E-business sites are increasingly utilizing dynamic web pages. Dynamic page generation technologies allow a web site to generate pages at run-time. But very little work has been done so far to address the dynamic page generation delays. The proposed approach is based on concept of caching entire pages of dynamically generated content. The novelty of this approach lies not only in the caching of dynamic page fragments but also in the utilization of intelligent cache management strategies.

6.1 INTRODUCTION

Improvements in networking and internet technology have had a significant impact for the need to access thousands of heterogeneous information resources on various platforms [WPA2006] since they could be immediately made available for many users through the existing computer networks. But there is a growing need for tools to maximize the portability, reusability and interoperability of arbitrary computing services while maintaining the autonomy of the pre-existing applications in a federated approach. The problems of achieving interoperability among heterogeneous and autonomous sources, global query decomposition and optimization of queries have long been a subject of heavy research in the field [USR2006].

Web Caching is the caching of web documents (e.g. web pages, images) in order to reduce bandwidth usage, server load and improve latency time. A web cache stores copy of documents passing through it. Florescu et al. [DFL1999] proposed a model where caching helps to bridge the performance gap between local activity and remote content. In short term, caching helps to improve web performance by reducing the cost and end-user latency for web access.

One approach to coordinate caches in the same system is to set up a caching hierarchy. With hierarchical caching, caches are placed at multiple levels of the network [HSU1999]. In distributed web caching system proposed by Christian et al. [PRO1994], there are no other
intermediate cache levels than the institutional caches which serve each other’s misses. In
direct order to decide from which institutional cache to retrieve a miss document, all institutional
caches keep meta-data information about the content of every other institutional cache. With
distributed caching [SAD1996], most of the traffic flows through low network levels which
are less congested. In addition, distributed caching allows better load sharing and is more
fault tolerant [EIC1994]. Nevertheless, a large-scale deployment of distributed caching may
encounter several problems such as high connection times, higher bandwidth usage, administrative issues etc. There are several approaches to the distributed caching [BBH2003]
like Internet Cache Protocol (ICP) which supports discovery and retrieval of documents from
neighbouring caches as well as parent caches. Another approach to distributed caching is the
Cache Array Routing protocol (CARP), which divides the URL (Uniform Resource Locator)-
space among an array of loosely coupled caches and lets each cache store only the documents
whose URL are hashed to it [GBA2000].

6.2 PROBLEM DURING WEB CACHE BASED QUERY HANDLING
Dynamic scripting technologies such as Active Server Pages (ASP) and Java Server Pages
(JSP), allow web sites to assemble pages based on various run-time parameters in an attempt
to tailor content to each individual user [JWA1999]. A major disadvantage of such dynamic
scripting technologies, however, is that they reduce web and application server scalability
because of the additional load placed on the web/application server. In addition to pure script
execution overhead, the delays caused by dynamic scripting technologies include: delays due
to fetching content from persistent storage (e.g. database systems), delays due to data
transformations (e.g. XML to HTML transformations) and delays due to executing business
logic (e.g. personalization software). The main emphasis of the proposed model is to address
the problem of dynamic page generation delays.

6.3 PROPOSED MODEL
In the proposed model, query processing through cache maintenance has been performed at
three levels: one at user level, second at proxy server level and the third at database server
level. When there is any request for a query from user, proxy server searches for results in
database server cache firstly and results are given back to the end user as shown in Figure
Chapter 6: Three Tier Cache Based Query Optimization model in Distributed Database

6.1. The result is also stored in local cache of user so that if the same query is generated by the same user again, then the results can be found easily in the local web cache. But if the same query is generated by another user, then proxy server checks for the results of that particular query in its proxy cache where results of queries are stored. Due to this, the query submitted by user need not to go to the cache of database server. This approach reduces the time taken for searching the required information.

![Diagram of Three Tier Cache Based Query Optimization model]

**Figure 6.1 Architecture for Web Cache Based Query Optimization in Distributed Database**

6.4 PROPOSED SOLUTION

To address the problem of dynamic page generation delays, the modified Dynamic Content Acceleration (DCA) solution utilizes a fragment-level caching approach which focuses on reusing HTML fragments of dynamic pages. A dynamic script typically consists of a number of code blocks where each code block performs some work that is required to generate the page and produces an HTML fragment as output. A ‘write to out’ statement which follows each code block, places the resulting HTML fragment in a buffer. When a dynamic script runs, each code block is executed and the resulting HTML fragment is placed in the buffer.
Once all code blocks have executed, the entire HTML page is sent as a stream to the user. A high-level outline of the dynamic scripting process for news page is shown in the Figure 6.2. If we know that the current headlines and navigation components are reusable, we may choose to cache these components. This is accomplished by marking or tagging the corresponding code blocks within the script. When the script is executed, the tags instruct the application server to first check the cache before executing the code block. If the requested fragment is found in cache, then the corresponding code block logic is bypassed. Otherwise, the code block is executed; the requested fragment is generated and subsequently placed in the cache.

Figure 6.2 Dynamic Scripting Processes for News Page

A critical aspect of any caching solution is cache management. As the cache becomes full, the effectiveness of the cache replacement policy dictates the hit rates of the cache and thus its performance. The cache replacement algorithm is based on a predictive technique. When choosing a replacement victim, it takes into account not only how recently a cached item has been referenced but also whether any user is likely to need the item in the near future. Also, as the underlying source data changes, some mechanism is required to keep the cached components fresh. The solution supports several existing invalidation techniques (e.g. time-
based and event-based invalidation) which have been adapted to work in the context of our component level cache. A simplified framework (without routers, firewalls etc.) of an end-to-end web site architecture is shown in Figure 6.3, where DCA sits adjacent to the server rack, along with other resources such as site content databases.

![Figure 6.3 An End-to-End Web Site Architecture](image)

### 6.5 PERFORMANCE ANALYSIS

The main contribution of this work provides the mechanisms for intelligent and fragmented caching by treating the static and dynamic information separately. The tag based classification of dynamically updated information utilizes the code block to perform the fragmentation. This saves the data retrieval time by preventing the complete page to be recovered and thus provides query optimization. Moreover, the performance is enhanced by increased throughput and scalability which are inherently provided by the implementation of caching.

Figure 6.4 shows how proposed scheme is efficient in improving query optimization as compared to existing schemes. According to Luo et al. [QLU2008], processing time for executing the requested pages increases in the same ratio as the number of pages increases. On the other hand, due to the caching of static part (reusable) part and generation of dynamic part as and when required, there is gradual increase in processing time in the proposed scheme. For example, if requested page size is 100 kb consisting of 5 code blocks in which 2 code blocks are reusable and are available in cache then only 3 code blocks need to be regenerated. We can say that only 60% of the whole page needs to be regenerated. This leads
to faster generation of results for the requesting queries using proposed scheme than the old
schemes. Hence, it saves the cache storage space and fulfils the aim of query optimization.

![Graph](image)

**Figure 6.4 Performance Analysis in Terms of Processing Time**

### 6.6 MATHEMATICAL MODEL FOR THE PROPOSED ARCHITECTURE

In our proposed model, Cache is considered as a collection of $n_m$ modules rather than a
simple collection of $n_p$ pages.

\[ n_m = \sum_{i=1}^{n_p} m_i \]  \hspace{1cm} (1)

Where $m_i$ is number of modules on $i^{th}$ page

$n_p$ is total number of pages

Total number of bytes occupied by $n_m$ modules is calculated as:

\[ \sum_{i=1}^{n_m} n_b(i) \]  \hspace{1cm} (2)

Where $n_b(i) =$ number of bytes in $i^{th}$ module.

Now if a user requests for $m_i$ and $m_j$ modules which may or may not be located on same
page and let $m_i$ and $m_j$ be present on $p_i$ and $p_j$ page (where $i=j$ or $i\neq j$).

In existing scheme, total number of bytes transferred for such requests are:
Chapter 6: Three Tier Cache Based Query Optimization model in Distributed Database

\[ B_i = p_b(i) + p_b(j) \]  \hspace{1cm} (3)

Where \( B_i \) is the bytes transferred and \( p_b(i) \) and \( p_b(j) \) are total number of bytes on \( i^{th} \) and \( j^{th} \) pages respectively.

In the proposed scheme, since only the requested modules are of prime concern, so number of bytes transferred \( b_t \) for such requests are:

\[ b_t = n_b(i) + n_b(j) \]  \hspace{1cm} (4)

The number of bits transferred is an effective parameter to calculate the bandwidth utilization. Now we define a bandwidth utilization factor \( \beta_\mu \) that is the ratio of number of bits transmitted in existing and the proposed scheme:

\[ \beta_\mu = \frac{B_t}{b_t} = \frac{p_b(i) + p_b(j)}{n_b(i) + n_b(j)} \]  \hspace{1cm} (5)

To generalize for \( k \) requested modules:

\[ \beta_\mu(gen) = \frac{\sum_{i=1}^{k} p_b(i)}{\sum_{i=1}^{k} n_b(i)} \]  \hspace{1cm} (6)

The higher the value of \( \beta_\mu(gen) \), higher is the bandwidth utilization.

Now another important parameter considered in cache management is query response time. If ‘\( t \)’ is the time for 1 byte to travel over network, then in existing scheme, total time taken for byte transfer (\( t_T \)) i.e. query response time is:

\[ t_T = \left( \sum_{i=1}^{k} p_b(i) \right) t \]  \hspace{1cm} (7)

Whereas in the proposed scheme, total time taken for byte transfer (\( T_p \)) i.e. query response time is:
Chapter 6: Three Tier Cache Based Query Optimization model in Distributed Database

\[ T_p = \left( \sum_{i=1}^{k} n_b(i) \right) t \]  

(8)

Therefore the query response time is enhanced by a factor of \( Q_t \) when the proposed scheme is used

\[ Q_t = \frac{t_T}{T_p} = \left( \frac{\sum_{i=1}^{k} P_b(i)}{\sum_{i=1}^{k} n_b(i)} \right)^t \]  

(9)

An Example:

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>100</td>
<td>250</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>P2</td>
<td>300</td>
<td></td>
<td>200</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>P3</td>
<td>150</td>
<td>600</td>
<td>400</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 6.1 Page-Module Table

Suppose a user requests for module M1 from P1 and module M3 from P2 consisting of 100kb and 200kb data respectively. In existing scheme, upon a request from user, both dynamic pages will be executed causing data transfer in order of 100 for M1, 250 for M2 in P1 and 300 for M1, 200 for M3 modules in P2 respectively causing total data transfer equal to 850kb, whereas in the proposed scheme only a fragment of P1 and P2 is executed i.e. 100kb of M1 in P1 and 200kb of M3 in P2 respectively causing total data transfer equals to 300kb only.

So, bandwidth utilization factor:

\[ \beta_u = \frac{850}{300} = 2.8 \]

i.e. bandwidth utilization using the proposed scheme is 2.8 times better than using the traditional scheme in case of given example.
Chapter 6: Three Tier Cache Based Query Optimization model in Distributed Database

In the proposed scheme, a two dimensional matrix table named Page-Module table shown as Table 6.1 is maintained in Apache cache which keeps the records of all dynamic requests and maintains a record of number of bytes occupied per module in a page so as to give the total bytes consumed per page.

6.7 CONCLUSION
Web Caching in Internet reduces the latency because the request is satisfied from the cache (which is closer to the client) instead of the origin server. In our proposed scheme, it takes less time for the client to get the object and display it and causes reduction in traffic because each object is retrieved from the server only once and it also reduces the amount of bandwidth used by a client. This saves money if the client is paying by traffic and keeps their bandwidth requirements lower and more manageable.

In future, this work will be enhanced by sharing of caching between all the users who request the same web page which is now working for personal caching for every user. To implement shared caching between all the users, concept of clustering will be used for caching in distributed databases. In future, there will be blocks for every user separately to store the information regarding its cache and all those blocks will be sharing all the information of all users on the basis of request given to its local server. For this purpose the main issue will be security in caching for every user to make synchronization between all the users caching.