Improve Query Caching for Dynamic Content Web Sites

Kakkatil Binita [1], Chandan pal singh [2], Surendra kr.Jangir [3]

[1]Department of Electronics Engineering, PIIT,Navi Mumbai, Maharashtra, INDIA
[2]Departmentof CS & IT,Jagannath University, Jaipur, Rajasthan,INDIA
[3]Departmentof CS & IT,Jagannath University, Jaipur, Rajasthan,INDIA
Binik2@gmail.com[1], chandanengg@gmail.com[2], jangir_cse@yahoo.co.in [3]

ABSTRACT

In this paper, we study how improves performance of query caching for dynamic content web sites. The dynamic web server is composed of a traditional web server attached to an application server. The web servers are typically replicated for load balancing and fault tolerance. The application server performs the necessary computation to determine what data is required from the database. It then queries the database and constructs replies based on the results of the queries. For the purpose of this paper we consider the combination of the web server and the application logic collectively as the front-end and the database as the back-end. We propose and apply a cycle of optimizations for a fully crystal clear dynamic content cache suitable for any web application. We optimize the query cache to handle full and partial coverage of query results at the expense of additional processing. We propose several optimizations that reduce the load on the database leading to higher performance.

Index Terms— *query cache, web server, application server, web application, database*

1. INTRODUCTION

Dynamic web pages are built according to a user's references or depend on mutable data. The dynamic web server is composed of a traditional web server (WS) attached to an application server (AS). The web servers are typically replicated for load balancing and fault tolerance. When a user accesses a dynamic web site, the request is received by the web server and then forwarded to the application server. The application server performs the necessary computation to determine what data is required from the database. It then queries the database and constructs replies based on the results of the queries. For the purpose of this paper we consider the combination of the web server and the application logic collectively as the front-end and the database as the back-end. To speed up the delivery of dynamic web pages, database query caching has been proposed by [1, 2]. With query caching, the results of recent queries are cached locally and are reused on later queries. By caching, both the latency of retrieving the results and the load on the back-end is reduced. Caching dynamic content is more complex than caching static content, because the cached entries may become invalid as a result of database updates. Moreover, earlier designs lack the ability to retrieve partial results from the cache. In this paper, we propose several optimizations that turn misses caused by INSERT queries into partial misses. These further reduce the load on the database leading to higher performance.

2. BACKGROUND

Logically, our dynamic content web server consists of a front-end, containing the web server and the application logic, a cache, and a database back-end. The cache functions as a transparent proxy between the application logic and the database to the application logic, the cache appears as the database and to the database; the cache appears as the application server. The cache takes as its input the database queries generated by the application logic. On a read query, the cache checks whether the results of the query reside in the cache, and, if so, returns them

immediately to the application. Otherwise, it forwards the query to the database. The database returns the results of the query to the cache, where they are inserted in the cache and forwarded to the application. On an update query, the cache performs the necessary invalidations and forwards the update to the database. The cache may be located on the same machine as the front-end, on a separate machine, or on the same machine as the database. If the cache resides on a separate machine, it can be connected to the rest of the network through an L4 switch, so that the server can continue to function when the cache machine is down. There may be multiple machines executing the web server and the application logic. There may also be multiple cache machines.

3. QUERY CACHE

For the fast access the database we use the query cache. Query cache will store all record of executed query. Query cache will keep record of newly executed queries. The major goal of the query cache is to reduce the response time of query. It will increase the brainpower of data ware house so that system will memorize the latest work it has performed.[12]This memory will be used afterward for answer the result of queries which has been earlier performed by the users. The cache will maintain two state valid and invalid state. When any query submitted by the user, the cache memory is first examined to check whether requested query is already store in the cache. If the query is stored, then check the state is valid or invalid. If state is valid then data can be access and if state is invalid then data can't be access. but If user send a query of insert, update, delete and drop then data will be alter in database and state of related query will be invalid. Now invalid state data and query can't be access by user. This can save important time and improve data warehouse performance by not reevaluating the queries which are already stored in the cache. One of analyst place a query to show me the employee of a company, who working under the manager_id is 100,101,201. The query will look like as follows: -

SELECT emp_id, name, salary, manager_id FROM employees WHERE manager_id IN (100, 101, 201);

When the query is submitted, query cache will be examined to check whether this query is available or not and state is valid or invalid. If it is not available, query will be evaluated and result will be store in the query cache. The results of the query are shown in the table1 –

Table 1 – output of the above query

Emp_id	Name	Salary	Manager_id
Emp204	Ganesh	12000	100
Emp213	Ramdev	15500	201
Emp218	Mohan	13200	100
Emp222	Kishan	10690	101
Emp225	Raghuveer	14600	101

If any other user submitted the same query the result will be retrieved from query cache because that query is already stored in the cache. We will call this Query1.Let suppose another user wants to the employee of a company, whose salary greater than equal to 10000 AND manager_id is 100,101. The query will look like as follows:

SELECT emp_id, name, salary, manager_id FROM employees WHERE manager_id IN (100, 101) and Salary>=13000;

When the query is submitted, cache memory is examined. Same query is stored in the cache memory and state is valid then we can get the result of Query 2 as shown in Table 2.

Emp_id	Name	Salary	Manager_id
Emp218	Mohan	13200	100
Emp225	Raghuveer	14600	101

Now result of Query 2 will be generated from the Query 1 result set instead of going through from all the data stored in the data warehouse. This process will save lot of time and effort required to go through all the records. Queries against complex view definitions must be answered very fast because users engaged in decision support activities require fast and quick answers to their questions. Even with sophisticated optimization and evaluation techniques, there is a limit to how fast one can answer such queries. The main objective of a materialized view is to improve query performance [11]. However, when a warehouse is updated especially due to the changes of remote information sources, the materialized views must also be updated. While queries calling for up-to-date information are growing and the amount of data reflected to data warehouses has been increasing, the time window available for making the warehouse up-to-date has been shrinking. Hence, an efficient view maintenance strategy is one of the outstanding issues in the data warehouse environment. This can improve the performance of query processing by minimizing OLAP queries down time and interference. [3].

3.1 Query cache state

- Invalid –if query is not stored in query cache then state will be invalid. If data is updated by user by any these query insert, update delete the state will be invalid.

- Valid – if query is stored in query cache and not updated in database from any these query insert, update delete the state will be valid.

Our problem is that we have a query and query result stored in the cache. But if the warehouse is updated with the new data the cache query result will reflect to old data. We will create a mechanism of state; Query 1 is submitted by the user and his result is stored in the query cache. When next user submit the same query on updated data warehouse the query cache will check the state if state is invalid, it means the data warehouse is updated with new data. Now the query doesn't have to go through from all of the records. It will get the last index of the query result stored in the query cache. Then it will start searching the records which meet the query criteria from onward to that index. This can save lot of time and effort required to search the large amount of data.

4. EXPERIMENTAL EVALUATION

4.1 Hardware Platform

We use the same hardware for all machines running the client emulator, the web servers, the cache, and the database. Each machine has dual AMD Athlon 2400 processors (running at approximately 2.1 Ghz), 512MB SDRAM, and a 120 GB disk drive. All machines are connected through a switched 100 Mbps Ethernet LAN.

4.2 Software Environment

All machines run RedHat Linux 9. We use Apache Tomcat 4.1 as our Web/Application server. We use MySQL v4.0 as our database server. We increase the maximum number of Apache processes to 150. With that value, the number of Apache processes is never a limit on performance.

5. RELATED WORK

5.1 Overview of dynamic data caching

Dynamic web data can be cached at different stages in its production: the final HTML page [5, 6], intermediate HTML or XML fragments, database queries, or database tables [4]. Combinations of various caches are also possible. Intuitively, caching at the database stage typically offers higher hit ratios, while caching at the HTML or XML stage offers greater benefits in the case of a hit. There is no conclusive evidence at this point that caching at any single stage dominates the others. or instance, Labrinidis and Roussoulos use a synthetic workload and conclude that HTML page caching is superior [5], but Yagoub et al. use TPC-D and conclude that database query caching is more effective [7].It appears that the different caches are complimentary [8,7]. This paper is concerned with database query caching.Our methods can be extended to record dependencies between HTML pages or fragments and database data items,and we intend to investigate this in further work.

5.2 Non-transparent approaches

Luo et al. [2] require the database designer to specify which tables are cached. Updates to the cache are performed once a minute. As of version 4.0.1, MySQL has begun implementing a query cache alongside the database [9]. It works similarly to our design but it runs on the same machine as the database. Our method allows the cache to be placed on a different machine allowing us to reduce the load on the database machine. Oracle 9i also provides table-level caching in the middle-tier and invalidation based on time and events (database triggers), but no generalizable solution for generating invalidations [4]. Yagoub et al. [7] describe a declarative system for specifying a web site that allows a designer control over HTML, XML and query caches, including what to insert or to remove from the cache and how to invalidate or update items in the cache. Challenger etal. propose a cache API to control the contents of the API [10]. Datta et al. propose annotating the application logic to inform the cache which HTML fragments are cacheable, similar to existing web site design, and automatically maintains consistency at all times. Nonetheless, we have been able to demonstrate substantial performance benefits.

6. CONCLUSIONS

Web traffic increases, many solutions to improve web performance have been proposed. The primary solution has been the implementation of caching proxies that replicate content to sites closer to the clients. However, simple caching techniques are not adaptable for improving performance of dynamic content delivery. This is due to the difference between static and dynamic content.

Query cache will keep record of recently executed queries. Query cache will also be responsible for keeping result of recently executed queries. Query Cache technique is to store

queries and their corresponding results. If similar query is submitted by any other user the result will be obtained using cache memory. . This memory will be used afterward for answer the result of queries which has been earlier performed by the users. The cache will maintain two state valid and invalid state. When any query submitted by the user, the cache memory is first examined to check whether requested query is already store in the cache. If the query is stored, then check the state is valid or invalid. If state is valid then data can be access and if state is invalid then data can't be access. it will start searching the records which meet the query criteria from onward to that index. This can save lot of time and effort required to search the large amount of data.

11. REFERENCES

7. REFERENCES

[1] J. Challenger, P. Dantzig, and A. Iyengar. A scalable system for consistently caching dynamic web data. In Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies, New York, New York, 1999.

[2] Q. Luo, S. Krishnamurthy, C. Mohan, H. Pirahesh,H. Woo, B. G. Lindsay, and J. F. Naughton. Middle-tier database caching for e-business. In Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pages 600–611.ACM Press, 2002.

[3] Ideal Strategy to Improve Datawarehouse Performance by Fahad Sultan & Dr. Abdul Aziz. (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 02, 2010, 409-415

[4] Oracle. Oracle9i application server web caching. 2000.

[5] A. Labrinidis and N. Roussopoulos. WebView materialization. pages 367–378, 2000.

[6] K. S. Candan, W.-S. Li, Q. Luo, W.-P. Hsiung, and D. Agrawal. Enabling dynamic content caching for database-driven web sites. In Proceedings of the 2001 ACM SIGMOD international conference on Management of data, pages 532–543. ACM Press, 2001.

[7] K. Yagoub, D. Florescu, V. Issarny, and P. Valduriez.Caching strategies for data-intensive web sites. In Proceedings of the 26th International Conference onVery Large Databases, 2000.

[8] K. Rajamany. Multi-tier caching of dynamic content for database-driven web sites. PhD thesis, Rice University, Houston, Texas, 2000.

[9] MySQL. http://www.mysql.com.

[10] A. Iyengar and J. Challenger. Improving web server performance by caching dynamic data. In USENIX Symposium on Internet Technologies and Systems,1997.

[11] Efficient incremental view maintenance in data warehouses. Ki Yong Lee, Jin HyunSon, Myoung Ho Kim. Korea Advanced Institute of Science and Technology.

[12] Strategy to make superior Data ware house by Vishal Gour in International Conference on advance computing and creating entrepreneurs Feb2010.