# IMPROVE SQUID PROXY'S PERFORMANCE USING NEW CACHE REPLACEMENT ARCHITECTURE

**Thanki Kunal Umeshchandra***

**Patel Chirag Ramjibhai***

***Abstract:***

The usage of internet has been drastically increased in few years. This increase demands larger and larger bandwidth to work in easy and fast manner. But with limited bandwidth usage we can provide faster internet working by using proxy caches. That is where proxy server comes in to the picture. We have large amount of proxy servers available out of which some are open source. We have to configure these servers according to our requirement by setting it's architecture and cache replacement algorithm. This report discuss new hybrid caching architecture by setting hybrid cache replacement algorithm. In this report we proposed a new hybrid architecture instead of using single server which should improve working of server.

***Keywords:*** Proxy, caching, cache replacement algorithm, hybrid architecture, co-operative caching

* ME ,Computer science and engineering -Sem 4th, Gov. Engineering college, Modasa.

## INTRODUCTION

A proxy server is a computer system sitting between the client requesting a web document and the target server (another computer system) serving the document. In its simplest form, a proxy server facilitates communication between client and target server without modifying requests or replies. When we initiate a request for a resource from the target server, the proxy server hijacks our connection and represents itself as a client to the target server, requesting the resource on our behalf. If a reply is received, the proxy server returns it to us, giving a feel that we have communicated with the target server. In this manner it hides private network and private IP from public network. In advanced forms, a proxy server can filter requests based on various rules and may allow communication only when requests can be validated against the available rules. The rules are generally based on an IP address of a client or target server, protocol, content type of web documents, web content type, and so on.

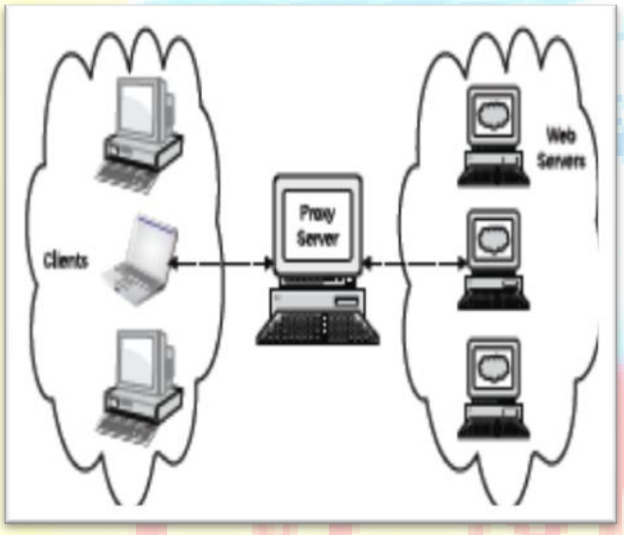

**Fig:1 Proxy server sits between private network and public network.**

As seen in the figure 1, clients can't make direct requests to the web servers. To facilitate communication between clients and web servers, we have connected them using a proxy server which is acting as a medium of communication for clients and web servers.

Sometimes, a proxy server can modify requests or replies, or can even store the replies from the target server locally for fulfilling the same request from the same or other clients at a later stage. Storing the replies locally for use at a later time is known as **caching**. Caching is a popular technique used by proxy servers to save bandwidth, empowering web servers, and Improving the end user's browsing experience.

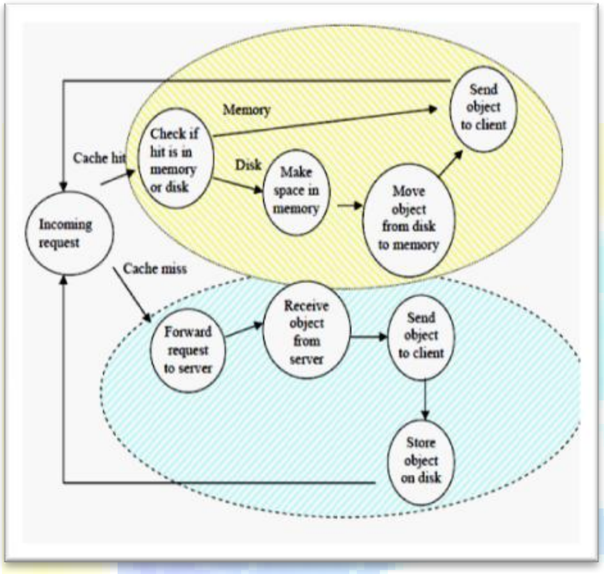

**Fig. 2. State diagram of caching proxy server**

As shown in figure 2 above, the state diagram can be drawn like this showing how actually request is served from the client. On the request of the object available in cache, it'll directly be served from cache after making sure that copy that is available with cache is fresh. If copy is not fresh or not available with cache it has to retrieve from original server.

Proxy servers are mostly deployed to perform the following:

- Reduce bandwidth usage
- Enhance the user's browsing experience by reducing page load time which, in turn, is achieved by caching web documents
- Enforce network access policies
- Monitoring user traffic or reporting Internet usage for individual users or groups

- Enhance user privacy by not exposing a user's machine directly to Internet
- Distribute load among different web servers to reduce load on a single server
- Empower a poorly performing web server
- Filter requests or replies using an integrated virus/malware detection system
- Load balance network traffic across multiple Internet connections.
- Relay traffic around within a local area network

## Co-operative caching*:*

In a proxy cache, a single host is used for one to several hundred clients. Each client sends its requests through the proxy which keeps copies of objects in the cache. Hence, the involvement of the proxy cache has decreased the user perceived network latency. Proxy caches are placed at different places in the network to serve the clients. These proxy caches also cooperate with one another in case of a cache miss. If a requested web object from a client is not found in a local proxy cache, the proxy cache communicates with its sibling or nearby proxy caches to find that object. In the event that object is not found in the nearby caches only then the request is forwarded to the original server. This cooperation between the proxy caches is called the cooperative caching. There are two scenario to provide co-operative caching: distributive and hierarchical caching. But both have some advantage as well as disadvantage. One of the drawbacks of hierarchical architecture is that it can introduce an additional delay whenever the request is not satisfied in a given level of cache since the request is forwarded to upper levels in hierarchy. Another drawback is that the higher level cache can have a longer queue than the lower level cache resulting in a bottleneck. While distributive caching broadcasts ICP queries when cache miss occurs and retrieve object from sibling cache . so it will produce additional traffic in

## Proposed architecture

In our proposed architecture we have to make hybrid architecture including both, sibling as well as parents. All proxy cache server will maintain one reference table which will contain the details of [proxy_id],[object_id],[date_created], and [date_modified]. So when cache miss occurs instead of directly approaching to original server , first it will send ICP query to it's parents or sibling by referring it's reference table. Which ever proxy cache has the requested object available, it'll reply to the requesting server. Here the object will not be send to the server. Instead reference to that proxy server will be sent to requester informing the location of required object. So by this technique we can provide co-operation without involving more traffic. Here in this architecture we can perform different cache replacement scenario with different cache replacement algorithm. So we can observe and simulate this situation to measure different parameter for performance improvement

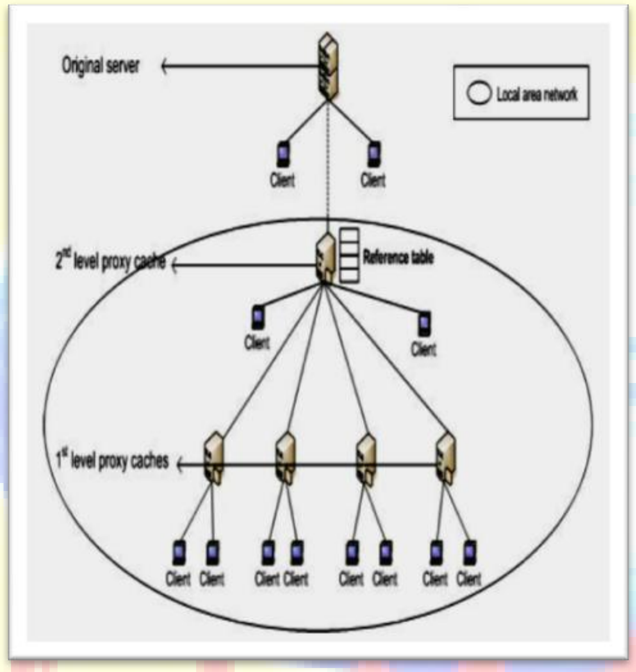

**Fig3. Proposed hybrid architecture**

**Implementation***:*

***Basic Requirements***

- Linux operating system(we have used Fedora 14 OS).
- Squid 3.1Stable Version(available at www.squid-cache.org)
- Web polygraph tool to simulate original environment.

## INSTALLING AND CONFIGURING SQUID PROXY SERVER.

1 GET THE SOURCE CODE

2 compile it by

```
./configure-- prefix= /usr/local
          Make
          Make install
```

3. starting squid server: squid –z

```
          service squid start
```

Default directory structure of squid:

```
/usr/local/squid/
/bin/
/cache/
/etc/
/src/squid-3.1/
```

Working through each directory below /usr/local/squid

**Configure:**squid.conf located in etc/squid/squid.conf

Basic Changes that are required for proxy caching is:

```
http_port 3128
icp_port 0
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY
cache_mem 32 MB
cache_dir ufs /cache 200 16 256
```

```
redirect_rewrites_host_header off

replacement_policy LRU

acl localnet src        192.168.1.0/ 255.255.255.0

acl localhost src 127.0.0.1/ 255.255.255.255

acl Safe_ports port 80 443 210 119 70 21 1025-65535

acl CONNECT method CONNECT

acl all src 0.0.0.0/0.0.0.0

http_access allow localnet

http_access allow localhost

http_access deny !Safe_ports

http_access deny CONNECT

http_access deny all                        cache_mgr kunal.thanki@gmail.com

cache_effective_user squid

cache_effective_group squid

log_icp_queries off

cachemgr_passwd my-secret-pass all

buffered_logs on
```

## WORKING WITH WEB POLYGRAPH TOOL

Web polygraph is freely available tool used to generate reports of proxy performance. It can simulate real world scenario of caching proxy servers. We can generate different cache architecture and can measure it's performance based on traffic generated by polygraph client and request served by polygraph server.

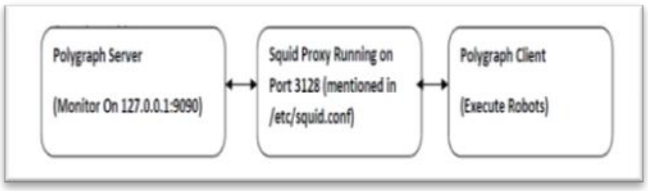
Polygraph Server
(Monitor On 127.0.0.1:9090)
Squid Proxy Running on Port 3128 (mentioned in /etc/squid.conf)
Polygraph Client
(Execute Robots)

**Fig 4. Web Polygraph environment**

**Polygraph's features include:**

•High-performance HTTP clients and servers

•Realistic HTTP and SSL traffic generation

•HTTP Basic, Negotiate proxy authentication

•flexible content simulation

•Ready-to-use standard workloads for report based on the reply received by the polygraph-client.

**Configure Web poly graph with Squid proxy:-**

**To Start Server:-**

[root@root] #polygraphserver–config/usr/local/polygraph/share/polygraph/ workloads /simple.pg --verb_lvl 10 –log /tmp/serv.log

**To Start Client :**

[root@root] #polygraph-client–config/usr/local/polygraph/share/polygraph /workloads /simple.pg --verb_lvl 10 –proxy 127.0.0.1:3128 –log /tmp/clt.log

**To Generate Report :**

[root@root] #polygraphreporter –label "Test_1" /tmp/*log

**Experimental setup for squid proxy server**

Instead of using single proxy server I suggest different architecture of caching server and generate output report in web polygraph by using access.log that will be generated by squid. Here we are dealing with cache hierarchy which will surely improve the proxy performance by communicating with other. The basic figure of working architecture is given bellow. we describe a new hybrid caching architecture that can take advantage of both hierarchical and distributed caching. As shown in Fig 5, we assume there are two-level hierarchies of caches and each level consists of more than one cache. Therefore, when a client requests a web object to its nearest proxy cache it will serve the request if it has that object in it's cache memory. If not, than the server forward this request to the next proxy in it's hierarchy and so on. Each server other than lowest in hierarchy maintain one reference table, so, on getting miss on local cache each server sends request according to its reference table. Each reference table maintain some entry in table showing where the required web object resides. Major fields are {Proxy ID} and {Object ID}. Proxy id represents the location of object where it currently resides and object id defines object itself. So each reference table contain eventually proxy id with respect to its object id. Following gives detailed description of each field in reference table:

**{Object ID}:** A unique ID for all the web objects cached at the lower level proxy cache.

**{Proxy ID}:** Specifies the proxy cache where a particular object is cached.

***{Date Created}*:**Specifies the exact date and time when an object is cached on the proxy cache from the original server.

***{Date Modified}*:** Specifies the recent date when that object is accessed.

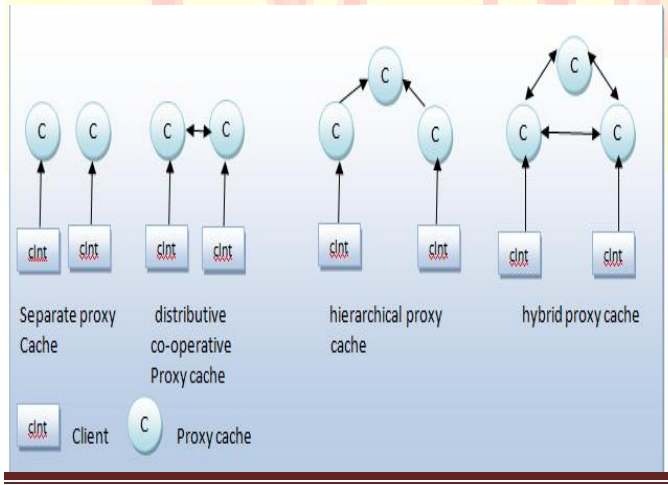

**Fig 5. Implemented scenario**

This involves following steps

```
1. Your browser sends a content request to proxy on your machine.
2. Check: if a cache HIT from institute proxy cache (HIT means content was found in cache)
    2a. Check: if content is older than the original upstream content
        2aa. Fetch content from upstream and serve the client
    2b. else
        2ba. Serve the content from the cache
3. Check: if cache HIT from proxy on your machine
    3a. Check: if content is older than the original upstream content
        3aa. Fetch content from upstream and serve the client
    3b. else
        3ba. Serve the content from the cache
4. Cache MISS from both the proxies
    4a. Fetch the content from upstream and serve the client
```

**Experimental outcomes**

Here we compare the performance of different cache hierarchies built with 2 and 3 Squid caches. Figure 5 lists the name and topology of the hierarchies we proposed. Experimental results are shown in Figure 6 and 7.

We point out some interesting observations. Comparing the hit ratios of separate caches with corresponding all distributive co-operative (sibling) hierarchies, we observe that the overall hit ratio increases significantly – from around 33% to 55%, which underlines the benefits of peering. A more interesting observation is that the local hit ratios are also higher when there are siblings.

The reason is that the popular objects are more likely to stay in a cache because of remote hits, which improves the local hit ratio also. The average response time for all cooperative hierarchies is better than separate caches. This shows that the impact of improved hit ratios due to parents and siblings outweighs the delays and overheads of cooperative caching in this setup. However, this situation can change for higher delay between caches.

The overall response time performance is best for an all sibling hierarchy distributive cooperative caching for 3 caches. It appears that the role of siblings is more crucial than parents in our experiments since all the higher ranked hierarchies have sibling configurations. Normally, sibling caches are part of a single organization, here as parent caches are located closer to the Internet backbone. Parents are also expected to have more storage and CPU power, and have several children..

By comparing the hit ratio as well as mean response time for more than 2 cache server we can conclude that as the number of cache server increase in hybrid architecture the hit ratio will also increase and bandwidth utilizing will decrease. But in distributive caching all server have to communicating with each other for the requested object which in turn increase the response time

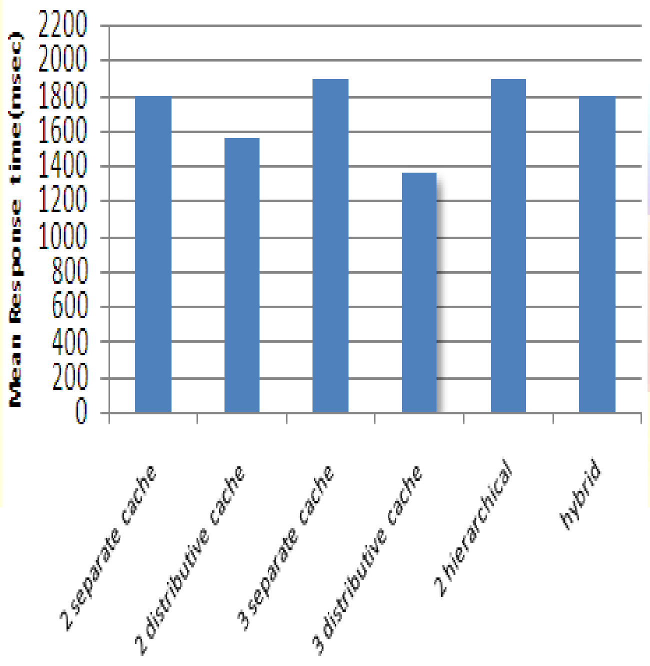

**Fig.6 Mean response time for all architecture**

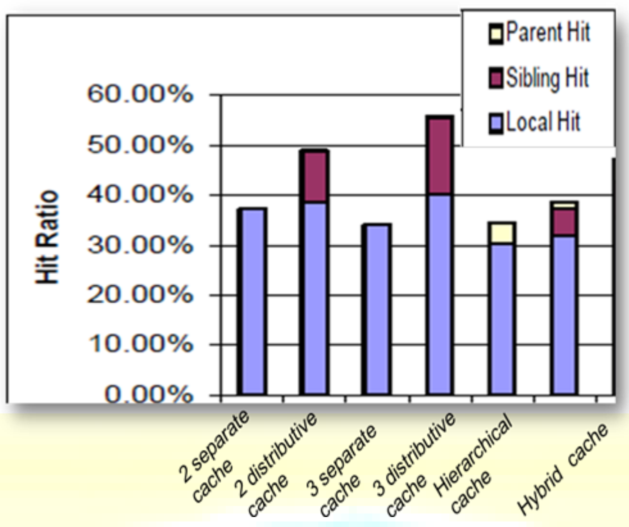

**Fig.7.Mean response time for all architecture**

Another important question in Web cache usage is the extent of performance improvement with increasing disk space, and whether that depends on cache cooperation

**Conclusion and future work**

By referencing this paper and after this experiment, we can say that co-operative caching dramatically increase the cache hit ratio due to locality of reference. And correspondingly response time may be little less due to accessing object directly from co-operative cache instead of original server. For future work we can test same architecture with different cache replacement algorithms and with varying cache disk size.

## References

[1] P. Du. “Evaluating of Cooperative Web Caching with Web Polygraph”. Master thesis. University of Houston, Department of Computer Science, October 2009

[2] J. Baek and M. Kanampiu, *A NAK suppression scheme for group communications considering the spatial locality of packet losses*, International Journal of Computer Science and Network Security, vol. 6, no. 10, pp. 158-167, October 2006

[3] *An Adaptive Coherence-Replacement Protocol for Web Proxy Cache Systems* Jose Aguilar Ernst L. Leiss , Department of Computer Science University of Houston .Houston, TX 77204-3475, USA *Article received on June 28, 2002; accepted on June 30, 2004*

[4] Silberschatz, A., and Galvin, P., Operating System Concepts, 7th ed., Addison Wesley Longman, 2003.

[5] Psounis, K., “Probabilistic Methods for Web Caching and Performance Prediction of IP Networks and Web Farms”, PhD Thesis, Stanford University, December, 2002. [11] Fiat, A., Karp,

[6] ACDFJ99. M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, “Evaluating Content Management Techniques for Web Proxy Caches”, in Proceedings of the 2nd Workshop on Internet Server Performance (WISP ‘99), Atlanta GA, May 1999

[7] LRU-based algorithms for Web Cache Replacement A. I. Vakali Department of Informatics Aristotle University of Thessaloniki, Greece

[8] Rizzo, L., and Vicisano, L., “Replacement Policies for a Proxy Cache”, IEEE/ACM Transactions on Networking, Vol 8., No. 2., April, 2000.

[9] Squid Proxy Server 3.1 Beginner's Guide, ISBN 978-1-849513-90-6

[10] Jin, S., and Bestavros, A., “GreedyDual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams”, *5th International Web Caching and Content Delivery Workshop*, Lisbon, Portugal, May, 2000.

[11] X. Zhuo, Q. Li, G. Cao, Y. Dai, B. Szymanski, and TL Porta. Social-based cooperative caching in dtns: A contact duration aware approach. In Proc. of IEEE MASS, 2011.

[13] Ager, B. (2010). Revisiting Cacheability in Times of User Generated Content. 2010 INFOCOM IEEE Conference on Computer Communications *Workshops*.

***Bibliography:***

http://www.deckle.co.za/squid-users-guide/ Cache_Hierarchies/

http://www.faqs.org/docs/securing/chap28sec231.html/

http://squid-docs.sourceforge.net/latest/html/ book1.htm/

http://squid.visolve.com/squidconf.html

http://www.squid-cache.org/

http://www.novell.com/documentation/suse91/suselinux-adminguide/html/index.html