a template with the fragment ids. The mapping table is just a list of the internal list of fragment ids and their corresponding URLs in a parseable file.

Additional information for each fragment, such as the TTL of the fragment, is calculated by looking at a long term overview of the website. Numerous instances of the web page are collected over a regular period of time. Each instance is parsed and separated into the different fragments, template, and mapping table by the content generation module. By comparing the fragments and their position over the period of time, an accurate dynamic behavior can be seen that allows the correct generation of the mapping table to be most efficient with regards to fragment movement. It also allows the generation of the TTL of the fragments to be supplied to the content delivery module.

The content delivery module is responsible for serving the template, fragments, and mapping table to downstream caches. The content delivery module needs to implement the extensions to the protocols in order to send the data created by the content generation module in an appropriate way. It needs to add information regarding the mapping table, and so notify the cache, when a request is dynamic and has a template, or when it is static. By sending the mapping table to the cache, it can then get the static URLs of the fragments to be able to access them. The content delivery module also needs to support the cache updating only its mapping table, so that bandwidth wouldn't need to be wasted about already cached fragments, or templates.

Implementation The dynamic content generator is implemented as a Java Servlet in an Apache Web server that parses existing websites and outputs a dynamic, ESI-based webpage. When a request from a cache for a document is made, if the document has fragments then the mapping table is looked up and added as an `X-CONCA-MAPPING` HTTP header to the response. The expiration of the fragment is set based on the TTL contained with the mapping table. The body of the response is just the ESI augmented

WWW2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

	No fragment cache	Static Template	Static Objects	Mapping Table
New York Times	4 MB	490 KB	246 KB	200 KB
India Times	4 MB	560 KB	220 KB	247 KB
Slashdot	2.4 MB	246 KB	202 KB	202 KB

Table 1: Total transfer bytes between server and cache.

template. Once the cache has the template, it will go on and request the fragment as needed from the server. The cache can then build up the proper final document and send that off to the client. The servlet's support of the If-Modified-Since part of the HTTP protocol on the template is crucial to the efficiency of the mapping table. When the template expires on the cache, the cache will request the template over again using the If-Modified-Since header of the HTTP protocol. Since the template rarely changes, this will mean that the cache will usually get a 'Not Modified' response. This response will contain the new mapping table, allowing the cache to update it's mapping table without having to retransfer the template. If a static document, or a static object is requested from the content delivery module, then the content delivery module will behave just like a regular web server.

ESI Augmentation In trying to remain as standard compliant as possible, the ESI format was picked to represent the structure of the template in DyCA. Yet the ESI standard only supports document fragments identified by static URLs, which will not suffice in our case. Thus we needed to augment the ESI standard by adding the `esi:xconca-include` tag. This tag allows the specification of an ID number that can be looked up by the cache in the mapping table and retrieve the fragment's static URL.

2. PERFORMANCE EVALUATION

The home pages from three websites, India Times, NewYork Times, and Slashdot, were chosen for performance evaluation of our proposed approach due to their dynamic nature, and since none of them supported any form of cacheable dynamic data. Ten-hour traces are used in the simulation for our performance evaluation.

Four approaches are compared in terms of both total transfer bytes and user perceived latency. The first method, using *no fragment caching*, was implemented by disabling caching in the proxy, and having the server send the original webpage. This is consistent with the behavior of real dynamic content in today's Internet. In the second method, *static template*, the template of the document remain static, while the fragments of the fragments are updated for content change. The template was cacheable for the whole testing session, while the fragments were cacheable for as long as their TTL was valid. In the third caching behavior, *static objects*, the template is updated when content moves across the document, and the fragments only change according to data changes only, and not spatial changes. The last model represents our proposed *mapping table* approach. When the template is returned a mapping table is returned with it, the proxy can then cache the mapping table, and update the mapping table when a spatial change happens. The template remained static for the testing session, while the fragments only changed for data changes.

2.1 Results and Analysis

Table 1 shows the total bytes used in transferring data in all four methods. As can be seen from the table, using a static template and changing to objects to support object movements requires considerable more data transfer between the server and the proxy. This is the method that is most commonly used today in dynamic websites. Using static objects and a dynamic template to achieve a dynamic webpage might seem efficient enough when looking at the bytes transferred, but, as we will see later, this efficiency is lost when looking at the user perceived latency. There is no comparison, though, between any of the dynamic methods and using traditional caching approach.The amount of data transfer when not using a fragment based architecture can be more then 10 times the amount of data transfer when using a good fragment based caching. When using a mapping table to transfer the data transfer can be seen to be considerably smaller. In fact, the total bytes transferred with a mapping table is little over the total size of the data, meaning a minimal amount of wasted data is being transferred. This should considerably reduce the server load when using such an architecture. Figure 1 shows the user perceived latency for NewYork Times. Due to space constraint we could not include the similar results for the other two sites that we have tested on, and they can be looked up in our technical report [1]. From this figure we see that the user has to wait the least time for the website when the mapping table architecture is used. With regard to user perceived latency, the static fragment method's performance is almost as bad as the performance of using regular static webpages. The only method that comes close to the method of using a mapping table is the static template method, which as we saw before performed badly when testing the amount of data transferred. This type of optimization is very important for the client so it may receive its data in a timely manner. Otherwise, from the users prospective, the actual retrieval of the page is slower. From these figures we can conclude that the mapping table method has performed better then all of the other proposed methods in all of our tested fields.

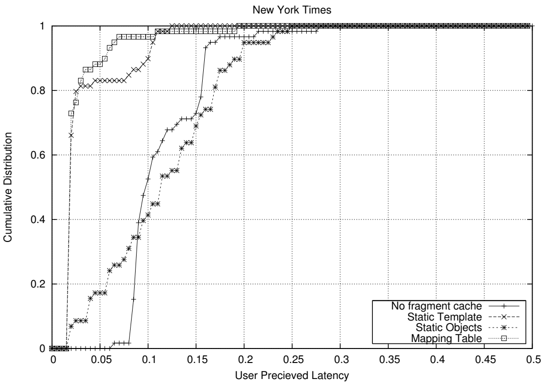

Figure 1: User perceived latency for NewYork Times.

3. SUMMARY AND FUTURE WORK

Our preliminary results show that the proposed keyword-based approach and mapping tables are very useful and promising. Our future work consists of continuing testing of these implementations to further refine the design of our CONCA prototype [4], which incorporates a novel design for efficient caching of dynamic and personalized content.

4. REFERENCES

[1] D. Brodie, A. Gupta, and W. Shi. Acelerating dynamic web content delivery using keyword-based fragement detection. Tech. Rep. CS-MIST-TR-2004-004, Department of Computer Science, Wayne State University, Feb. 2004.

[2] J. Challenger, A. Iyengar, and P. Dantzig. A scalable system for consistently caching dynamic web data. *Proceedings of INFOCOM'99*, Mar. 1999.

[3] F. Douglis, A. Haro, and M. Rabinovich. HPP:HTML macro-pre-processing to support dynamic document caching. *Proceedings of USITS'97*, Dec. 1997.

[4] W. Shi and V. Karamcheti. CONCA: An architecture for consistent nomadic content access. *Workshop on Cache, Coherence, and Consistency*, June 2001.

[5] M. Tsimelzon, B. Weihl, and L. Jacobs. ESI language sepcification 1.0, 2000, `http://www.esi.org`.