AN INTEGRATED APPROACH FOR REDUCING DYNAMIC WEB PAGE CONSTRUCTION TIME

Suresha

Database Systems Lab
SERC, Indian Institute of Science
Bangalore 560012, INDIA.
suresha@dsl.serc.iisc.ernet.in

Jayant R. Haritsa

Database Systems Lab
SERC, Indian Institute of Science
Bangalore 560012, INDIA.
haritsa@dsl.serc.iisc.ernet.in

ABSTRACT

e-Business web sites employ dynamic web pages to deliver more appropriate contents to their users, based on their customization and personalization. Such page generation puts heavy load on the web server, due to script execution overheads, leading to increased response time. The overall response time is made up of two types of latencies, namely network latency and server-side latency. In this paper we address primarily the server-side latency. We have proposed an integrated approach, which reduces dynamic web page construction time by more than 60% when compared to recently proposed fragment caching technique, for input independent dynamic web pages. We have integrated prediction based dynamic web page pre-generation with fragment-caching, assuming that a user-profile based page prediction is available. As we show subsequently our novel integrated approach reduces dynamic web page construction time by achieving both eventual benefit through fragment-caching and immediate benefit through dynamic web page pre-generation. We have developed a detailed simulation model and have provided the experimental results that validate our claims.

1. INTRODUCTION

In the recent years many web sites are adopting dynamic web page generation technologies to serve the page content dynamically, with significant flexibility in delivering custom contents to users. To provide web visitors the personalized experiences, web sites are increasingly relying on dynamic content generation applications. Dynamic web pages enable much wider range of interactions than static HTML pages can provide. However, these benefits come at a huge cost. Dynamic web page generation significantly reduces the web server scalability due to on-demand page generation. Hence, the performance and scalability are becoming major issues for e-business web sites as the Internet traffic is increasing tremendously every day, as well as web sites are becoming complex by attempting to generate more dynamic web pages. Over the past few years, web sites have transitioned from a static content model to a dynamic content model. The dynamic content generation increases the load on the web server infrastructure, and which in turn increases the response time. According to recent research, server-side latency accounts for 40% of the total page delivery time end users experience [12]. Delays can be detrimental for web sites, as users tend to leave a site if the response time is too long. A widely cited study by Zona Research has helped to quantify this phenomenon [9]. This study indicates that users will abandon requests with exponentially increasing probability as response time grows.

The potential bottlenecks in serving dynamic web pages are of two types: (a) network latency and (b) server-side latency. While much work has been done to address network latency, the attention is now turning to wards server-side latency. We have attempted to reduce server-side latency by integrating prediction based page pre-generation with recently proposed fragment caching technique [4, 3, 5]. We expect that the proposed integrated approach would make the page construction time as close to zero as possible. We assume user-profile based page prediction is provided.

1.1. Organization

The remainder of this paper is organized as follows: In Section 2 we give a brief description of related work. Then, in Section 3, we discuss our proposed architecture by integrating page pre-generation with fragment-caching. The Section 4 describes the simulation model used. In Section 5, the experimental results are discussed. The Section 6 summarizes our contributions, along with future work.

2. RELATED WORK

A widely used approach to address the world wide web performance problems is based on Content caching. A variety of such methods exist [8]. These caching approaches can be classified as proxy-based caching, server-side caching, and combined caching solutions.

2.1. Proxy-based Caching Approaches

Proxy-based caching approaches are based on caching content outside the site's infrastructure. Such content can include static content such as media files or HTML files. These solutions do not guarantee the correctness of output. In the recent years, the interest is to study the usefulness of using proxies to cache the output of dynamic web sites. Two broad approaches exist in using proxies to cache dynamic pages: *page-level caching* and *dynamic page assembly.* In Page-level caching approach, the proxy caches full page outputs of dynamic sites. There are many limitations associated with using page-level caching solutions to cache dynamic pages. Firstly, the page level caching solutions must rely on the request URL to identify pages in cache. Secondly, the page-level caching solution has often very little re-usability of full HTML pages. Thirdly, caching at the page level causes unnecessary invalidation, even if only one or a few elements on a page become invalid, then the entire page becomes invalid. Dynamic page assembly is an approach popularized by Akamai [10] as part of the Edge Side Includes (ESI) initiative [6]. A key drawback is the requirement that a site follows a specified page design paradigm, specifically, the use of templates. This requires that the page layout be known in advance.

2.2. Server-Side Caching Approaches

The server-side caching approaches can help to reduce the delays associated with generating content. Also, since they reside at the server, these solutions do guarantee the correctness of the output, unlike proxy-based caches.

There are approaches based on the idea of caching at the various layers within the site architecture. For instance, various types of database caching have been suggested, including caching the results of database queries [7] and caching database tables in main memory. But these solutions are limited in scope and do not address other delays associated with dynamic web page generation as such. There are some solutions based on caching of entire dynamic page [1]. The page level caching of dynamic pages has many limitations as discussed in Section 2.1. A recently proposed solution is based on caching of components of dynamic pages, called fragment caching [3]. This solution reduces dynamic page construction time, but limited by the fragment cacheability and the page construction takes place only after receiving the page request.

2.3. Combined Caching Approach

The Combined Caching approach was recently proposed in [2]. It caches dynamic content fragments in the proxy caches, but the layout information would be determined, on demand, from the source site infrastructure. The Combined Caching approach has the following limitations: The page construction for getting the page skeleton starts only after receiving the request by the server. The skeleton has to be scanned multiple times for filling the holes with fragment contents from proxy at each proxy level.

3. PROPOSED ARCHITECTURE

We propose an integrated architecture to reduce dynamic web page construction time. We integrate a page pre-generator with fragment caching solution proposed in [4, 3, 5]. We are making an assumption that most of the user sessions consists of sequence of input independent page clicks, may be with a very few input dependent page clicks. The input dependent pages are really hard to pre-generate and we are working on that. The page pre-generator can be implemented on

an independent processor and can independently start pre-generating the page assigned to it in consultation with application servers.

3.1. Existing Fragment Caching Model

When a user requests a dynamic page, his/her browser sends the corresponding URL to the web site and the web server maps this URL to a corresponding script and submits the script for execution to the application server and waits for the response from application server. The application server executes the script that generates the page. This script may access many data resources like DBMS, to retrieve the content required and create several objects and format the content for display. This step often requires significant work, especially for sites running complex business logic. The response from the application comes back to web server waiting for it. Then the web server sends the response to the browser.

A script can be thought of as a number of code blocks. Each code block carries out some computation to generate a part of the required page and results in an HTML fragment. An output statement after the code block places the resulting HTML fragment in a buffer. Once all the code blocks in a script have been executed, the resulted HTML is sent as a page to the user. If we know that a code block's output does not change for a sufficiently long time, then such a code block can be tagged as cacheable. When the script is executed, these tags instruct the application server to first check for the fragment in the fragment cache. If the requested fragment is found in the cache, then the code block execution is bypassed and the content is returned from the cache. If requested fragment is not found in the cache, then the code block is executed and the fragment is generated freshly and a copy of the fragment is also cached for eventual benefit. However, fragment-caching technique does not address the question **"what portion of the dynamic web page can be cached?"** and **does not totally eliminate script execution time**.

3.2. Proposed Integrated Model

An high level representation of our proposed architecture is given in Figure 1. We retain the fragment caching architecture given in [4, 3, 5] as it is. The

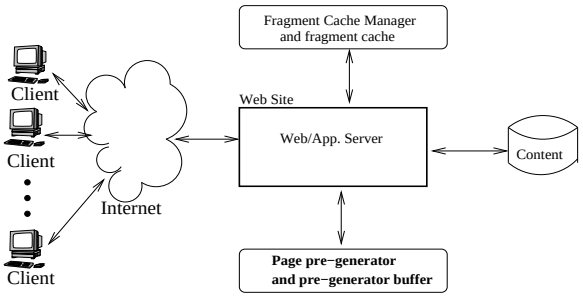

Figure 1: **Proposed Architecture**

page pre-generator is a separate engine. The working of the page pre-generator is controlled by web server. In our proposed approach, the web server maintains a set of recent sessions' information of users in-memory. If a user session is just starting, then an entry is made in the session list. For each user session, when a response for a request leaves the system, the system decides whether to pre-generate a next most expected page or not, for the same user, based on some prediction knowledge and other system considerations such system's current load, the benefit of pre-generating a page, the type of user and so on. Once the system decides to pre-generate a page for a particular user, it selects the most probable expected page for the user and starts pre-generating it. In fact, the page pre-generator submits the request to one of the application servers for executing the script. The selected application server carries out the execution of the script requested by the page pre-generator in consultation with fragment cache and returns the response back to the page pre-generator. The page pre-generator keeps the response in its buffer. There are several page predictions techniques available [15, 14, 13]. We assume that a user-profile based page prediction model is available to our proposed integrated approach.

When web server receives next page request for the same user in the same session, if the page is available at page pre-generator, it serves the page directly from the page pre-generator. If not, then page is freshly computed as usual. Note that the user response leaving the web server will take some time to reach the user and user will take some time to click the next page. By that time the page pre-generator can complete the page pre-generation. There by whenever the page pre-generated is same as the page requested by the user, then the page construction time is almost zero. If the same user is

not asking for any page for next some fixed amount of time, then the session is treated as over and the page pre-generated if any is discarded. The exact amount of time that how long the page pre-generator should wait before discarding the page pre-generated for a user session can be decided based on the available resources to page pre-generator, like buffer size.

In case of a hit we expect page construction time almost zero as the page would be ready by the time request reaches the site. Where as in case of a miss there is no wasted page construction time with respect to user requests. By fragment caching we are achieving the eventual benefit whenever the fragment is reused in course of time. Whereas by page pre-generation we are achieving immediate benefit.

The page pre-generator is allocated a buffer, which is a part of the fragment cache. The total size of all the pages pre-generated can not exceed this buffer size. When a new page pre-generated has to be put into the page pre-generator buffer and if the buffer is full, then a set of already pre-generated pages have to be vacated as victims, to make the room for new page pre-generated. We have used Least Recently Used (LRU) page replacement policy here. The pages in the page pre-generator are invalidated if any of the fragment involved in the assembly is invalidated.

4. SIMULATION MODEL

To evaluate the performance of the proposed architecture, we have developed a detailed simulator. Our simulation model is a scale down version of a web server. The web site is modeled as a directed graph. Each node in the graph represents a dynamic web page. Each page is connected to a number of other nodes representing hyper links, through which one can traverse the web site. The complete web site as a directed graph is pre-generated by a data generator.

The sessions are generated with an independent Poisson stream, with a given arrival rate. Each such session will generate many page requests. The parameters used in our simulation are given in table 1[without explanation due to space limitation].

Time units per second	100000
Fragment cache hit cpu time	10 time units
Fragment cache miss penalty	10 time units
Avg. number of pages in a session	10
Fragment invalidation rate	0.001 per sec.
Avg. think time betn. page requests	5 secs.
Avg. number of fragments in a page	10
Mean fragment size	2000 bytes
Mean fragment cost	2000 time units
Total number of fragments considered	8000

Table 1: **Simulation parameters**

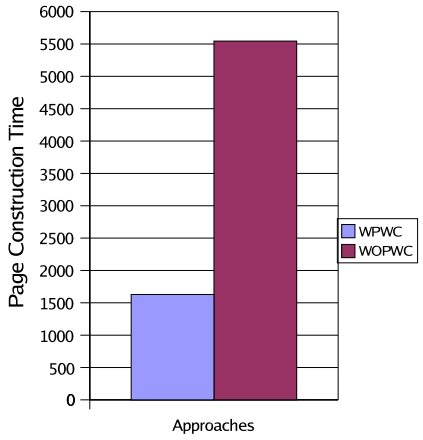

Figure 2: **Page Construction Time**

5. EXPERIMENTS AND RESULTS

We conducted an experiment to prove our idea. The main goal of this work is to reduce the dynamic web page construction time. In the experiment we have used an arrival rate of 1200 page requests per minute and a prediction accuracy of 60%. We have used pages with 80% of fragment-cacheability. We compare two strategies to construct dynamic web pages: The first strategy is when both page pre-generation and fragment caching are used: **With Page-pre-generation and With fragment Caching (WPWC)**. The second strategy is when only fragment caching is used, but without page pre-generation: **With Out Page-pre-generation and With fragment Caching (WOPWC)**. The Figure 2 gives the relative performance WPWC and WOPWC. It is clear from the Figure 2 that our integrated approach performs better than the fragment-caching technique.

6. CONCLUSION AND FUTURE WORK

We have proposed an Integrated Architecture to reduce the dynamic web page construction time assuming that a user-profile based page prediction model available to the web server. Our proposed solution reduces dynamic page construction time by integrating page pre-generation with fragment caching. Our preliminary experimental results have shown that our proposed model reduces the dynamic page construction time more than 60% when compared to fragment caching technique.

The most challenging part of our work which is still open is pre-generating those pages which are input dependent. At present what we have simulated is a scale down version. We are working on extending the work to a more realistic web site. We are carrying out experiments to study the proposed integrated approach with exhaustive set of inputs, the effect of cacheability, different prediction probability. We believe that the fragment access may not show any temporal locality of reference due to their random assembly in different pages. Hence we are working on suitable fragment-cache replacement policy. We also plan to study considering pre-generating more than one probable page and second level of storage for pages replaced due to page replacement at page pre-generator.

REFERENCES

[1] Arun Iyengar and Jim Challenger "Improving Web Server Performance by Caching Dynamic Data". *Proc. of Usenix Symp. Internet Technologies and Systems, Monterey, California, December 1997.*

[2] Anindya Datta, Kaushik Dutta, Helen Thomas, Debra VanderMeer, Suresha and Krithi Ramamritham, "Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web:An Approach and Implementation". *Proc. of ACM-SIGMOD Conference, June 2002, Madison, Wisconsin, pp. 97-108.*

[3] Anindya Datta, Kaushik Dutta, Helen Thomas, Debra VanderMeer and Krithi Ramamritham "Accelerating Dynamic Web Content Generation". *IEEE Internet Computing 6(5): 26-35 (2002).*

[4] Chutney Technologies, Inc. "Dynamic Content Acceleration: A Caching Solution to Enable Scalable Dynamic Web Page Generation". *Proc. of ACM SIGMOD Conference, Santa Barbara, CA, May 2001*

[5] Anindya Datta, Kaushik Dutta, Helen Thomas, Debra VanderMeer, Krithi Ramamritham, Dan Fishman "A Comparative Study of Alternative Middle Tier Caching Solutions to Support Dynamic Web Content Acceleration". *Proc. of VLDB Conference, Roma, Italy, 2001, pp. 667-670.*

[6] ESI Consortium. *Edge side includes. http://www.esi.org. 2001.*

[7] Q. Luo, J.F. Naughton, R. Krishnamurthy, P. Cao and Y.Li, "Active query caching for database web servers". *Proc. of WebDB 2000.*

[8] C. Mohan, "Tutorial: Caching technologies for web applications". *Proc. of VLDB Conference, September 2001.*

[9] Zona Research. *Quoted in Interactive Week Vol.6, No. 36, September 1996.*

[10] Akamai Technologies. *http://www.akamai.com.*

[11] K. Yagoub, D. Florescu, V. Issarny and P. Valduriez, "Caching strategies for data intensive web sites". *Proc. of VLDB Conference, 2000, pp. 188-199.*

[12] Christian Huitema. *"Network vs. server issues in end-to-end performance"*. Keynote address, Performance and Architecture of Web Servers workshop, held in conjunction with ACM SIGMETRICS, June 2000.

[13] S. Schechter, M. Krishnan, and M. D. Smith. "Using Path Profi les to Predict HTTP Requests". *Proc. of International World Wide Web Conference, Brisbane, Qld., Australia, April 1998, pp. 457–467.*

[14] Dan Duchamp. "Prefetching Hyperlinks". *Proc. of Second USENIX Symp. on Internet Technologies and Systems, Boulder, CO, 1999, pp. 127-138.*

[15] O Kit Hong, Fiona Robert P. Biuk-Aghai. "A Web Prefetching Model Based on Content Analysis". *Proc. of Macau IT Congress 1999, Macau, 17-20 March 1999, pp. 61-66.*