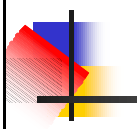
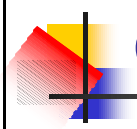


Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation

A. Datta, K. Dutta, H. Thomas, D. VanderMeer, Suresha, K. Ramamritham, (SIGMOD), 2002.

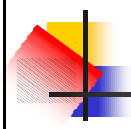
Presented by

James SHE
james@bbcr.uwaterloo.ca



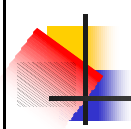
Quick Insight

- ..... is about a dynamic proxy caching technique combining the benefits of both proxy-based and back end caching approaches, but does not suffer from their limitations.....



Outline

- Basic ideas about Caching and Dynamic Content
- Existing approaches and their limitations
- Proposed Idea
- Technical Details
- Analytical and Experimental Results
- Conclusion



Basic ideas about Caching

- Caching is widely-used to improve the performance of WWW content distribution and delivery.
  The idea: Content generated for one user is saved, and used to serve subsequent requests for the same content.

In general, 2 types of basic caching approaches:

- Back-end caching - resides within a site, and caches at the granularity of a fragment (a portion of a web page)
- Proxy-based caching: stores content at the granularity of full web pages, and resides outside the site's infrastructure

Basics idea about Dynamic Content

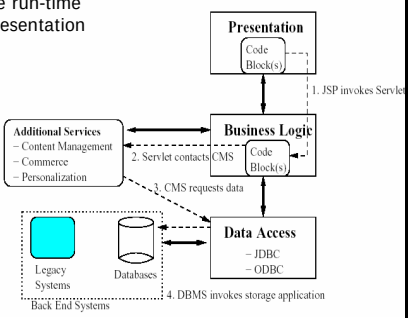
- A lot of Web Data today is dynamic and customized to individual's preferences,
  e.g. user's preferred stock quotes, a personal greetings
  Like IBM's WebSphere and BEA's WebLogic, they deal with dynamic page generation tasks and manage connections to back-end services (DBMS, Content Management, etc.)

- **2 dimensions of a dynamic website:**

1. Dynamic content - the actual information displayed in the run-time
2. Dynamic layout - a set of markup tags that define the presentation

In high level saying, a dynamic page is generated as below:

A user request --> maps to a script .jsp -->
the script execute the necessary logics -->
retrieve and process data --> format the content -->
return a customized page to the user



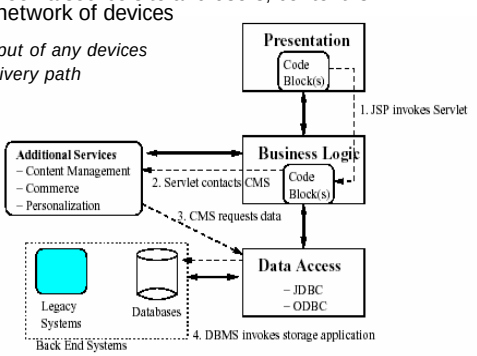
Performance Bottlenecks of Serving Dynamic Content

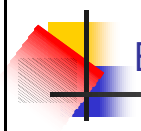
- **Network Latency**
  Usually, it's a long distance between a source site and users, content is delivered through an extensive network of devices

  The Delay is affected by the throughput of any devices (e.g. routers, switches) along the delivery path

- **System Latency**

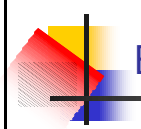
1. Session Processing Delay
2. Content Generation Delay





Existing Approaches for Dynamic Content

- Back-end Approaches
- Proxy-based Approaches



Back-end Approach for Dynamic Content

- **Database Caching Approaches:**

1. caches results of db queries
2. caches db tables in main memory

=> reduce delays associated with query processing operations

- **Presentation Layer Caching Approaches:**

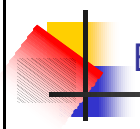
1. caches HTML fragments

=> reduce delays due to presentation layer tasks

- **Component Level Caching:**

1. caches arbitrary objects, including HTML fragments and programmatic objects

=> reduce computation and communication delays



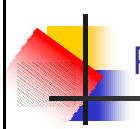
Back-end Approach: Benefits and Limitations

- **Benefits:**

1. Reduce delays in generating content
2. Guarantee the correctness of the output
3. Caching at finer granularities, achieve higher reuse of content and fine-grained invalidation

- **The major limitation:**

  All content are from the dynamic content application itself, do not address network related delays



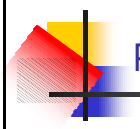
Proxy-based Approach for Dynamic Content

- **Page-level Caching:** cache full page outputs of dynamic sites

- **2 modes:**

  Reverse proxy mode - sitting between the site and the Internet cloud. e.g. Products from Inktomi, Network Appliance

  Forward proxy mode – i.e. deployed in distributed caching architectures located at numerous points around the Internet, also known as Content Delivery Networking (CDN). e.g. Akamai, Digital Island, SinoCDN



Page level caching: Benefits and Limitation

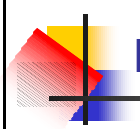
- **Benefits**

Reduce delays associated with

1. generating the content
2. packet filtering or firewall-related delays
3. bandwidth require to transmit the content from the back-end app. to the proxy-based cache

3 major limitations:

1. Rely on the request URL
2. Less reusability of full HTML pages
3. Unnecessary invalidation and regeneration. Think about a page with a stock quote, headlines, and historical research data



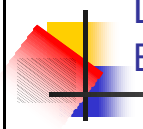
Dynamic Page Assembly

- **Dynamic page assembly:**

  establish a template for each dynamically generated page. The template specifies the content and layout of the page using a set of special markup tags.

- Each page is factored into a number of fragments (specifically, separate dynamic scripts) that are used to assemble the page at a network cache

- Content generated from templates and factored fragments are cacheable as separate HTML files in distributed caching architecture



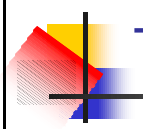
Dynamic Page Assembly: Benefits and Limitation

Benefits:

- Responses are assembled at distributed caching locations around the Internet, rather than hitting the origin server
- Faster response time and less network bandwidth consumption

Major Limitation:

- A specified page design paradigm for a site, specifically, the use of templates, which in turn call separate dynamic scripts for each dynamically generated fragment.
- Disabled the interdependency between fragments and duplicated calls to some fragments.
  e.g. A page the page consists of Personal greetings and Recommended products based on the same user profile object generated



The goals of this paper

- The objective is to deliver dynamic pages from proxy caches
- Recall: Dynamic pages are "dynamic" across two dimensions – content and layout
  => allow dynamic page layouts without a particular site design be enforced

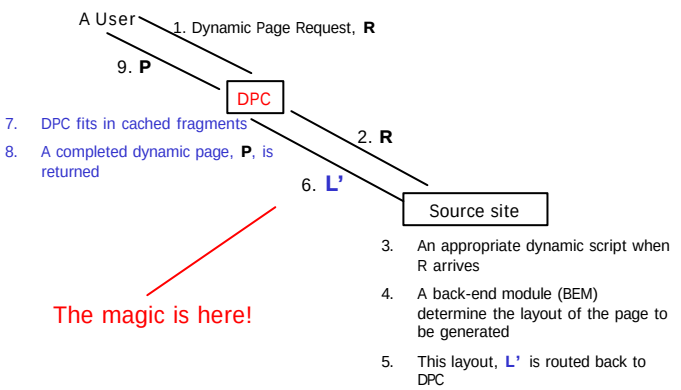
The intuition...

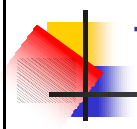
- The primary weakness of existing proxy caching schemes is their inability to map a URL to the appropriate content and layout

- **The essential intuition:**
  We will cache dynamic content fragments in the proxy caches, but the layout information would be determined, on demand, from the source site infrastructure.

The Proposed Approach

The DPC (Dynamic Proxy Cache) is a proxy cache that store dynamic fragments and assembles these fragments on demand using run-time page layout instruction





Technical Details

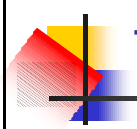
The DPC system consist of two main phases:
(a) System Initialization and (b) Run-time Operation

- System Initialization phase

  - *all cacheable fragments in a dynamic script will be identified and marked*

1. Once the cacheable fragments are identified,
2. Each corresponding code blocks in the dynamic script is tagged. The tagging is actually marking and indicating a code block, which generates a cacheable content,
3. A unique fragment id and metadata, say TTL, will be assigned to each cacheable fragments

- **This tagging process enables page layouts to be determined dynamically at run-time**



Technical Details

- **Run-Time Operation phase**

1. A user request comes
2. A script, catalog.jsp, runs as usual until a tagged code block is encountered
3. Check whether that fragment exists in the DPC (by doing lookup fragment ID in BEM's cache dir)

2 possible cases about the cacheable fragment:

- **not in cache or in cache but invalid**

1. An entry is inserted into the cache directory
2. The content is generated
3. A SET instruction is written to the layout
4. The fragment is inserted into DPC

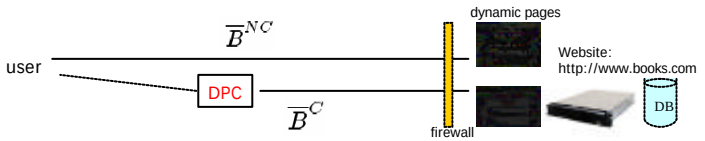
- **in cache and is valid:**

1. A key and a GET instruction are written to the layout L'
2. Retrieve the fragment from DPC

4. Similar processing for the remaining cacheable code blocks

Analytical Results

In our analysis, we want to compare bandwidth savings for two cases: (a) with DPC and (b) without cache



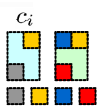
$\overline{B}^{NC}$: Without cache, the average number of bytes served by the Web site that is hosting the application during some time interval

$\overline{B}^{C}$: With DPC , the average number of bytes served by the Web site that is hosting the application during some time interval

Analytical Results

The Model:

- a given Web application with a set of pages, $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$
- the set of all possible fragments, $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$
- the set of fragments corresponding to page c_i, E_i, $E_i \subseteq \mathcal{E}$
- average size of a fragment, e_j
- size of the response corresponding to page c_i, S_{c_i}
- number of times the page c_i is accessed, $n_i(t)$
- average number of bytes served by the Web site that is hosting the application during some time interval $\overline{B}$



$\overline{B}$ over a given time interval is given by:

$$\overline{B} = \Sigma_{i=1}^{n} S_{c_i} \times n_i(t)$$

Analytical Results

For the term $n_i(t)$, let's characterize the access rate for a given page,

e.g. the probability that the Fiction category page is requested, and the arrival rate of requests to the www.books.com site

Let...

$\mathcal{P}(i)$ be the probability that page c_i is accessed for a given request, in which $\mathcal{P}(i)$ is governed by the Zipfian distribution

$f(t)$ be the probability density function (pdf) that describes the arrival rate of requests

So, the number of times $n_i(t)$ that page c_i is accessed during the time interval (t_1,t_2) can be figured out as below:

$$n_i(t) = \mathcal{P}(i) \int_{t_1}^{t_2} f(t)dt$$

Analytical Results

Parameter	Value
hit ratio (h)	0.8
fragment size (s_e)	1K bytes
number of fragments per page	4
number of pages	10
average size of header information (f)	500 bytes
tag size (g)	10 bytes
cacheability factor	0.6
number of requests during interval (R)	1 million

For the term S_{c_i}, we have 2 cases:

1. No cache: $S_{c_i}^{NC} = \Sigma_{\forall e_j \in c_i} s_{e_j} + f$

   * Each page has f bytes of header information

2. With cache:

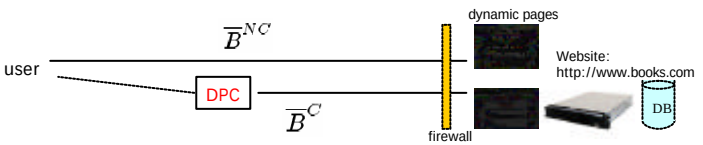
$$S_{c_i}^{C} = \Sigma_{\forall e_j \in c_i}[X_j[(h \times g) + (1-h)(s_{e_j} + 2g)] + (1 - X_j)(s_{e_j}) + f]$$

$$\forall e_j \in E, X_j = \begin{cases} 1 & \text{if fragment } e_j \text{ is cacheable} \\ 0 & \text{otherwise} \end{cases}$$

So, now we can figure out the values of $\overline{B}^{NC}$ and $\overline{B}^{C}$ as well as their ratio $\frac{\overline{B}^{C}}{\overline{B}^{NC}}$
Done! However, we want to know the cost as well.

Analytical Results

There is a cost to leverage the advantages of DPC!!!



Assembly of the page at DPC requires that each response be scanned for the tags.

So, does the savings ($\frac{\overline{B}^{C}}{\overline{B}^{NC}}$ or $\overline{B}^{NC} - \overline{B}^{C}$) in bytes transferred offset the cost to scan?

Analytical Results

Let y and z be the scan cost per byte for the firewall and DPC

No cache: $scanCost^{NC} = \overline{B}^{NC} \times y \quad (1)$

With cache: $scanCost^{C} = \overline{B}^{C}(y + z)$

$scanCost^{C} = \overline{B}^{C} \times 2y \quad (2)$

string matching algorithms (e.g., KMP [18])
Linear time algorithm, same orders of costs in firewall and DPC
=> $z \approx y$

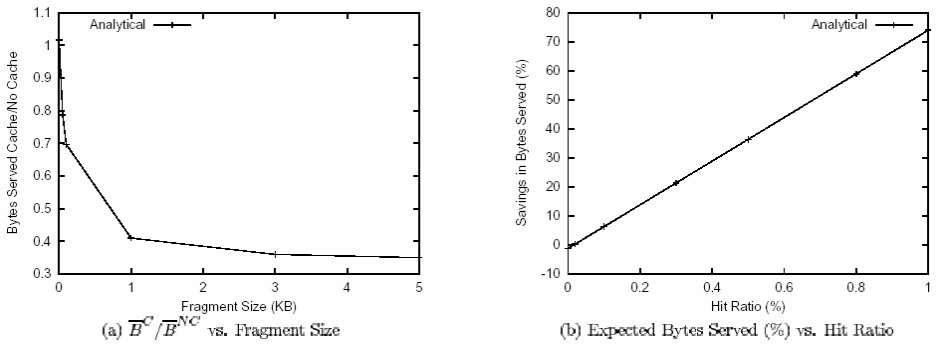
To justify the use of DPC, we should have (1) > (2), i.e.

$$\overline{B}^{NC} > 2\overline{B}^{C}$$

* An effective indicator to determine the dynamic page/site to be cached by DPC or not.

RESULT 1. *It is preferable to use the dynamic proxy cache when the expected bytes served with no cache are more than twice the expected bytes served with cache.*

Analytical Results



(a) $\overline{B}^{C}/\overline{B}^{NC}$ vs. Fragment Size

(b) Expected Bytes Served (%) vs. Hit Ratio

Figure 2: Analytical Results - Expected Bytes Served

Experimental Results

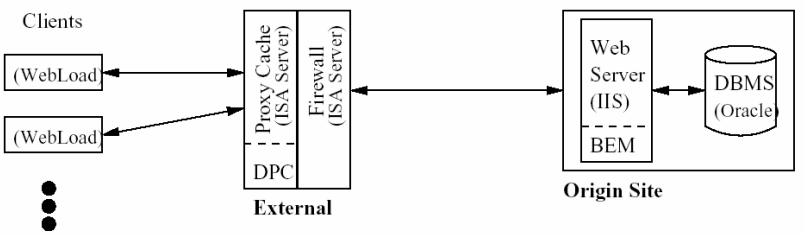
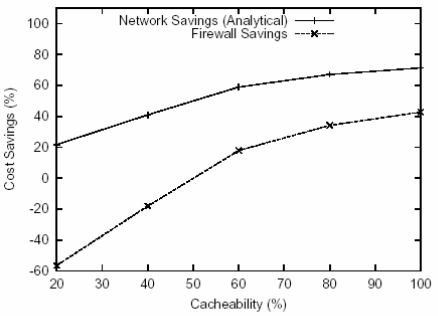
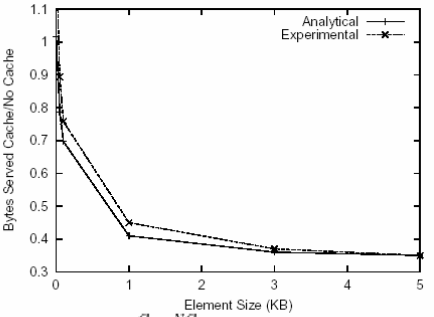


Figure 4: Test Configuration

Analytical and Experimental Results



(a) Comparison of Cost Savings



(b) $\overline{B}^C/\overline{B}^{NC}$ vs. Fragment Size

Figure 3: Analytical and Experimental Results

Analytical and Experimental Results

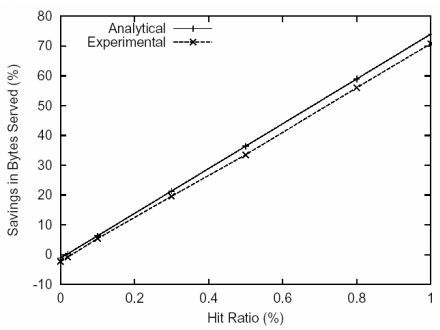


Figure 5: Expected Bytes Served (%) vs. Hit R

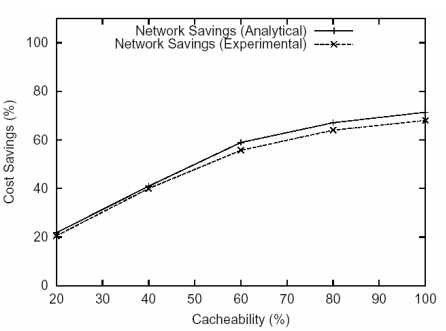
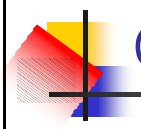
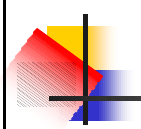


Figure 6: Validation of Cost Savings



Conclusion

- The idea is neat and realistic
- An effective approach for proxy-based caching of dynamic content with a higher granularity caching and validation.
- Allow both the content and layout to be flexibly dynamic
- Combining benefits of proxy-based and back-end caching approaches, without their respective limitations
- Proven with real implementation and experimental results
- Enable significant reductions in bandwidth requirements as well as end-to-end response times.



End / Q&A