ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 9, September 2014

High Performance E-Business using Application Level Caching

Shaina[1], Mrs.Anshu Kamboj[2]

Student, CSE, JCDMCOE, Sirsa, India[1]

Assistant Professor, CSE, JCDMCOE, Sirsa, India[2]

Abstract: E-commerce applications face many challenges in cost and manageability, especially for database servers that are deployed as those application's backends in a multi-tier configuration. One solution is middle-Tier Database. For E-Commerce websites there are mainly two issues, cost and time. As the numbers of users increase, performance related issues needs to be solved by using caching concepts. In E-Commerce websites dynamic pages are used to provide wider range of interaction than static HTML pages. At the same time, much performance related issues arises by using dynamic page generation technologies because of load placed on server-side resources. For many web applications system support for caching is insufficient. This Paper provides a new way of Application level caching to improve performance of web applications. Concept of shared and unshared caching has been used in proposed method.

Keywords: Caching, Proxy, Shared, Unshared, Application.

I. INTRODUCTION

In today Scenario regarding performance, there are various techniques of caching which have been used for deploy in E-Service like E-Business to increase the performance web-based applications in for increase the performance of multi tier architecture services on the Internet. These applications are critical application with regards of accessibility such that the information should be right with us on right time. High performance implies cost of servers according to the type if application which are running on servers. This type of applications achieve a measure of scalability with application servers running on multiple systems connecting to a single database system. For solving the issue of performance, sever side issues can be resolved using caching.

The business related web services are often critical part of infrastructure which is needed for the success of organization in terms of processing time. The application performance benefits can be achieved by the query calculation. The cached objects are often stored in hash table and balanced tree and indexing which can be byte stream, numerical value or by the Strings. The application level cache has been designed with API which allows the developer to manage the cache contents explicitly. There are all features likely to the database which can be update or Modify, Add and Delete. The developers can easily work with the cache by taking the knowledge of application specific Structure when caching of data. There are different approaches for caching which can be Unshared and Shared.

Application Caching can be done either on the client side, in between the client and the server, or on the server side (data caching/page output caching). So we can classify caching locations like this:

1. Client Caching
2. Proxy Caching
3. Reverse Proxy Caching
4. Web Server Caching

In web server caching, cached data is stored inside the web server. Data caching and page caching uses the web server caching mechanism. It will increase performance of website. We will work on web server caching as shown in Fig. 1

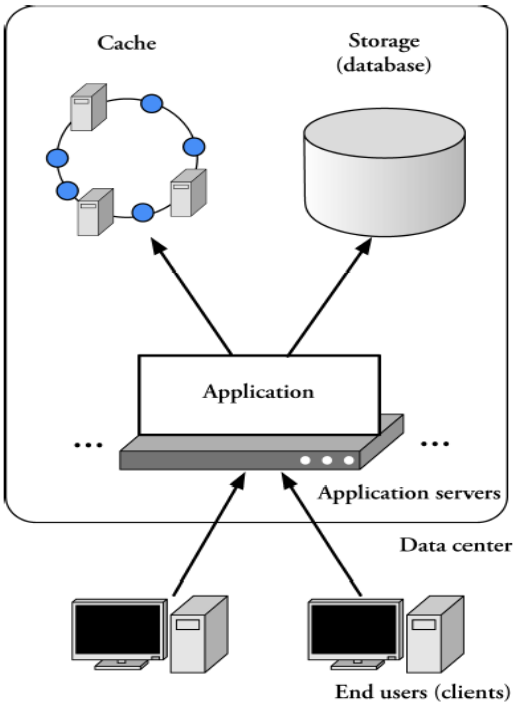

Fig 1 Application Level Cache

II. LITERATURE REVIEW

The author has introduced client-side data-caching scheme for relational databases with a central server and multiple clients. Based on queries executed on the central database at the server data is loaded into each client cache. Predicates that describe the cache contents are formed using these queries. A subsequent query at the client may be fulfilled in its local cache if we can find out that the query result is entirely contained in the cache. This matter

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 9, September 2014

is called cache completeness. A separate matter, cache currency, deals with the effect on client caches of updates committed at the central database. Various performance tradeoffs and optimization issues involved in addressing the questions of cache currency and completeness are examined by us using predicate descriptions and suggest solutions that promote good dynamic behaviour. Lower query-response times, less message traffic, superior server throughput, and better scalability are some of the expected benefits of our approach over commonly used relational server-side and object ID-based or page-based client-side caching [1].

A semantic model has been proposed for client-side caching and replacement in a client-server database system and this approach is compared to page caching and tuple caching strategies. Our caching model is based on, and its advantages are derived from, three key ideas. Usage information for replacement policies is maintained in an adaptive fashion for semantic regions, which are related with collections of tuples [4].

A transactional data cache consistency maintenance algorithm is required to make ensure that such caching does not result in the violation of transaction semantics. So many algorithms have been proposed in the literature and, as all provide the same functionality, primary concern is performance in choosing among them. In this article we a taxonomy is presented by us that describes the design space for transactional cache consistency maintenance algorithms and show how proposed algorithms relate to one another. Then performance of six of these algorithms is investigated by us, and these results are used to examine the tradeoffs inherent in the design choices identified in the taxonomy. It is shown by the results that the interactions among dimensions of the design space can impact performance in many ways, and that classification of algorithms as simply “pessimistic” or “optimistic” do not exactly characterize the similarities and differences among the many possible cache consistency algorithms [5] For scalable web content delivery at web intermediaries cache consistency is required. The Web Content Distribution protocol (WCDP) has been introduced, which is an invalidation and update protocol to provide cache consistency for a large number of often changing web objects. WCDP supports different levels of consistency: strong, delta, weak, and explicit. It supports atomic invalidates and mutual consistency among objects and handles multiple deployment architectures. By grouping objects and messages together and by using a hierarchical organization for message delivery, scalability is achieved by WCDP. WCDP operates between the origin server, mirror sites, and the participating web intermediaries. It is not, however, targeted for inter-CDN operations [11].

III. OBJECTIVES

For middle-tier database caching, using a general-purpose industrial-strength DBMS is especially attractive to e- Businesses. This is mainly due to crucial business requirements such as reliability, scalability and manageability. For instance, an industrial-strength DBMS, provides a variety of tools for application development and closely tracks SQL enhancements. More importantly, transactional support, multiple consistency levels, and efficient recovery services is provided by it. Finally, an ideal cache should be transparent to the application that uses it, and with a special-purpose solution it is very difficult to achieve.

Existing Problems:

1. High Response time in E-Commerce websites due to High number of request/ Load.
2. Fetch same data from database for every User.
3. Repeat processing for same kind of search.
4. Multiple query execution at same time decrease site performance.
5. Shared caching architecture having problem of higher latency for accessing a cached object because of inter-process communication.
6. Unshared caching is required extra space for storing all n copies. Maintain Consistency is also required.

Objectives:

1. Remove disadvantages of Shared and Unshared architecture of caching.
2. Design a new Architecture, Hybrid architecture by combining both caching architectures.
3. Design an Algorithm which will handle problem of same data fetching process by maintaining search keywords and counts.
4. Remove caching Memory to store new Data.

IV. PROPOSED METHODOLOGY

While using a full-fledged database engine for middle-tier database caching much research question arises.

The answers to some of them affect the relevance of others. In decreasing order of importance, these are.

1. What are the performance issues in e-Business applications, or in other words, and does the right problems are addressed by us focusing on database caching?
2. Will performance be acceptable using a commercial DBMS as a middle-tier data cache? Features such as transactional semantics, consistency, and recovery come with some operating cost. What features can be dispensed with in such an situation?
3. What database caching schemes are suitable for e-Commerce applications?
4. How can a database caching scheme be implemented in a business database engine and how does it perform under practical e-Commerce workloads?
5. What is the consequence of running a database server in the same computer as an application server?
6. How these results can be generalized to other kinds of web applications?

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 9, September 2014

Work Flow of proposed Method:

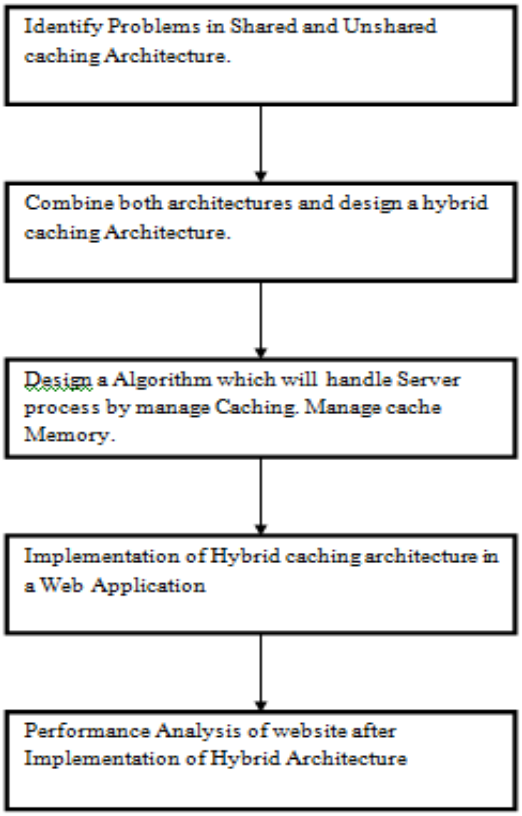

Fig 2 Work Flow

V. CONCLUSION AND FUTURE WORK

We have examined the opportunities in e-Commerce applications for middle-tier database caching. We have categorized the events and use hybrid approach of Caching by combining shared and unshared caching according to nature of event. We have measured Page Load Time for data returned from Database and cache and compared them.

It's clear from results that caching is effective for large number of user requests and same kind of searching. Site performance can be increased using this caching scheme.
A web application has been designed in Asp.net with caching management algorithm. Results are measured by running it in local system. This site can be deployed on server. Log files will be generated in home directory by name: "Home.txt" and "Search.txt".

Future work includes extending the Caching Scheme and Algorithm to handle special SQL data types, statements, and user defined functions. Investigating of alternatives for handling frequent database updates. Usability enhancements, such as cache performance monitoring, and dynamic identification of candidate tables for caching are important directions for us to pursue.

REFERNCES

[1] Arthur M. Keller and Julie Basu, 1995, A predicate-based caching scheme for client-server database architectures
[2] Sibel Adali, K. Selçuk Candan, Yannis Papakonstantinou, and V. S. Subrahmanian. Query Caching and Optimization in Distributed Mediator Systems. Proc. ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 1996.
[3] Abdelsalam Heddaya, 1996, WebWave: Globally Load Balanced Fully Distributed Caching of Hot Published Documents.
[4] Shaul Dar, 1996, Semantic Data Caching and Replacement
[5] Michael j. Franklin, Transactional Client-Server Cache Consistency: Alternatives and Performance, ACM Transactions on Database Systems, Vol. 22, No. 3, September 1997.
[6] Jim Challenger, Arun Iyengar, and Paul Dantzig. A Scalable System for Consistently Caching Dynamic Web Data. IEEE INFOCOM 1999.
[7] Boris Chidlovskii, Claudia Roncancio, and Marie-Luise Schneider. Cache Mechanism for Heterogeneous Web Querying. Proc. 8th World Wide Web Conferences (WWW8), Toronto, Canada, 1999.
[8] Jim Challenger, Arun Iyengar, and Paul Dantzig,1999, A Scalable System for Consistently Caching Dynamic Web Data.
[9] Yeol Song, 2000, "Database Design for Real-World E-Commerce Systems", IEEE
[10] K. Johnson, J. Carr, M. Day, and M. Kaashoek. 2000, The measured performance of content distribution networks. In 5th Int. Web Caching and Content Delivery Workshop, Lisbon, Portugal.
[11] B. Krishnamurthy and C. Wills.2000, Analyzing factors that in ence end-to-end web performance. In International World Wide Web Conference.
[12] A. Labrinidis and N. Roussopoulos.2000, WebView Materialization. In ACM SIGMO.