Journal of Theoretical and Applied Information Technology
30th June 2013. Vol. 52 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPROVING WEB SERVER PERFORMANCE USING TWO-TIERED WEB CACHING

[1]FAIRUZ S. MAHAD, [2]WAN M.N. WAN-KADIR
Software Engineering Department, Faculty of Computer Science & Information Systems,
University Teknologi Malaysia
E-mail: [1]fsafwan88@gmail.com , [2]wnasir@cs.utm.my

ABSTRACT

The purpose of this study is to improve the web server performance. Bottlenecks such as network traffic overload and congested web server has still yet to be solved due to the increasing of internet usage. Caching is one of the popular and efficient solutions to reduce network latency. However, there is a need to study the current caching technique and further optimize the result. Therefore, this study proposed a Two-Tiered Caching System (2TCS). The 2TCS adapts a selective caching system concept which contains two tiers in its caching architecture. The 2TCS is used to further increase the probability of achieving a cache hit on top of utilizing the normal caching system. The proposed technique utilizes SQUID as its web cache server and also Mozilla Firefox's default caching system at the client side. A performance comparison of a normal caching system against the implementation of the 2TCS is made using the IBM Rational Performance Tester to prove the result.

Keywords: *Web cache, Two-tiered caching, Web Server Performance, Software Engineering*

1. INTRODUCTION

The web servers have been serving on the World Wide Web for years. It ensures client's requests are responded as fast as possible. However, the performance issue of a web server still remains unsolved. Despite installing high-end networking related hardware, it still remains insufficient to accommodate the ever increasing of bandwidth usage by users. Currently, performance still caused bottlenecks in the network such as congested networks which leads to the increase in network latency thus resulting in the increased average response delay among users. This states the importance of measuring the throughput of a web server[1].

Solving congested network problems by installing state-of-the-art hardware or increasing the bandwidth would only be a temporary solution. It is almost impossible to accommodate the bandwidth demand of users which tends to increase rapidly over time. Therefore, solving congested networks with upgrading hardware is not an efficient and effective solution.

The term web server refers to software which receives requests from clients, process and responds them accordingly [2]. The performance of a web server can be defined as a measurement to determine the effectiveness of web servers in responding to the client's requests. On the other hand, the degree of performance concerns with the capability of a web server. There are many research papers, journals and articles that address the performance issues of a web server. Performance is a vital issue among web servers. In order to ensure an efficient web server, performance is essential [3]. Other researches also state that the web server performance is a vital matter especially for websites that service a large amount of requests [4]. Due to the low web server performance, users doubt the reliability of the web application despite the useful features it presents.

There are known solutions to solve issues regarding the performance of a web server of which popular ones are such as caching, load balancing and memory compression. Among them, caching is the most widely used technique which is also seen as effective and efficient in increasing the web server performance. Web cache is a technique which stores copies of elements in a web page of which could consists of images, scripts, video and etc. These copies will be used to serve future similar requests which prevent the requests from travelling all the way to the origin server thus able to save bandwidth while reducing the degree of network congestion.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

This study focuses on improving the web server performance by utilizing the existing cache system available in Mozilla Firefox in conjunction with SQUID, a web cache server. The study proposes a web cache architecture which is able to further increase the probability of requests being served at the client-side rather than the origin server. The performances of the existing cache system are recorded using IBM Rational Performance Tester. The recorded data will be plotted against the data with the implementation of the proposed technique.

2. LITERATURE REVIEW

Web caching is common nowadays and many researches have acknowledged its effectiveness in reducing bandwidth usage. There are other techniques which are basically load-balancing based technique of which works by analyzing the relationship's dependency followed by adjusting the weight among each part of the session to dynamically manipulate the initiation of the session [5] and also another by clustering two components to allow sharing of information between components [6]. Apart from load balancing, there are also others such as main memory compression which utilizes a reserved space of a physical memory used to store compressed data allowing accesses to the disk being mitigated for other application to run [7]. There is also a technique which focuses on detecting dead requests made by users and eliminating them to avoid deadlock [8]. However, among all of these techniques, web caching is seen to be the most efficient.

As mentioned, web caching is the process of storing copies of web page elements to be used for future requests. However, web cache also has its downside. Storing saved data requires storage space. Allocating a large amount of storage space for caching would be inconvenient. Therefore, there are numerous methods and algorithms that are proposed by researches to eliminate chosen stored cached data to give way to newer cache entries while still providing a fast respond service. Meanwhile some researches do not entirely delete the cached data but creatively proposed methods to manipulate them effectively.

A. Caching Dynamic Data

One of the factors that majorly reduce the performance of a web server is the presence of dynamic web pages. Unlike static web pages, dynamic web pages really affect the web server's performance. It is impossible to avoid dynamic web pages all at once as those are the frequently used among most web sites nowadays compared to static web pages. In their paper, caching dynamic data, Iyengar and Challenger develops a cache architecture which comprises of a cache manager and several caches and application program. The cache manager acts as a medium allowing communication between one or more cache and the application program. The application program will communicate with the cache manager to either add or delete data from the cache. As a result, the performance boosted at least 58% in a worst case scenario [9].

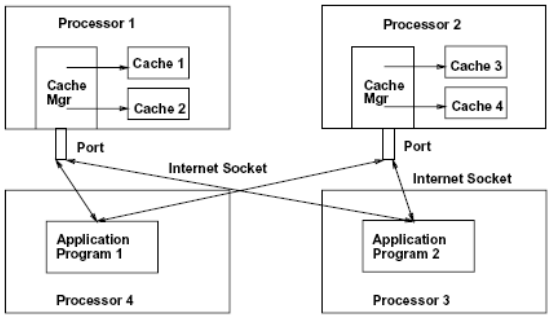

Figure 1. Cache Manager and Application Communication Over Internet Sockets [9]

The technique developed by Iyengar and Challenger proves to be able to improve the performance of the web server. However, there are still room for improvements. Their technique works well only with dynamic data.

B. Delayed Caching

Daesung and Kim proposed a technique which they termed delayed caching. Their delayed caching technique is said to be able to improve the overall system reliability [10]. The technique is implemented on a SQUID server. The performance of the SQUID server is measured using an open source software known as the POLYGRAPH which is especially used for testing cache server's performance.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
extern Boolean store_Ignore = FALSE;
extern int q_Count = 0;
extern storeURLData *pURLData = NULL;
…
if (storeURLData not NULL &&
   store_Ignore == FALSE)
   • request to web server again using URL stored
     in storeURLData;
if (Cache Miss) {
   if (++q_Count <= Q_THRESHOLD
      && store_Ignore == FALSE)
      Q_push(DiskQ, storeWriteData);
   else {
      Q_Count--;
      Store_Ignore = TRUE;
      • create storeURLData structure and store
        meta information on it;
   }
}
```

Figure 2. Delayed caching Algorithm [10]

The study proposed an algorithm which serves requests in a unique way. The algorithm is created so that SQUID does not cache any data when the request rate is high instead the meta information of the requested data is recorded for the meantime. When the request rate goes back to normal, the delayed cache will retrieve the data from the meta information initially recorded. This is how the delayed caching technique basically works. However, certain users will experience certain delays but in return, most users will benefit from the proposed technique [10].

3. RESEARCH METHODOLOGY

A. First Stage Development

According to the Iyengar and Challenger, in their paper, improving web server performance by caching dynamic data, they develop a cache architecture that consists of a cache manager [9]. The purpose of the cache manager that they develop is to manage the storage of one or more caches. In addition the application will interact with the cache manager to either delete or add items in a cache. Figure 3 below represents the cache architecture that Iyengar and Challenger propose.

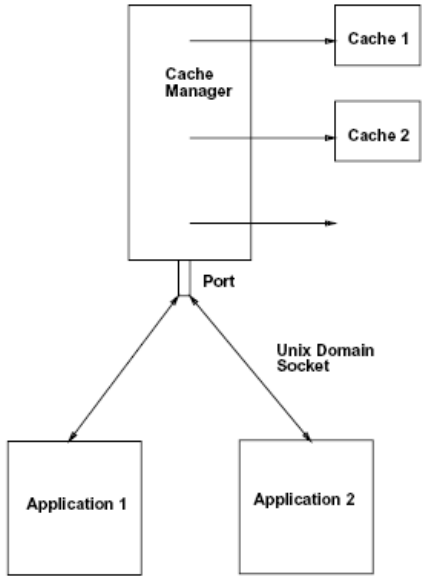

Figure 3. Dynamic caching Architecture [9]

The concept of the cache manager developed by Iyengar and Challenger will be adapted by the proposed technique of this paper. The Cache Manager that will be adapted will still serve the same purpose as Iyengar and Challenger's Cache Manager which is to manage one or more cache storage, however, the proposed technique's Cache Manager will interact with only SQUID instead of many applications. On top of that, the Cache Manager will only allow adding cache items to the one or many cache storage without able to delete it. The one or many cache storage is referred to as the client's Firefox cache directory.

In Iyengar and Challenger's cache manager, the cache manager concerns one or more cache storages, it applies the same concept to the proposed technique's Cache Manager. In addition, SQUID is used as a base for the Cache Manager to perform its functions. SQUID also has its own cache function which will be utilized by the Cache Manager. On top of that, SQUID increases the probability of acquiring a cache hit as SQUID caches data of the entire client's request passing through. However, the focus of the proposed technique is to evaluate the improvement of the web server performance. Allowing only SQUID to perform its function does not prove any effort on further improving the web server performance with respect to the proposed technique. The Cache Manager has to work in conjunction with SQUID to optimize the web server performance. In addition to the applied concept, the Cache Manager is made up

Journal of Theoretical and Applied Information Technology
30th June 2013. Vol. 52 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of an algorithm to select the top thirty cache item with the highest cache hit rate which will then be pushed to the cache storage referred to as the client's Firefox cache directory. In this way, the probability of responds to a request achieving a cache hit will be increase dramatically. Figure 4 below demonstrates the improved cache manager.

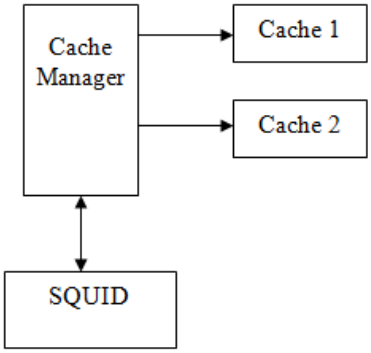

Figure 4. Improved Cache Manager

B. Second Stage Development

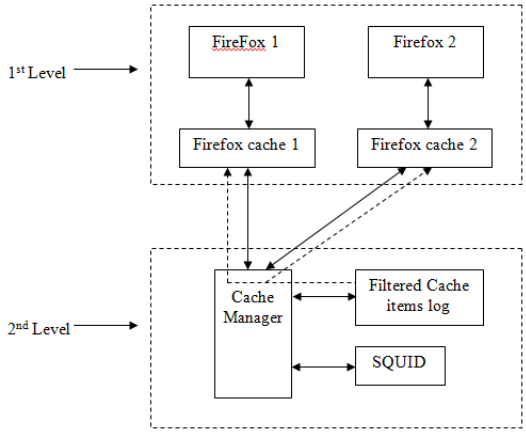

Figure 5. Additional Cache Level

The improved cache manager offers a higher probability in achieving a cache hit rate. However, there is still room for improvements which could further increase the probability of achieving a cache hit rate. By simply adding another level of cache which contains a pre-selected cached data considering only the highest cache hit rate, the probability of achieving a cache hit rate can be further improved. Improving this probability also means improving the web server performance because it prevents the request to reach the origin web server as much as possible which reduces the usage of bandwidth and also the degree of network congestion.

This additional level of cache will contain an existing common caching system available at all computers which is the web cache. The web cache is referred to as the client's Firefox cache directory. A web cache is stored on the client's computer itself and Firefox itself will cache the data and stores it in the web cache. This additional level is generally the reusing of existing caching system to maximize the effectiveness of the proposed technique in this paper.

As mentioned earlier, this additional level of cache contains cached data which are pre-selected with only the highest cache hit rate. These pre-selected cached data are actually fetched from the Cache Manager. With the Cache Manager providing the client's Firefox cache directory with selected cache items with high cache hit rate, it increases the probability of acquiring a cache hit over at the client side instead of having to travel to the origin web server. This however describes the communication between the two levels.

4. TWO-TIERED WEB CACHING

A. Overview

This study proposes a two-tiered caching system or in short called the 2TCS. In brief, the two-tiered caching system is made up of two tiers. It is termed two-tiered due to the active correlation between both tiers in performing the caching process. The core tier lies within the second tier. It serves as the major tier that manipulates and decides what to cache in both tiers. The first tier receives instructions from the second tier. In short, the first tier is dependent on the second tier. The 2TCS adapts the fusion of two concepts namely the proxy server and the selective caching concept. The similarities between the proxy server and the 2TCS can be seen in the first few processes. The later processes are where the selective caching concept takes place. The 2TCS implements SQUID, a well-known web cache server and also a Cache Manager developed using Java. A cache item with a high cache hit rate also means that the cache item is popular as it is frequently requested by clients.

The 2TCS as mentioned consists of two tiers. The first tier contains the clients in which this study refers to as the web browser. The clients have their own caching system which is embedded in their system. For instance Mozilla Firefox and Internet Explorer are popular web browsers which have their own embedded caching system to provide a faster response time to the users. The first tier

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

exploits the presence of the client's caching system. In this study, Mozilla Firefox is used and referred to as the client's web browser. The second tier consists of the Cache Manager, SQUID and also the Filtered Cache Data Folder which contains the FCD.log. The Cache Manager is responsible to utilize components available in SQUID to be able to calculate and select the top thirty cache items with the highest cache hit rate and forward the information to the FCD.log while the selected cached items will be forwarded and appended to the client's Firefox cache directory. Figure 6 illustrates the two-t**iered web caching architecture.**

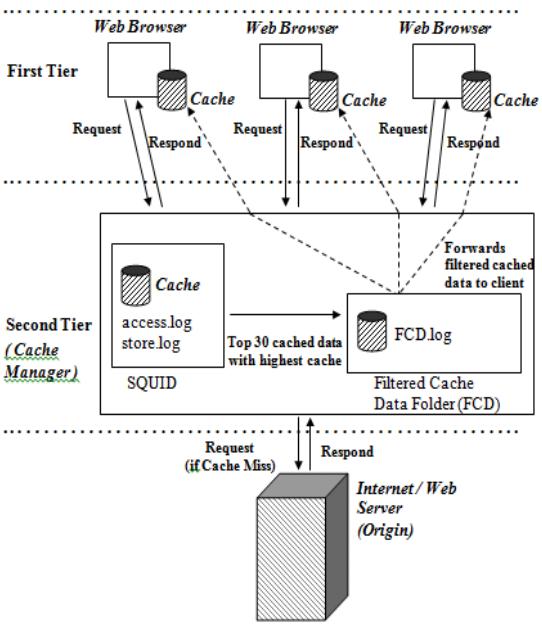

Figure 6. Two-Tiered Web Caching Architecture

This explains briefly the two-tiered caching technique. Whenever a client requests a web page, the browser cache will be visited first. Given if there is a cache hit, the response time is greatly reduced as clients will receive the responds much faster since the request is fulfilled at the client side. Should the scenario depicts a cache miss, the request will be forwarded to SQUID the web server. If a cache hit is achieved, the response time is still greatly reduced though it is not as fast as compared to if the cache hit occurs at the Firefox cache. On the other hand, should a cache miss still occur, the request will be forwarded to the origin web server. When the origin web server responds to the request, the respond will travel through SQUID. The cache manager will utilize the information stored by SQUID throughout the request and respond session of all clients and process the information to retrieve the top thirty cache items with the highest cache hit rate. In brief, a request has to visit two cache locations before it reaches the origin server should the requests still cannot be fulfilled.

B. Cache Manager

The second tier is where the core process performs and where the Cache Manager resides. The Cache Manager is developed using Java. The Cache Manager performs most of the core operations throughout the 2TCS. The Cache Manager is responsible for interacting with SQUID by retrieving SQUID's log files and compute the retrieved data based on the algorithm developed.

The log files consist of the access log and the store log. Both the log files will be fully utilized by the Cache Manager. Generally, the Cache Manager listens from the communication between all the clients and SQUID and extracts both the access and cache information. Based on this information, the Cache Manager will determine the top thirty cache item with the highest cache hit rate. Since SQUID does not involve performing any statistic value, the Cache Manager is developed to utilize the access log information to be able to calculate the cache hit rate of each cache item and translates all of those data into percentage. Upon acquiring the percentage, the top thirty cache item with the highest cache hit rate can be selected. The selected cache items data will then be stored in the FCD.log. Once the FCD.log is filled with the selected cached items, the Cache Manager will continue its operation to locate the location of the selected cached items. Utilizing SQUID's store log, the Cache Manager will read its format and extract the location of all the selected top thirty cached items. However, at certain times, not all selected cache items can be found in SQUID's default cache folder. It is due to the fact that at certain times, SQUID will cache an item and store it temporarily in the memory and not the disk.

The Cache Manager is only able to locate the location of the cached item which is stored on the disk and not in the memory of the web server. Despite acquiring the top thirty cached items with the highest cache hit rate, occasionally not all thirty cached items can be located. Once the Cache Manager has located the cached item, it will be copied and appended to Firefox cache directory. At this point, the Firefox cache directory will contain the client's individual cached items cached by Firefox as well as the Cache Manager's selected cached item based on SQUID. All clients whom

Journal of Theoretical and Applied Information Technology
30th June 2013. Vol. 52 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

open a connection with SQUID will receive the same set of selected cache items by the Cache Manager. This way, it increases the probability of acquiring more cache hit at the client side given that the Cache Manager's selected cache items are of the top thirty cache items with the highest cache hit rate based on all the client's requests. The Cache Manager will run continuously on the web server side and keeps the FCD.log update with the top thirty highest cached items. The figure below illustrates the algorithm of the Cache Manager. Figure 7 shows the algorithm of the cache manager which is the core code for the two-tiered web caching architecture.

```
While (access.log != null) {
If (urlArray != new URL ) {
urlArray = new URL;
}
else {
frequencyArray = frequency + 1;
}
Percentage = frequency/total*100;
}
InsertFCDlog(urlArray, frequencyArray, percentage);

While (store.log !=null) {
If (urlArray == storeArray) {
Search for cache file name in store.log and
search directory in SQUID;
Duplicate and forward to Firefox cache directory;
}
```

Figure 7. Cache Manager Algorithm

C. Filtered Cache Data Folder (FCD)

The Filtered Cache Data Folder or in short the FCD contains the FCD.log. The FCD.log contains information such as the selected cache item's URL address, frequency and also its calculated percentage. This information are produced and stored by the Cache Manager. Based on the FCD.log information, the Cache Manager will be able to utilize it in conjunction with the store log information to locate the location of the selected cache items and copy and append it to the Client's Firefox cache directory. Figure 8 below illustrates a screenshot of the FCD.log and its format.

```
- CACHED ITEM - - FREQUENCY -   - % -

http://pagead2.googlesyndication.com/pagead/osd.js      6       7.14
http://202.76.226.181/thumbs/98/9852047585.jpg  2       2.38
http://202.76.226.181/thumbs/95/9509810542.jpg  2       2.38
http://202.76.226.181/thumbs/94/9449327581.jpg  2       2.38
http://202.76.227.241/thumbs/97/9700320995.jpg  2       2.38
http://202.76.227.241/thumbs/97/9734101839.jpg  2       2.38
http://202.76.227.241/thumbs/95/9511436635.jpg  2       2.38
http://202.76.226.181/thumbs/97/9706083977.jpg  1       1.19
http://202.76.226.181/thumbs/95/9568250834.jpg  1       1.19
http://202.76.226.181/thumbs/96/9600133364.jpg  1       1.19
```

Cached item - http://pagead2.googlesyndication.com/pagead/osd/js
Frequency - 6
Percentage - 7.14

Figure 8. FCD.log Format

5. RESULTS AND DISCUSSION

A. Performance Evaluation

The performance evaluation is performed with the IBM Rational Performance Tester software. The evaluated data illustrates between the Firefox cache directory without any pre-selected cache items pushed by the Cache Manager and the Firefox cache directory with the pre-selected cache items pushed by the Cache Manager. All test results are simulated with five virtual users, five iterations and one iteration per minute for every test. The same setting is used for all test sets. The IBM Rational Performance Tester is used to simulate five virtual users at once to run the recorded simulations and also all sets are being iterated for five times with one iteration per minute. The iterations are configured because every run would not produce the same results. Once all five iterations are completed, IBM Rational Performance Tester will compute the average response time of all five iterations for each set. This allows a much more accurate test results to be produced. On the overall, there are four different set of tests. The tests are divided into two categories, the static page and the dynamic page. Each category consists of two other test sets which are the ones with the pre-selected cache items pushed by the Cache Manager and the ones without the pre-selected items. The evaluation does not specify the performance of the cache items specifically but illustrates performance of the web page as a whole. In most cases, a web page contains more than one cacheable item to be cached which is deemed sufficient in this evaluation. Figure 9 shows the websites performance evaluation.

Type	Website		Web Applicaiton
	Medium Scale	Large Scale	
Web-site	www.mudah.my	www.mbocinemas.com	www.elearning.utm.my

Figure 9. Websites Performance Evaluation Summary

B. Static Pages

All three websites are navigated to only their home or index page without involving any further navigation. Navigation to other pages is not

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

necessary since all three web pages have sufficient items to be cached at their home or index page. The evaluation criteria chosen is the average response time taken for each home page of a website to be successfully displayed. The unit of the average response time is in milliseconds. The figure below represents the average response time of each page to be displayed for both with and without any pre-selected cached items pushed by the Cache Manager.

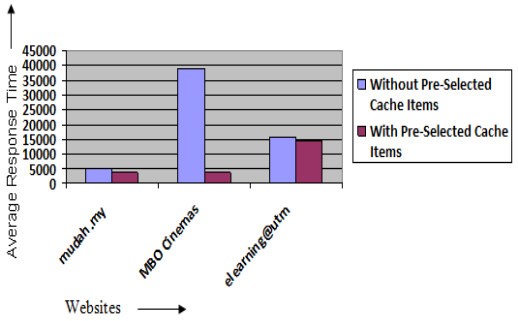

Figure 10. Summary of Static Pages

Figure 10 shows that unlike the rest, the MBO cinemas website illustrates a very huge gap in average response time compared to the other website and web application, www.mudah.my and the www.elearning.utm.my respectively. It could be assumed that due to MBO cinemas containing many images with large sizes could be the affecting criteria for the average response time taken to load the whole page. These however, realizes and proves that by selecting and comparing different websites containing different image sizes are able to show the degree of effectiveness of the Two-Tiered Web Caching technique under various types of websites and web application. The graph also proves that though an improvement can be seen under web application category which is the www.elearning.utm.my website, it does not have as much effect for static pages. The degree of improvement has almost the same level as www.mudah.my which is of a website containing medium scale images.

It could be concluded that with the pre-selected cache items which consists of the top thirty highest cache hit rate being pushed by the Cache Manager to the Firefox default cache directory increases the probability of acquiring a cache hit over at the client level which directly reduces the number of requests being sent to web server thus reducing the network latency. However, the above findings states for static pages only. The following section concerns the dynamic page with the same evaluating criteria.

C. Dynamic Pages

Figure 11 shows that there are drastic improvements from all of the websites. In fact, the degree of improvements is far better compared to the test results obtained from the evaluation of the static pages. Based on the test results from the evaluations of the dynamic pages, with the implementation of the 2TCS technique where the Cache Manager pushes the pre-selected cache items to the Firefox default cache directory illustrates an improvement in the range of 55% to 83% are achieved. This particularly means that the 2TCS technique works better with websites containing dynamic elements rather than static websites.

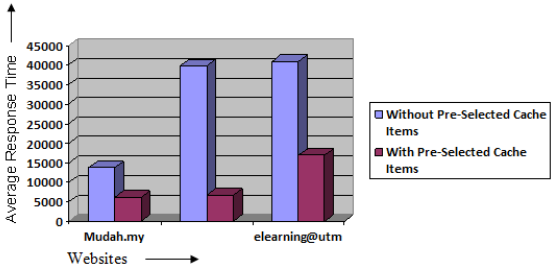

Figure 11. Summary of Dynamic Pages

Similar to the evaluations performed for the static pages, the evaluations of the dynamic pages are also performed in the day. Since networks are bound to serve more clients' request in the day than in the night, it proves networks are much more congested in the day rather than during the night. This also states that performing evaluations during the day serves to be the worst case scenario and despite the congested networks, with the implementation of the 2TCS, an astonishingly more than 50% of improvement can still be achieved. This also states that regardless of the time, the 2TCS is still able to achieve a considerable degree of improvement.

The graph proves that despite the static or dynamic page, the cache items which are being pushed by the Cache Manager all the way to the Firefox default cache directory affects the performance of loading the pages. The average time taken to load the pages is reduced greatly whenever the pre-selected cache items are being pushed to the Firefox default cache directory.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6. CONCLUSION AND FUTURE WORK

The number of internet users has drastically increases over time and has caused networks to be congested thus bottlenecks occurs in networks. Caching which is a common but effective and efficient technique has been implemented as part of the solution. The study proposed a Two-Tiered Web Caching architecture which utilizes the default caching system found in Mozilla Firefox, a web browser and also SQUID, a web cache server. In realizing the theory that by increasing the probability of acquiring a cache hit as near as possible to the client or better still at the client itself, the performance of the web server can be improved. In addition, the evaluation proves that the Two-Tiered Web Caching technique has the ability to improve the web server performance upon implementation. The theory of the technique which reduces the number of requests to the web server by pushing selected cache item with high cache hit rate over to the client level is proven. It can also be concluded that a higher cache hit rate will result in a lower usage of bandwidth thus reducing the network latency and also improving the web server performance.

In the future work with regards to this study, further improvements can be made regarding the algorithm of the Cache Manager. The algorithm in selecting the cache items can be further optimize by inducing more rules such as how to define popular cache items. Factors that concerns are the number of user determine by IP address along with the frequency. On top of that, disk space can be reduce should the algorithm intelligently pushed individual sets of cache items based on each individual browsing trend to their browser cache directory instead of pushing the same set. Another improvement could be the compatibility wise, it would be better should the algorithm supports chrome and Internet Explorer which increases the efficiency of the technique.

ACKNOWLEDGEMENTS

The authors would like to express their deepest gratitude to Universiti Teknologi Malaysia (UTM) for their financial support under Research University Grant Scheme (Vot number Q.J130000.7128.01H13).

REFERENCES

[1] Darrel, I. (2001). Web server performance metric. *Dictionary of the Internet*. Retrieved on March 21, 2011 from http://www.encyclopedia.com/doc/1O12Webserverperformancemetrc.html

[2] Kayne, R. (2011, March 16). *what is a web server?*, Retrieved on February 20, 2011, from http://www.wisegeek.com/what-is-a-web-server.htm

[3] Ling, Y., Chen, S. and Lin, X. (2003). Towards better performance measurement of web servers. *Proceedings of the 2003 joint Conference of the Fourth International Conference*. 15-18 December. Singapore: IEEE, 453-457.

[4] Iyengar, A., MacNair, E. and Nguyen, T. (1997). An analysis of web server performance. *Proceedings of the 1997 IEEE Global Telecommunications Conference*. 3-8 November. Phoenix, AZ: IEEE, 1943-1947.

[5] Kurebayashi, R., Obana, K., Uematsu, H., and Ishida, O. (2008). A Web Access Shaping Method to Improve the Performance of Congested Servers. *Proceedings of the 2008 7th Asia Pacific Symposium on Information and Telecommunication Technologies*. 22-24 April. Bandos Island, Maldives: IEEE, 120-125.

[6] Ho, L. K., Sit, H. Y., Ho, K. S. H., Leong. H. V. and Luk, W. P. R.(2004). Improving Web Server Performance by a Clustering-Based Dynamic Load Balancing Algorithm. *Proceedings of the 18th International Conference on Advanced Information Networking and Application*. 29-31 March. Fukuoka, Japan: IEEE, 232-235.

[7] Beltran, V., Torres, J. and Ayguad´e, E. (2008). Improving Web Server Performance Through Main Memory Compression. *Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems*. 8-10 December. Melbourne, Australia: IEEE, 303-310.

[8] Carter, R. and Cherkasova, L. (2003). Detecting Timed-Out Client Requests for Avoiding Livelock and Improving Web Server Performance. *Proceedings of the 5th IEEE*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Symposium on Computers and Communications. 3-6 July. Antibes-Juan Les Pins, France: IEEE, 2-7.

[9] Iyengar, A. and Challenger, J. (1997). Improving Web Server Performance by Caching Dynamic Data. *Proceedings of the 1997 USENIX Symposium on Internet Technologies and Systems*. 8-11 December. Monterey, California: USENIX, 262-265.

[10] Daesung Lee; Kim, K.J. A Study on Improving Web Cache Server Performance Using Delayed Caching. *Information Science and Applications (ICISA), 2010 International Conference*, pages 1-5, April 2010.