Research Inventy: International Journal Of Engineering And Science
Vol.3, Issue 3 (June 2013), PP 55-59
Issn(e): 2278-4721, Issn(p):2319-6483, Www.Researchinventy.Com

Implementation of Techniques for Improving Web Applications Performance

[1]Eljona Proko, [2]Ilia Ninka
[1](Department of Computer Science)

ABSTRACT : *Over the last three decades the World Wide Web has undergone a continuous evolution. To respond to ever increasing demands in the field of web applications are constantly proposed standards and techniques in order to develop even more effective and high performance applications. These new technologies have made possible the emergence of a new generation of Web applications called Rich Internet Applications. Rich Internet Applications are built using successful technologies like Ajax. The purpose of this study is to measure and analyze the performance as one of the key topic in an application. Performance study of web applications is a complex process that depends on different factors. The methodology used in this study is testing Query Cache technique used mainly in MySql database. To have a clear view of the efficiency of this technique, besides the explanation of its configuration, we will make a comparison with real data, pre and post activating this technique. Testing is performed on an application developed by Ajax technology. The results of experiments shown that in sections with more query to be executed is a significant difference between activating and not activating the Query Cache. We conclude in this study that Query Cache it is a useful technique recommended for use in all web applications where as back-end is used MySql.*

KEYWORDS: *Ajax, performance, technique, web application*

I. INTRODUCTION

Web platforms offers big advantages for business development, communications, data transmitting in real time, but it has its problems according to data security, continuity, performance, and other issues. Performance [1] is one of the key topic of a application that needs studying carefully on the beginning of the project and during its development. Performance study of web applications is a complex process that depends on different factors and the solutions offered today by the specialists and different companies specialized in the estimation of applications, are not uniform for all products. It is related with the kind of the used platform, database, application typology, etc. Even inside the same used platform, there are many techniques used for the improvement of the platform. One of these techniques is the Query Cache [2] mostly used in MySql [3] database. In order to have a clear view of the efficiency of this technique, besides the explanation of its configuration, we will make a confrontation with real data, pre and post activating this technique. For this reason we will use an application used in the bank system M.I.S. This paper begins by present a view of Rich Internet Applications. Section 3 describes experimental methodology, to continue with results in Section 4. We conclude this study in Section 5.

II. RICH INTERNET APPLICATION

In its beginning World Wide Web was a platform for accessing static or dynamic content encoded in hypertext markup language. User interaction was limited to navigating links and entering data in forms. Modern Web solutions resemble desktop applications, enabling sophisticated user interactions, client-side processing, asynchronous communications, and multimedia. An important role nowadays play rich internet applications (RIAs). The term RIA refers to a heterogeneous family of solutions, characterized by a common goal of adding new capabilities to the conventional hypertext-based Web. RIAs combine the Web‘s [4] lightweight distribution architecture with desktop applications [7] interface interactivity and computation power, and the resulting combination improves all the elements of a Web application (data, business logic, communication, and presentation). Rich internet applications are developed using Web 2.0 techniques and technologies, such as Ajax [5], or Ajax platforms such as ASP.NET Ajax [6], or non Ajax platforms such as Microsoft Silverlight, Adobe AiR, Adobe Flex, etc. Using RIA technologies creates users convenience and interaction with web applications. But it all comes at a cost, which means the use of RIA raises issues involving language and architectural standards that are used to develop these applications. A specific crucial issue for RIAs is that of finding suitable approaches and technologies for supporting all the software lifecycle activities effectively, and the maintenance and testing activities above all. The relevance, complexity, expensiveness and criticalities of software maintenance processes are well known for any type of software application. However, these problems are even more relevant in the context of RIAs, since these applications are usually developed in short times by programmers that don‘t use well known practices of Software Engineering, and often use frameworks and tools that, on the one hand simplify RIAs‘ development, on the other hand produce complex code that is difficult to understand, at the expense of the quality of the final product. Moreover, both the asynchronous and the heterogeneous nature of the RIAs, which are developed by means of several technologies and are based on a client-server [7] architectural model in which the communication between the client and server may be asynchronous, make the RIAs [10] harder to comprehend and consequently difficult to maintain and test [8].

III. EXPERIMENTAL METHODOLOGY

M.I.S is a program used in the bank system, specialized in NPL sector (non performing loan). This program is not just used as a store of data, but helps the operator bring results in real time, give predictions based on events happened before. This program, also, communicates in all compatibility with other programs changing data in all database known formats. Assuming that the bank system is based in the AS400 system, MIS is an innovation for this system that combines the data bank administration model using new technology. Globalization and the changing of the way of working accelerated this application. MIS is successfully used in the evaluating and administrating of bad credits process. This platform today is used for the due diligence process from banks with international reputation such as Deutsche Bank or the Financial Institute Varde Group and Centrale Attività Finaziarie (CAF). The aim of this study is not promoting MIS as a product but just give us an understanding of the complexity and power of this application and that the used techniques have resulted successful and we wish to share this with the reader for guide purposes in comparable applications. Reliable and powerful, M.I.S can host millions of records, and is fully scalable on demand. It can import and export records in different format therefore can cover all monitoring activity along with processing, statistic analysis.
For this application is using the LAMP [9] platform where its basic components are:

- GNU/Linux: Operating system
- Apache: Web server
- MySQL: Database server
- PHP: Script language

Other advanced technologies used in this platform are XHTML, AJAX, ect.

In this section we will explain the use of query cache techniques, bringing real data in confrontation measurements in both system states. In order to have a real confrontation, measurements are made in the same internet line and the same browser (Google Chrome), with the same users connected. As mentioned before, MySQL is used as backend for the storage and management of contents of our software, most used query is SELECT type and the operation of reading, updating, deleting are minor of SELECT operation. In MIS is not used any kind of caching of data, so the database MySQL is interrogated any time a user request a page. At beginning MIS administrated a small number of data and users, so the performance was very appropriate. As the number of users and data grew up, we decided to use the function Query Cache that let a caching of data directly in MySQL server. Query Cache is very simple to use and above all doesn’t need any modification of PHP code. The Query Cache save the output of the first SELECT query inside the space reserved for caching of query in RAM, but we emphasis that only SELECT query can cached. This function helps using the same results for many users, the application search if the wanted result is in cache. If it is found, it is shown instantly, either way the query will run. The schedule of cache generation is decided in base of criticism and frequency of data changing. This technique may be used in the interested area or in all application. The schedulation and the activation for different area are not the same for all the applications, but the are decided by the analyses of IT department and users experience. Using this technique, we avoid running the same query in the same table and minimizing I/O operation on the disc. In case of running query that modification the content of the table (UPDATE, INSERT, DELETE, ALTER) all the saved queries/cache data will be deleted to let the new update. The Query Cache helps the server to manage the load of data without care the data validation. Query Cache is a function offered by MySQL database. By default, this function is set to off.
To activate first check if the query cache is active:

```
mysql> SHOW VARIABLES LIKE 'have_query_cache';
```

When set query_cache_size to a nonzero value, the query cache needs a minimum size of about 40KB to allocate its structures. (The exact size depends on system architecture.)

Then be activated by this command:

```
 mysql> SET GLOBAL query_cache_size = XXXXXXX ;
```

The following are also part of the code used in the procedure that manages:

QueryCacheManager.class.php

```
………………………………………..

function getCacheTableForSql($dp,$sql,$index_fields=array(),$seconds=60,$engine='MyISAM') {
        self::purgeCache(getFresh()->userSetting('FRESH_SKIP_QUERYCACHE')==1);
        $db = $dp->getDbName();
        $cacheDp = self::_getCacheDp();
        $cacheDb = $cacheDp->getDbName();
        $cacheTableDb = self::_getCacheTableDb();

        $id = sprintf("%u", crc32(md5($sql).md5($db)));

        if($cacheDp->fetchSQLValue("SELECT COUNT(id) FROM fresh_query_cache WHERE id=$id")==1) {
           return "`$cacheTableDb`.".self::_getCacheName($id);
        }
        //table does not yet exist: create and populate id
        self::_insertCache($dp,$id,$sql,$seconds,$engine,$index_fields);
        return "`$cacheTableDb`.".self::_getCacheName($id);
```

$dp-------------- provider data
$sql ----------- Query
$index_fields=array() ----------- array fileds that will be indexed after the execution
$seconds=60 lifetime of the Cache
$engine='MyISAM' --------------- storage type that is used
Then through IF function check if there is a cache for that query.

IV. RESULTS

In this section we present the results obtained from the tests. We conducted two tests, once with Query Cache activate and once non activate. We used two database with 500 records, and with 200000 records. Table 1 shows the execution time in several application areas using 500 records and in both situations, Query Cache active and inactive. In Table 2 we present the results obtained from testing conducted with 200000 records. Chart results are given in Fig. 1 and Fig. 2.

Area	Cache (Yes)	Cache (No)
FRMWarnings	0.0063	0.0076
TopHeader	0.085	0.0923
ChooseNDG	0.1534	0.1861
ActivityTimeline	1.393	3.9213
Home	0.1693	1.3234
Notifications	0.1271	0.5766

Table 1. Execution time in seconds in some areas of application

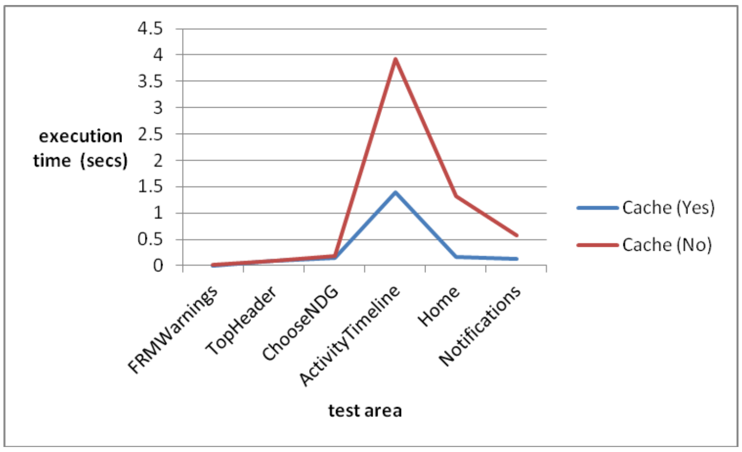

Fig.1 Graphic test results for 500 records

Area	Cache (Yes)	Cache (No)
FRMWarnings	0.0046	0.0082
TopHeader	0.0591	0.1252
ChooseNDG	0.1613	0.3032
ActivityTimeline	14.2095	44.6418
Home	3.2402	0.1447
BorrowerResultsLister	2.7635	7.821
NDGedit	0.292	1.8744
ResolutionsStrats	34.506	96.053
BpStrats	0.6014	2.4047
FRMTaskLister	1.3607	4.7414
Notifications	0.1583	5.7257
Watchlist	0.8517	2.7389
PropertiesLister	0.0379	5.8394

Table 2. Execution time in seconds in some areas of application

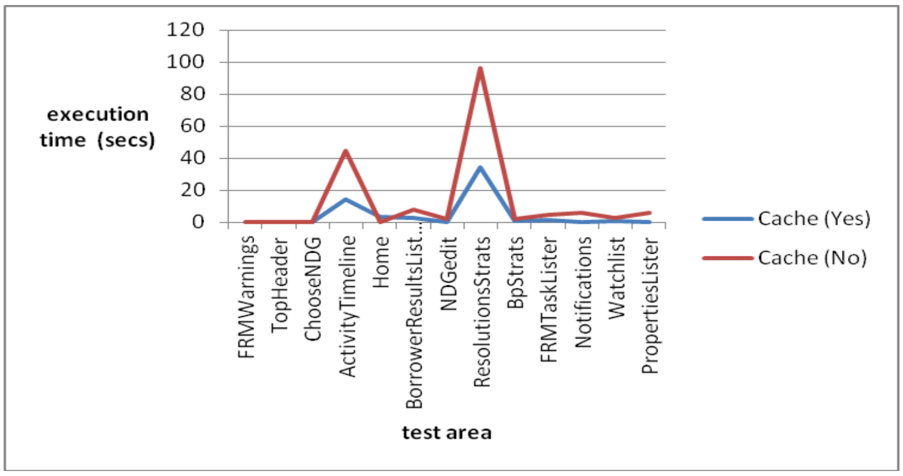

Fig.2 Graphic test results for 200000 records

V. CONCLUSION

Based on long experience in the design and administration of web applications and the results of measurements made we conclude that QueryCache is a useful technique and recommended for use in all applications where as back-end is used MySql. This technique improves application performance by 200%. Measurements have shown that more sections Query to be executed there a significant difference between activated and non-activated QueryCache, while those with less Query is small change. Advised that for the activation of this technique used a personalized configuration rather than using its default configuration.
As a part of our future work we will discuss another technique to improve the performance, which is Indexation.

REFERENCES

[1] Liu, H. (2009). Software performance and scalability a quantitative approach, Wiley series on Quantitative Software Engineering.

[2] Jim Challenger, Arun Iyengar, and Paul Dantzig. A Scalable System for Consis-tently Caching Dynamic Web Data. Proc. IEEE INFOCOM 99.

[3] MySQL AB. MySQL: The World's Most Popular Open Source Database. www.mysql.org. 2005.

[4] Murugesan, S. *"Understanding Web 2.0," IT Professional , vol.9, no.4, pp.34-41, July-Aug. 2007*

[5] J. Garrett, —AJAX: A new approach to Web applications, Adaptive Path, 2005

[6] ASP.NET Ajax: http://www.asp.net/ajax

[7] Meli , S.; G mez, J.; P rez, S.; D az, O.; , *"Architectural and Technological Variability in Rich Internet Applications," Internet Computing, IEEE , vol.14, no.3, pp.24-32, May-June 2010*

[8] G.A. Di Lucca, A.R. Fasolino, —*Testing Web-Based Applications: the State of the Art and Future Trends, Information and Software Technology Journal, Vol. 48, Issue 12, Pages: 1172-1186 (December 2006), Elsevier inc.*

[9] Dougherty D., LAMP: The Open Source Web Platform. *www.onlamp.com/pub/a/onlamp/2001/0 1/25/lamp.html, 2001 as accessed on May 2011.*

[10] Fraternali, Piero; Rossi, Gustavo; Sánchez-Figueroa, Fernando; , *"Rich Internet Applications," Internet Computing, IEEE , vol.14, no.3, pp.9-12, May-June 2010*