Indonesian Journal of Electrical Engineering and Computer Science
Vol. 7, No. 2, August 2017, pp. 474 ~ 482
DOI: 10.11591/ijeecs.v7.i2.pp474-482

Distributed Cache with Utilizing Squid Proxy Server and LRU Algorithm

Abdul Ghofir*[1], Rikip Ginanjar[2]
Faculty of Computer Science Jl. Ki Hajar Dewantara, Cikarang Baru, Bekasi, Indonesia
Phone: (021) 89109762-63
Corresponding author, e-mail: geoff@president.ac.id*[1], rikipginanjar@president.ac.id[2]

Abstract

In relation to the dissemination of information, the Internet is one of the fastest media to do so. The internet's presence is growing very swiftly and rapidly, so it has become recognized by people from all walks of life. For that, the people need the appropriate way to maintain effectiveness in the use of the Internet. The following paper describes a study of the distribution of the cache, which is performed by the squid proxy server by creating a storage network design on linux. Cache documents that are stored in the proxy server will be distributed to another over a network storage server. The process of caching on the proxy server is using the Least Recently Used (LRU) Algorithm. This research was carried out by developing the existing method of caching server process, then it is to be added aunitas a backup storagede vice for the data that must be erased because of the replacement policy applied to the squid proxy server. This study is looking at how the hit ratio and byte hit ratio after adding the storage server compared to not having a storage server. At the end of this research, it is concluded that the distributed cache processes a hit ratio and byte hit ratio higher than the cache onthe current proxy server.

Keywords : *LRU algorithm, squid proxy server, cache management, hit ratio, byte hit ratio*

1. Introduction

Cache is a method to save any document that is often accessed in order to make it easier for the user to re-access. It should be done in order for people to be able to easily and quickly access the internet. Commonly occurring in offices, when the internet connection is to slow, people usually think to increase the bandwidth. It is, however, not mistake, although it alsocosts a lot of money. There are many ways to improve a slow internet connection. One of them is through using cache management [1].

Cache management will process data from the internet that are stored on a local server so that they can be used properly to be presented again to the client in accordance with what was requested. To accommodate this objective, then a local server is needed to serve the client which is commonly called as a proxy server [2].

Proxy server is the key to improve web performance by keeping web objects that arelikely to be revisited by the client. It helps the client in the use of the Internet, especially with the issues of delays, slow response, bandwidth usage, and reducing the load on the original server [3].

To be able to store web objects, it needs a large space because it will be growing all the time. However, the space given to the cache is limited. Therefore, the space should be utilized effectively. Intelligence mechanism is needed to manage the content of web cached to be efficient. The essences of all those issues are cache replacements. Efficient cache replacement algorithm and network design is very paramount in building a computer network [4].

The network is acritical infrastructure for data and information exchange. The proxy server has limited space to store the web objects that are cached and it has limited bandwidth to access the web pages to the internet. Along with the increase of time, the data which is stored in the proxy server will be increasing in size and it willlead to a full storage. By the time the storage is full, network traffic will be jammed and the internet will be disconnected. Therefore, there should be an elimination of the web cache on the storage to make the network traffic empty and re-connect to the internet [5].

Received May 22, 2017; Revised July 5, 2017; Accepted July 23, 2017

Automatic deletion of data cache is the right choice to avoid full storage. However, this is still considered less effective because when the data that has been deleted are being questioned again, there will be a MISS and must be delivered to the original server. This will certainly slow down the process. For that, they need a way so that the data that has been deleted remain on the server.

2. Theory

Here are some theoretical explanations about cache, squid and LRU algorithms.

2.1 Web Caching

Cache (memory) is a memory that is located very close to the CPU, even one chip with a CPU to speed up the access process.As for web caching is a place of storage of web objects located close to the user to speed up re-access.The types of data or web objects include web pages (html), images, etc [6, 7].

Web objects can be stored locally on a client computer or on a web server. There are several kinds of cache for web objects [8].

a. Browser cache: The browser on the client computer caches the web object.The browser will initially search for the objects in the cache before requesting it on the website.
b. Proxy cache: A proxy cache that is installed close to the user, or usually in one company. Client user in the company will be configured in the browser to use the proxy that has been provided.The objects requested from the website will be retained and served by the proxy cache. If the requested object is not in the cache, then the proxy will continue the request to the destination website.
c. Transparent proxy cache: The proxy configuration that the client is connected to the proxy does not have to set the browser one by one, so only the redirect is done so the computer can take advantage of the proxy as cache.
d. Reverse (inverse) proxy cache: A proxy that is in front of the web server, is used as a cache or can also be used as a load balancer.

The web object on the cache has a time-bound restriction. After passing the specified time limit, it will be considered "stale" and can not be used again.Even web objects will no longer be available in cache storage.

2.2 Cache Replacement

Cache replacement is a cache update process from a cache that has staled with a new cache, in accordance with the regulation that have been determined. This should be done so that the cache on the storage is not full and stuck occurs which resulted in the internet stop.To perform this replacement process required a method of regulation or called an algorithm [9, 10].

2.3 LRU Algorithm (Least Recently Used)

The LRU algorithm is a popular algorithm for handling cache replacement. The working principle of this algorithm is a page that has not been used in the longest time will be replaced with a new page [11-13]. Here is an overview of LRU algorithm implementation.

Table 2.1. LRU Algorithm

Frame 1																								
0	3	1	3	2	3	6	0	4	1	4	2	4	7	0	5	1	5	2	5	8	0	3	1	3
Frame 2																								
0	3	1	3	2	3	6	0	4	1	4	2	4	7	0	5	1	5	2	5	8	0	3	1	3
	0	3	1	3	2	3	6	0	4	1	4	2	4	7	0	5	1	5	2	5	8	0	3	1
		0	0	1	1	2	3	6	0	0	1	1	2	4	7	0	0	1	1	2	5	8	0	0

In Table 2.1, it is illustrated as how the LRU algorithm works. There are 25 string references displayed in a frame (frame 1). Then the string is inserted into 3 frames (frame 2). The first string is 0, then it goes to the first frame under it. Next, the second string is worth 3,

goes to the first frame and replaces the number 0 which then shifts to the second frame below it. Then the third string, shifting the position of the number 3, then the number 3 replaced the number zero which shifts again to the bottom frame.Then the string moves to replace each other below it continuously, eliminating the string value that is not used for the longest.

2.4 Cache Management on Squid Proxy Server

Squid is a web cache proxy that supports HTTP, HTTPS, FTP, and so on. The squid function here is to reduce bandwidth usage and improve response time by caching and reusing the requested web page periodically.Squid has extensive access control and makes a great server accelerator and can run on a variety of operating systems, including Linux and Windows.

Figure 2.1 is the the structure area of squid version 3.3.13. Squid that is operated on the linux operating system will be in the directory: /etc/squid/squid.conf.As for the cached file will be in the directory: / var / spool / squid.No user authentication and blocking sites are enabled, so all clients can access any website and have the same internet access rights.

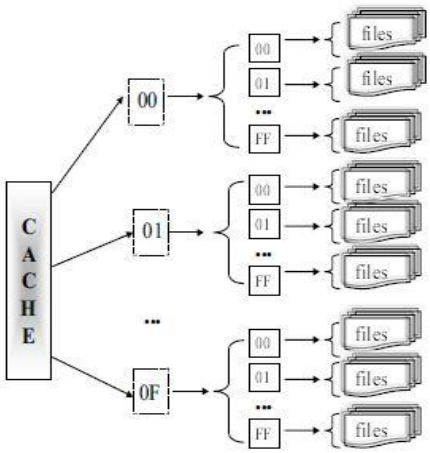

Figure 2.1. Structure of Squid Proxy Cache Area [14]

2.5 Design and Implementation of Page Replacement Algorithm for Web Proxy Caching

Proxy caching is a way to store data from the original web server, after one client accesses the internet. This is done to facilitate the re-access by these clients and other clients to become easier and faster. The data will be stored in a storage area of data (storage) on the proxy server [15].

One of the best applications to build a proxy server is squid. Squid is an open source application that is very famous and reliable, as well as can also be functioned as a firewall in a network.

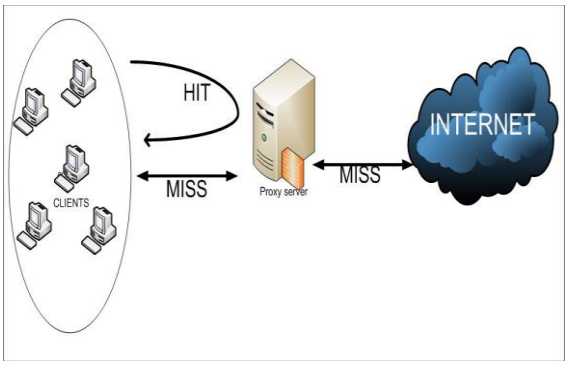

Figure 2.2. Design of Caching Proxy Server

Squid sets client requests to the proxy server. If the requested data is available on the proxy server, then the proxy server will serve the client's request (HIT). But if the requested data is not available on the proxy server (MISS), then squid will continue the request to the original web server. Then the web server will serve the request by going through the proxy server which will then be cached and forward it to the requesting client [16]. Design and implementation of proxy server caching as shown in Figure 2.2.

3. Design and Implementation of Distributed Cache with Utilizing Squid Proxy Server and LRU Algorithm

The following study is meant to create a new design, by adding a storage server as shown in Figure 3.1. The purpose of adding storage server is to increase data storage as well as to increase the probability of the existence of the data.

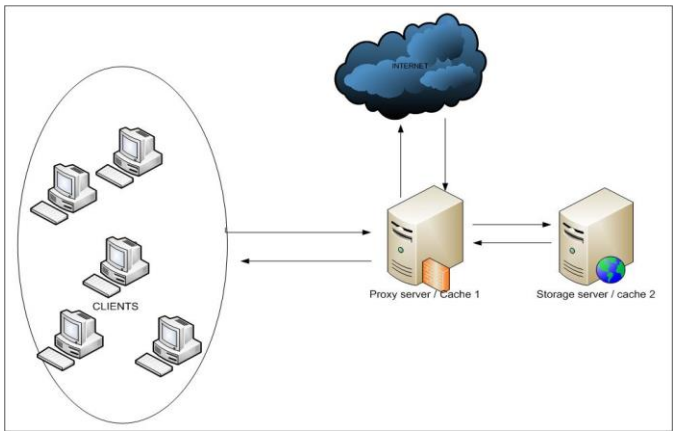

Figure 3.1. Proxy and storage server

When the data on the proxy server has been removed because it is not being accessed for a long time, then the proxy will convey the request to the original server. However, at the design of the following study, the proxy server does not deliver the request to the original server, but it will find the cached data to the storage server that has been made.

4. Research Method

The methodology in this research is using role playing simulation as shown in Figure 4.1. In this simulation, the data search is carried out bya player that seems to perform Internet activities.

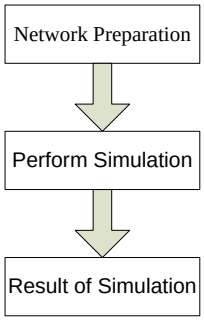

Figure 4.1. Simulation chart

4.1 Network Preparation

The first step in doing this research is the preparation of computer networks. There are two things that must be done in the preparation of computer networks, namely the preparation of hardware for computer networks and software installation.

4.1.1 Preparation of hardware and Computer Network

Some necessary hardware, including the computer, switch, Ethernet cable, and internet connection are installed at this step.

4.1.2 Software Installation

This simulation uses the operating system Linux Fedora 20 as aproxy server and storage server. Then the operating system forthe client computer is Windows 8. Mozilla Firefox is used as internet application.

4.2 Perform Simulation

The simulation will be done by two scenarios. First scenario is performing the network simulation as shown in Figure 2.2 and Ffigure 3.1. The second scenario is performing network simulation by combining both of them, as shown in Figure 4.2.

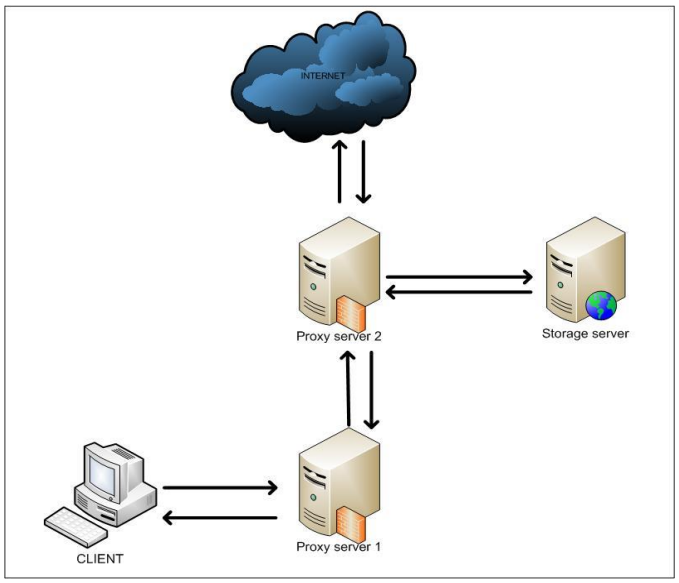

Figure 4.2 Combination of network I and network II

4.2.1 First Scenario

As mentioned above, there are two different networks prepared to simulate. Network I consists of one proxy server and one client. Network II consists of one proxy server, one storage server as addition, and one client. Each of the clientswilldo a simulation with internet browsing activities on certain websites at the same time as shown intable4.1.

Table 4.1. Internet simulation at first scenario

Client	Website	Number of Access	Interval (sec)
Network I	www.detik.com	500	25
	www.liputan6.com	500	25
Network II	www.detik.com	500	25
	www.liputan6.com	500	25

4.2.2 Second Scenario

At the second scenario, both of the networks as simulated in the first scenario is combined. It is intended that each proxy receives the exact same request. Table 4.2 is the scenario of what is done.

Table 4.2. Internet simulation at second scenario

Website	Number of Access	Interval (sec)
www.detik.com	500	25
www.liputan6.com	500	25

5. Results and Analysis

Here is a comparison for the results from the analysis of hit ratio and byte hit ratio between caching proxy and caches distributed.

5.1 First Scenario Result

5.1.1 Hit Ratio Analysis of First Scenario

Figure 5.1 is the hit ratio graph of first scenario. It shows that:

1. For object size 0 byte, hit ratio in network I is 54.61% and hit ratio in network II is 71.28%. Network II is higher than network I.
2. For object size 1 – 9 bytes and 10 – 99 bytes, it shows that hit ratio in network 1 is 0% and hit ratio in network 2 is also 0%. Both of lines are looked going down. It means that there are no hit for both of them.
3. For object size 100 – 999 bytes, hit ratio in network I is 0.32% and hit ratio in network II is 0.44%. Hit ratio in network II is higher than network I.
4. For object size 1,000 – 9,999 bytes, hit ratio in network I is 4.86% and hit ratio in network II is 5.97%. Hit ratio in network II is higher than network I.
5. For object size 10,000 – 99,999 bytes, hit ratio in network I is 11.15% and hit ratio in network II is 12.38%. Hit ratio in network II is higher than network I.
6. For object size 100,000 – 999,000 bytes, hit ratio in network I is 11.49% and hit ratio in network II is 15.38%. Hit ratio in network II is higher than network I.
7. For object size 1,000,000 – 9,999,999 bytes, hit ratio for both of them are the same, 0%. It means there are no hit for this object size.

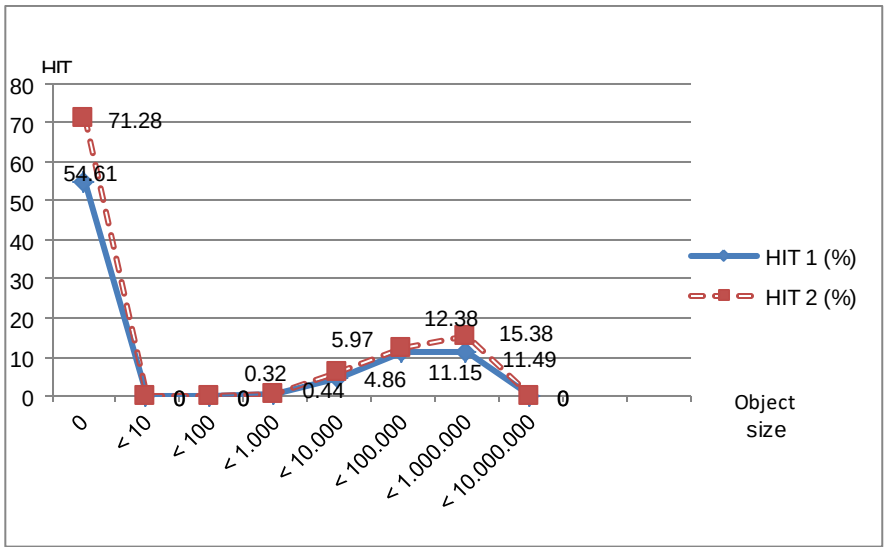

Figure 5.1. Hit ratio graph of first scenario

5.1.2 Byte Hit Ratio Analysis of First Scenario

Figure 5.2 is the byte hit ratio graph of first scenario. It shows that:

1. For object size 0 – 0 byte, 1 – 9 bytes, and 10 - 99 bytes, byte hit ratio for both of network are 0%. It means there is no data return to the client.
2. For object size 100 – 999 bytes, byte hit ratio in network I is 0.6% and byte hit ratio in network II is 0.89%. Byte hit ratio in this point for network II is higher than byte hit ratio in network I.
3. For object size 1,000 – 9,999 bytes, byte hit ratio in network I is 7.87% and byte hit ratio in network II is 9.18%. Byte hit ratio in network II is higher than byte hit ratio in network I.
4. For object size 10,000 – 99,999 bytes, byte hit ratio in network I is 12.81% and byte hit ratio in network II is 13.14%. Byte hit ratio in network II is higher than byte hit ratio in network I.
5. For object size 100,000 – 999,000 bytes, byte hit ratio in network I is 8.62% and byte hit ratio in network II is 10.76%. Byte hit ratio in network II is higher than byte hit ratio in network I.
6. For object size 1,000,000 – 9,999,999 bytes, byte hit ratio for both of them are the same, 0%. It means there are no data hit for this object size.

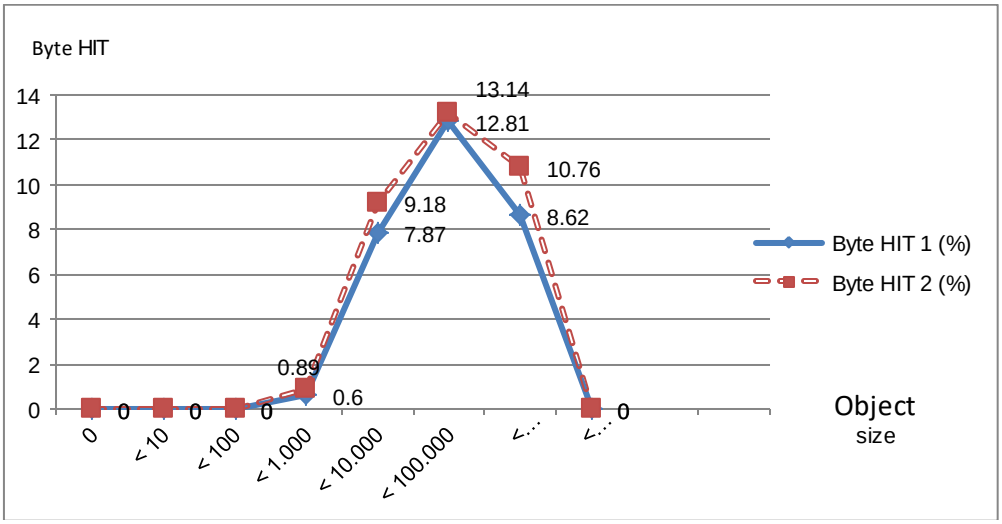

Figure 5.2. Byte hit ratio graph of first scenario

5.2 Second Scenario Result

5.2.1 Hit Ratio Analysis of Second Scenario

Figure 5.3 is the hit ratio graph of second scenario. It shows that:

1. For object size 0 byte, hit ratio in network I is 34.95% and hit ratio in network II is 37.35%. Network II is higher than network I.
2. For object size 1 – 9 bytes and 10 – 99 bytes, it shows that hit ratio in network I is 0% and hit ratio in network II is also 0%. It means that there are no hit for both of them.
3. For object size 100 – 999 bytes, hit ratio in network I and hit ratio in network II are the same, it is 0.95%.
4. For object size 1,000 – 9,999 bytes, hit ratio in network I is 1.69% and hit ratio in network II is also 1.69%. Hit ratio in network I and network II are the same.
5. For object size 10,000 – 99,999 bytes, hit ratio in network I is 5.4% and hit ratio in network II is 5.89%. Hit ratio in network II is also little higher than network I.
6. For object size 100,000 – 999,000 bytes, hit ratio in network I is 3.08% and hit ratio in network II is 3.08%. Hit ratio in network I and hit ratio in network II have the same position.
7. For object size 1,000,000 – 9,999,999 bytes, hit ratio for both of them are the same, 0%. It means there are no hit for this object size.

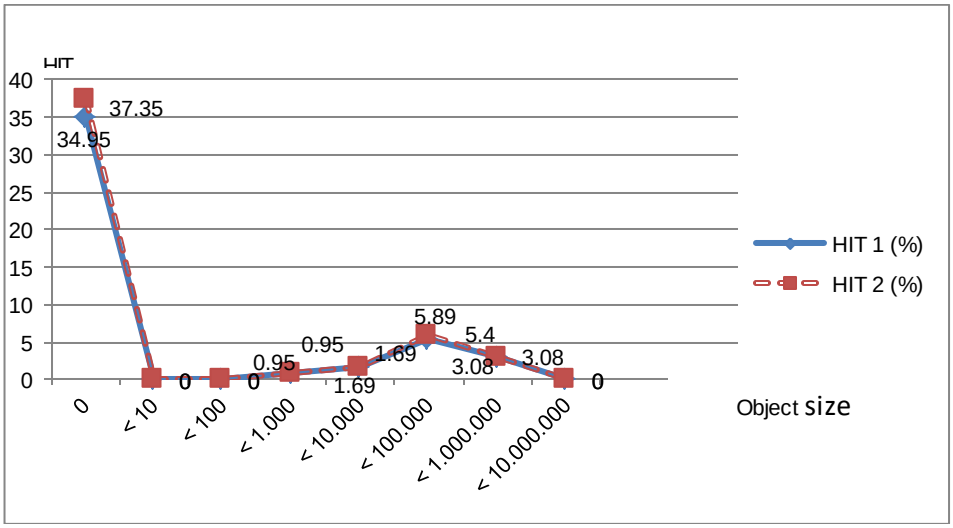

Figure 5.3 Hit ratio graph of second scenario

5.2.2 Byte Hit Ratio Analysis of Second Scenario

Figure 5.4 is the byte hit ratio graph of second scenario. It shows that:

1. For object size 0 – 0 byte, 1 – 9 bytes, and 10 - 99 bytes, byte hit ratio for both of network are 0%. It means there is no data return to the client.
2. For object size 100 – 999 bytes, byte hit ratio in network I is 0.57% and byte hit ratio in network II is 0.57%. Both of networks have the same byte hit ratio.
3. For object size 1,000 – 9,999 bytes, byte hit ratio in network I is 2.11% and byte hit ratio in network II is 2.12%. Byte hit ratio in network II is higher than byte hit ratio in network I.
4. For object size 10,000 – 99,999 bytes, byte hit ratio in network I is 6.32% and byte hit ratio in network II is 6.60%. Byte hit ratio in network II is higher than byte hit ratio in network I.
5. For object size 100,000 – 999,000 bytes, byte hit ratio in network I is 2.04% and byte hit ratio in network II is 2.04%. Byte hit ratio in network I and byte hit ratio in network II are the same.
6. For object size 1,000,000 – 9,999,999 bytes, byte hit ratio for both of them are the same, 0%. It means there are no data hit for this object size.

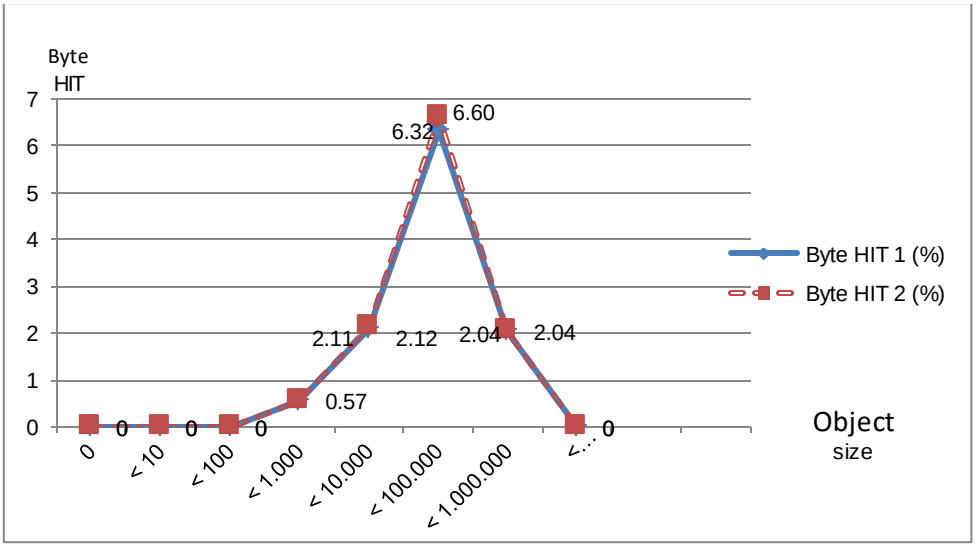

Figure 5.4. Byte hit ratio graph of second scenario

6. Conclusion

Two experiments have been conducted. Each experiment has yielded data and information that can be analyzed easily and clearly. For the first experiment, it was resulted in hit ratio 5.44% and byte hit ratio 10.2% in network I and in network II was resulted in hit ratio 6.86% and byte hit ratio10.33%.

Thus, in the experiment with the first scenario, network II hashit ratio and byte hit ratio higher than in the networkI.

For the second experiment, it resulted that hit ratio and byte hit ratio in network I are 1.56% and 3.68%. In network II, hit ratio and byte hit ratio are 1.86% and 3.81%. Thus, in the experiment with the second scenario, network II has higher hit ratio and byte hit ratio.

By analyzing the experiment resulted for both of scenarios, it is concluded that distributed cache has hit ratio and byte hit ratio higher than proxy cache.

References

[1] Mukesh Dawar, Charanjit Singh. A Review on Web Caching Techniques. *IJARCSSE*. 2014; 4(3): 18–22.
[2] Yogesh Niranjan, Shailendra Tiwari. Design and Implementation of Page Replacement Algorithm for Web Proxy Caching. *IJCTA*. 2013; 4(2): 221-225.
[3] Daniel Zeng, Fei-Yue Wang, Mingkuan Lui. Efficient Web Content Delivey Using Proxy Caching Techniques. *IEEE Transactions on Systems*. 2004; 34(3): 270-280.
[4] Subhash Chand, Sanjay Mathur. Squid Proxy Server Cache Management Using K-Means Algorithm. *IJCSIT*. 2014; 5(2): 1918-1923.
[5] Anirban Mahanti, Carey Williamson, Derek Eager. Traffic Analysis of a Web Proxy Caching Hierarchy. *IEEE Network Magazine*. 2000.
[6] Yun Ma, Xuanzhe Liu, Shuhui Zhang, Ruirui Xiang, Yunxin Liu, Tao Xie. *Measusment and Analysis of Mobile Web Cache Performance*. International World Wide Web Conference Committee (IW3C2). Florence. 2015: 691-701.
[7] Waleed Ali, Siti Mariyam Shamsuddin, Abdul Samad Ismail. A Survey of Caching and Prefetching. *IJASCA*. 2011; 3(1): 1-27.
[8] Deepak Sachan, Dhawaleshwar Rao Ch. Performance Improvement of Web Caching Page Replacement Algorithms. *IJCSIT*. 2014; 5(3): 3112-3115.
[9] Sam Romano, Hal El Aarag. A Neural Network Proxy Cache Replacement Strategy and Its Implementation in the Squid Proxy Server. *Neural Computing and Applications*. 2011; 20: 59-78.
[10] Charu Anggarwal, Joel Wolf, Philip Yu. Caching on the World Wide Web. *IEEE Explore*. 1999.
[11] Kapil Arora, Dhawaleswar Rao Ch. Web Cache Page Replacement by Using LRU and LFU Algorithms with Hit Ratio: A Case Unification. *IJCSIT*. 2014; 5(3): 3232-3235.
[12] Julian Benadit, Sagayaraj Francis, Nadhiya M. Enhancement of Web Proxy Caching Using Simple K-Means Clustering. *IJSER*. 2014; 5(5): 248-255.
[13] LudmilaChercasova. Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Proxy. *Computer Systems Laboratory*. 1998.
[14] K Bauknecht, S Kumar Madria, G Pernul. *LRU-based Algorithms for Web Cache Replacement*. Ec-Web 2000. Berlin. 2000: 409-418.
[15] Anawat Chankhunthod, Peter B Danzig, Chuck Neerdaels, Michael F Schwartz, Kurt J Worrel. A Hierarchical Internet Object Cache. *Departement of Computer Science*. 1996.
[16] Pei Cao, Sandy Irani. *Cost Aware WWW Proxy Caching Algorithms.* Proceedings of USENIX Symposium on Internet Technologies and Systems. California. 1997.