ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

A Novel In-memory Caching Technique for the Performance Optimisation of Web Services

Sidharth S Prakash, Visakha K

M.Tech in Software Systems, Department of I.T, School of Engineering, CUSAT, Kochi, India

ABSTRACT: Web applications are an essential part of today's cyber-centric world. Through the internet domain they provide services to their clients from a remote web server. Hence the quality of the service experienced by the clients depends on the web server performance metrics such as speed and response time. This paper presents a specific methodology to optimise the performance of a web application from the perspectives of speed and response time. The proposed method uses in-memory caching technique, which aims to boost the speed of database retrieval utilising an in-memory cached data set. This ensures an optimal response time from the server even in processing a heavy task such as a multiple SQL join operation. The paper presents the performance evaluation of in-memory caching technique in different scenarios by means of statistical analysis tools.

KEYWORDS: Web application; Performance optimization; In-memory caching; Benchmarking; Statistical analysis

I. Introduction

Today internet is omnipresent and hence a governmental organization, a company or the government itself may provide their services to the people in the form of web applications through the internet. Most of the clients access these services hosted in the internet with the help of a web browser in their personal computers or mobile device. Web application is a client-server application where the server side runs on a remote web server hosted in the internet whereas the client side runs on the web browser of the client.

The performance of a web server is one of the most important factors behind the success of a web application [1]. It is a very broad term that encompasses the speed, utility, usability and security of an application. This paper focuses on the aspect of speed and proposes a specific methodology to achieve it. This technique is the utilisation of in-memory caching which aims to improve the speed of database retrieval process using a cached data set [2].

The database operations are one bottleneck that web applications possess in terms of speed [3]. Especially heavyweight queries with many SQL join operations can take considerable amount of time to process and produce results. The reads from the database can be speeded up using the concept of caching where the heavyweight query results are cached. When a database read is requested by the client, these cached values are used rather than re-executing the queries again.

The paper is organised as follows. The first section deals with in-memory caching concept and the algorithm employed for database fetch request in a cached environment. The experiment methodology section follows the former section giving brief description on the tools and methodologies applied in the tests. Analysis and observation section analyses the statistical results obtained during the experimentation procedure. The paper concludes by drawing inferences from the analysis and observation of statistical results obtained during the experiment.

II. In-Memory Caching

Database normalisation is the process of eliminating the data redundancy through specific organisation of data in form of multiple tables in the database. Thus a database table alone cannot produce the complete details about a database entity and requires SQL join operations for this purpose. Clients usually communicate with the web server through an application programming interface (API) which produces a structured reply for every valid request by fetching data from the multiple database tables.

If the basic details about a database entity are cached, the web server can respond to the requests using the values fetched directly from the cache. Caches are storage spaces other than the hard disk which aims at high speed retrieval of data. The caches work on the principle of "locality of reference" which states that the data set once accessed, has a higher probability of access at a later stage [4]. Database caching aims at caching the results of heavy weight SQL queries, so that the web server need not re-execute the query during a later request but rather fetch the results from the cache. This process considerably increases the speed of the web server in responding to a user query [5]

A. *In-memory Caching Algorithm:*

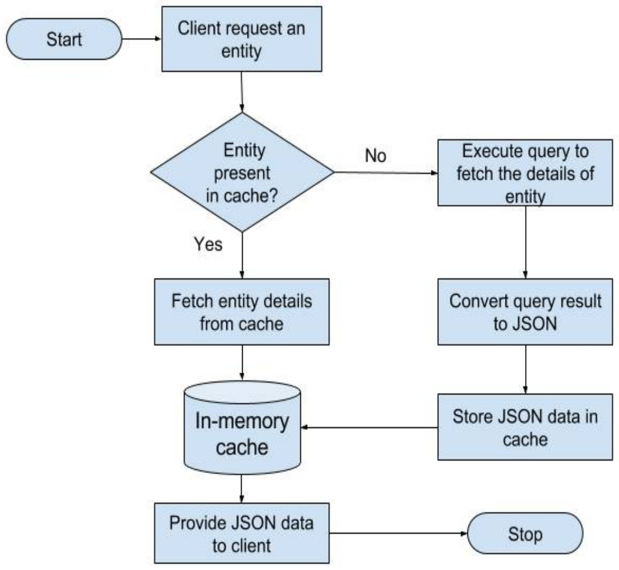
Start
Client request an entity
Entity present in cache?
No
Execute query to fetch the details of entity
Yes
Fetch entity details from cache
Convert query result to JSON
In-memory cache
Store JSON data in cache
Provide JSON data to client
Stop

Fig. 1. Algorithm to fetch entity details in a cached environment

The database cache can be implemented using an in-memory key-value storage mechanism using the algorithm presented in Fig. 1. When a client requests for a specific data set, the web server first checks the in-memory cache store using the entity key. If the requested data set is absent, the server executes the SQL query. The query result is converted to JavaScript Object Notation (JSON) and stored as the value of the requested entity key. If the requested data is present in the cache, it is directly fetched and provided to the client. The cached data will become stale when the actual source of data is updated. Hence an asynchronous task runner can be employed which updates the cache whenever the source data is updated.

III. Experiment Methodology

The web applications are deployed on web servers to serve each client's requests and hence to analyse the web server performance, the client request traffic load must be simulated. Apache benchmark is a tool for HTTP server benchmarking which produces test results for analysing the server performance in various scenarios [6]. Tool provides the facility to simulate web traffic with specifications like the number of requests and number of concurrent requests.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

The test scenarios are simulated using this benchmarking tool and the server performance in serving each request can be saved into a file for further statistical analysis.

R is a programming language for the statistical analysis of data sets [7]. The result of apache benchmark for a specific test scenario saved in a file must be analysed relative to other relevant test scenario results. Hence the benchmarking results to be compared relatively are fed into R for their statistical analysis. R can provide visual representation and easier data analysis through graphical plots.

In-memory caching is implemented using in-memory database called Redis for the experimentation [8]. There are other choices available like Memcached, but Redis is the most popular in-memory key value store for commercial application development because it can implement persistent data storage.

The test scenarios for analysing the effect of in-memory caching and background task runners are as follows. The performance of web application with in-memory caching and without caching are evaluated against the heaviness of the SQL query statement and the number of concurrent client requests. Relatively heavier queries are created using more number of SQL join statements so that the queries becomes more processor intensive and takes more time to complete execution. To obtain uniform network conditions, all the optimisation tests were done on uniform bandwidth.

IV. ANALYSIS AND OBSERVATIONS

This section is divided into two sections. Each section explains the observations derived from the plots created through R using the benchmarking results for the test scenarios of in-memory caching.

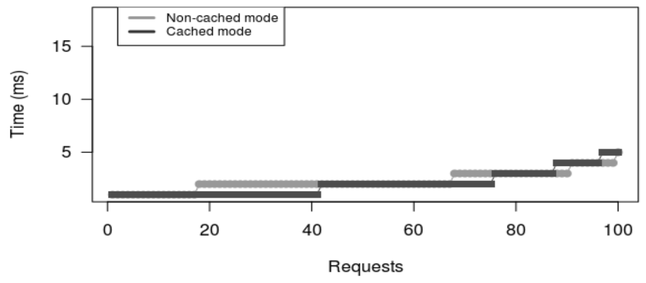

Fig. 2. Time taken for executing light weight query

A simple query with no join operations is chosen as a light weight query for experimentation. It is evident from Fig. 2. that the cached performance and non-cached performance is almost the same in the case of light weight query execution.

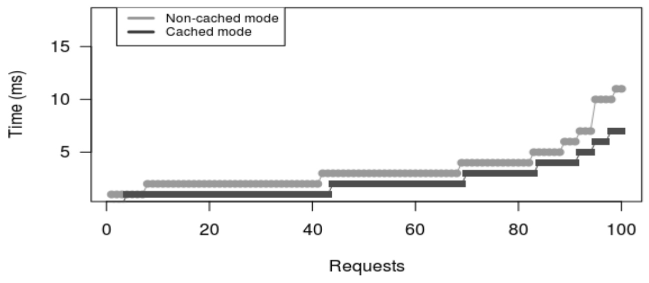

Fig. 3. Time taken for executing medium weight query

The Fig. 3. provides an impression that the cached performance is slightly better than the non-cached performance in case of medium weight query created with four join operations.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

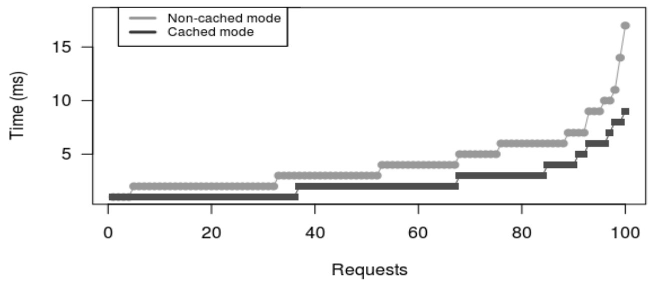

Fig. 4. Time taken for executing heavy weight query

A significant boost in performance of heavy weight query processing is noted in the Fig. 4. for cached implementation than in the case of non-cached implementation. A query with eight join operations was selected as the heavy weight query for testing.

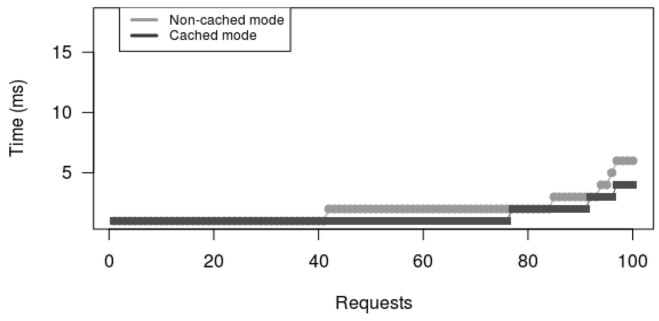

Fig. 5. Time taken for medium weight query for 5 concurrent requests

The overlap between the two experimentation plots in the beginning stage is clear from Fig. 5. But there exist a slight performance improvement using the cached mode of implementation towards the end of request phase when the number of concurrent users is 5.

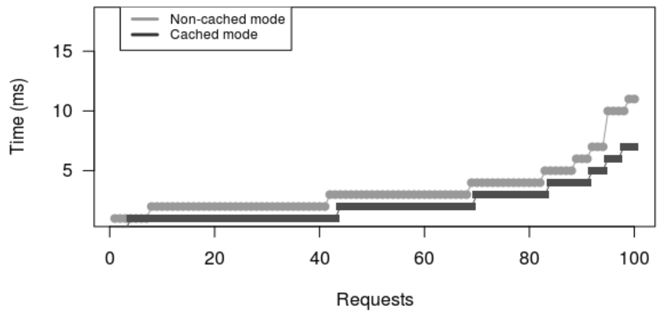

Fig. 6. Time taken for medium weight query for 10 concurrent requests

A satisfactory performance improvement can be noted from the Fig. 6. for the cached implementation than in the absence of caching when 10 users hit the server simultaneously.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

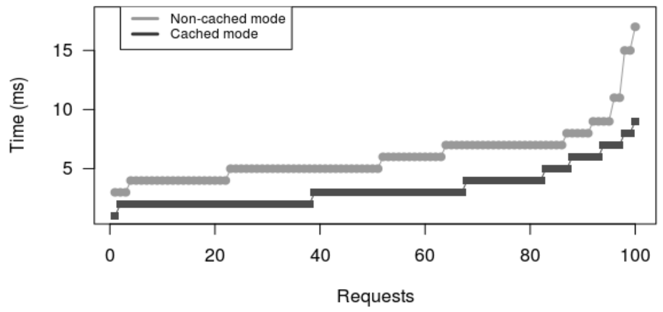

Fig. 7. Time taken for medium weight query for 15 concurrent requests

There is a significant boost in the performance when tested with 15 concurrent users for the cached implementation over the non-cached implementation as observed from the Fig. 7.

V. CONCLUSION

Web application performance can be improved by certain design choices. But the optimisations employed will not give satisfactory results in every scenario. In the case of in-memory caching, this design choice will produce a performance boost only when there exists heavy weight queries in the application which are processor intensive and time consuming. When the results of heavy weight queries are cached, the web server can easy use this cached data set to serve the client requests. Moreover, higher the number of concurrent requests, higher is the reduction in the time taken to serve the requests. The performance of in-memory caching depends on the RAM available on the server. Additionally if data persistence is employed, the hard disk space available on the server also plays a role in the success of optimisation.

Hence it is vital to analyse and understand the traffic a web application will be subjected to and other specific requirements so as to employ the best performance optimisation technique.

REFERENCES

1. Zhu, L. Z., and Li, F., “The Collection and Analysis of Performance Parameters in Web applications”, International Conference on Electronics and Optoelectronics (ICEOE), pp. 150-154, July 2011.
2. Sivasubramanian, S., Pierre, G., Steen, M.V., and Alonso, G., “Analysis of Caching and Replication Strategies for Web Applications”, IEEE Internet Computing, Vol. 11, pp. 60-66, January 2007.
3. Atserias, A., Grohe, M., and Marx, D., “Size Bounds and Query Plans for Relational Joins”, IEEE 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 739-748, October 2008.
4. Karakostas, G., and Serpanos, D. N., “Exploitation of different types of locality for Web caches”, Seventh International Symposium on Computers and Communications, pp. 207-212, 2002.
5. Yan, J., Chen, J., and Jiang, W., “Data Caching Techniques in Web Application”, Enterprise Systems Conference, pp. 289-293, August 2014.
6. Olson, M., Christensen, K., Lee, S., and Yun, J., “Hybrid web server: Traffic analysis and prototype”, IEEE 36th Conference on Local Computer Networks (LCN), pp. 131-134, October 2011.
7. Voulgaropoulou, S., Spanos, G., and Angelis, L., “Analyzing Measurements of the R Statistical Open Source Software”, 35th Annual IEEE Software Engineering Workshop (SEW), pp. 1-10, October 2012.
8. Stjepanovic, D., Savic, M., Jokić, J., and Marić S., “Performance measurements of some aspects of multi-threaded access to key-value stores”, 23rd Telecommunications Forum Telfor (TELFOR), pp. 831-834, November 2015.