

Next Generation Proxy Servers

W. V. Wathsala*, Buddhika Siddhisena†, Ajantha S. Athukorale‡

University of Colombo School of Computing

35, Reid Avenue

Colombo 7, Sri Lanka

Email: wvi@ucsc.cmb.ac.lk

University of Colombo School of Computing

35, Reid Avenue

Colombo 7, Sri Lanka

Email: bud@thinkcube.com

University of Colombo School of Computing

35, Reid Avenue

Colombo 7, Sri Lanka

Email: aja@ucsc.cmb.ac.lk

Abstract—Proxy Servers currently play an important role in a network by making efficient use of bandwidth through caching. In this paper we discuss on broadening the concept of proxy and Internet cache to suit current and future web 2.0 and web X.0 trends, especially in the areas of audio, video, blogging, social networking and mesh computing technologies. In doing so, we present ideas of a Next Generation Caching Server implemented as a platform for developing web service oriented applications and share the knowledge and experience gained in deploying a prototype at the University Network we work in.

Keywords—Next generation proxy server, cached content handling, web services for caching servers, NGP.

I. INTRODUCTION

Today the web has turned out to be a very rich medium for not only publishing web pages consisting of text, images and the occasional flash intro, but also as a medium for publishing Video, Music, Photos, Blogs and Podcasts. Web 2.0 has provided a new medium to truly collaborate among users by enabling the user to be the publisher and the subscriber of content [13].

Currently Proxy Servers or Caching Servers as we will refer from here on are primarily designed around the idea of reusing cache objects to save bandwidth and improve performance. In doing so several trade offs must be made to determine which objects to cache and which not to due to limited resources such as disk space, memory or processing power. In order to conserve these resources, in most situations, caching servers are configured to prefer small cache objects over larger cached objects as well as limit the total cache size. As a result, today's cache servers fail to cache mediums such as video, music and photos which continue to grow in popularity.

The project we introduce is aimed at extending the current functionality of Proxy Servers so that they address some of the current Web 2.0 trends as well as Next Generation Web requirements such as Web X.0. Current proxy servers make use of content as merely passive cache objects that can be reused to save bandwidth. We believe such content can be re-enforced and presented to the user in an active manner which

enables them to discover and share interests through cached content, similar to the way content is published and shared at collaboration sites such as YouTubeTM or FlickrTM, but it can take place locally and in a decentralized manner with the key difference being that the content gathered in the later was a result of user browsing patterns as opposed to content uploaded solely for the purpose of sharing. For instance a RSS TV web application could offer an RSS feed of videos that were downloaded and cached via the Bassa server as well as schedule external RSS feeds to grab content from (see more on RSS TV in the Applications section). Similarly a YouTube like application can offer downloaded videos by re-encoding and embedding them in a web interface.

In this paper we describe Next Generation Internet caching servers as a platform for building web services and applications on top of to enhance the value of caching servers by enabling the user to make better use of cached data. We do this by making use of existing Caching Server infrastructure such as SQUID and other technologies such as P2P, instead of re-inventing the wheel. We call this project "Bassa", which means Owl in Sinhala [21], to reflect that the system will enable knowledge to be gathered from a previously neglected source of data. We will use the terms cache object, cached content and cached file interchangeably to identify any file that is cached in a caching server.

II. ARCHITECTURE

Cache objects in current caching servers are not accessible from external applications directly or otherwise in any meaningful manner since they are stored on disk using a custom format unique to that caching server. On the otherhand, Bassa will store the actual data along with its meta data that is described in a Relational Database Management System (RDBMS) and provide access to it via Web Services. This is a key difference in Bassa when compared with current caching servers.

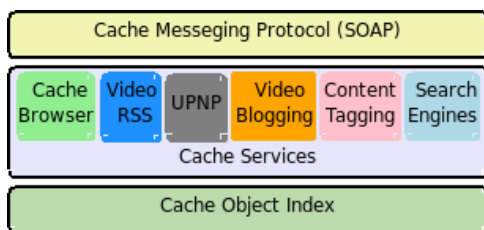


Figure. 1. CMS Layers

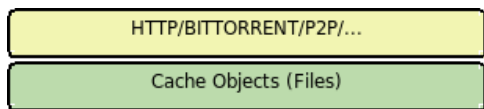


Figure. 2. OTS Layers

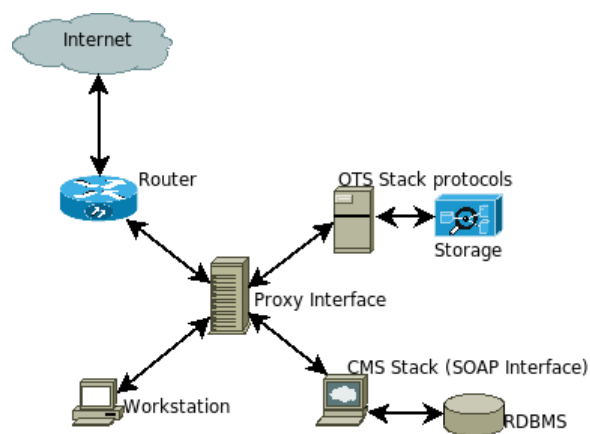


Figure. 3. Bassa Deployment Diagram

Bassa caching server consists of two layered stacks. The first is the Cache Messaging Service (CMS) that consists of a set of web services used for communicating with clients to offer discovery, query and exchange of application specific messages. The second layered stack is the Object Transport Service (OTS) that is used for transferring of cached objects between the client and Bassa. Both of these layered stacks are designed with extendability in mind to offer multiple implementations. For instance, CMS can be transported over standard http traffic, via XMPP or even possibly over a Gnutella like P2P network. Likewise OTS can make use of http, ftp, XMPP or BitTorrent to deliver the Cached Objects.

Figure.1 describes how layers of the CMS are organized. The first layer is the cache object index, which keeps the mapping between a URI and the corresponding local cache object. This is implemented using a RDBMS to store the mapping data for efficient storage and retrieval purposes. The next layer is the middleware that implements services for accessing Cache Object meta data. This layer will expose a web services API using SOAP in the form of the Cache Communication Protocol Layer.

As proof of concept, we have already developed and tested several web X.0 applications that utilize these layered stacks in Bassa. We are currently exploring more interesting applications that will make use of other social communication infrastructures such as FacebookTM [18] and Open Social [17]. Possible applications in this space will be discussed further in the Applications section.

An XML based meta data will be gathered and updated for each Cache Object based on its origin, headers, MIME type and usage statistics. Bassa operates under two modes, as defined by the caching policy consisting of an online mode for downloading content immediately and an offline mode for scheduled downloads when network is underutilized. A policy can be based on several factors ranging from allowed download times, availability of bandwidth to being next in a First In First Out (FIFO) queue. Other parameters such as prioritizing downloads depending on user, user type or content type can also be accommodated into this policy framework.

Bassa will encompass a policy framework that supports an authentication mechanism based on OpenID [16] or other Single Sign On (SSO) system. Traditional authentication mechanisms such as HTTP proxy and SOCKS based authentication will be supported to enable interoperability with standard clients. Policy level access will be developed to regulate how Cache Objects are used by different users/clients. The policy framework will encompass a cache replacement mechanism which will work in tandem with other cache expiration algorithms [4] to define how and when Cached Objects expire.

The Cache Services layer consists of a module management system for Bassa. Module management system performs loading, unloading and memory management functions of different modules that implement and extend functionality of Bassa. A service in Bassa, is made of one or more modules implemented as Unix Dynamic Shared Objects (DSO). This layer registers functions that corresponds to different services provided by the module within cache messaging protocol layer.

OTS will use several transport methods such as HTTP, FTP and Bittorrent for transporting cache objects between different nodes of the system. This service performs the same type of proxy/caching server functionality that was traditionally done but in a much broader manner. For instance using the HTTP transport method, Bassa can provide a web server like interface consisting of a URL that points directly to the cached object or provide a torrent file for that same object. In providing these enhanced transports, Bassa will support backward compatibility with traditional proxy clients. However depending on Bassa's mode of operation, being online or offline caching the user experience will vary. It is only in the online mode that Bassa will accurately model the modern Internet caching server. Figure.2 describes the layering of OTS stack.

Figure.3 shows how components of Bassa can be deployed over the network. The OTS and CMS stacks are hidden from the user when Bassa is functioning as a caching server. Even though the CMS is shown as a single web server, depending on the complexity of the applications it could be distributed among multiple servers for better scalability.

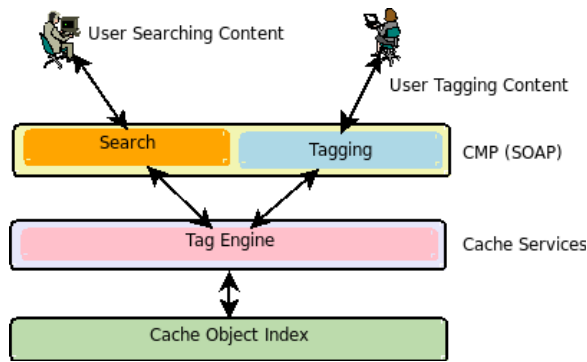


Figure. 4. Cache Tagging and Searching Through CMS Layers

III. APPLICATIONS

A. Cache Tagging

We have implemented several services as a proof of concept (POC) and these will be presented in this paper. At the end of this section we will mention other interesting applications which demonstrate the strength of a cache server as a platform for developing web X.0 applications.

Due to the large amount of content that resides in proxy servers it is useful to build services for searching cached content. To achieve this we have developed a service that runs in the Cache Services Layer of CMS and which enables external applications to retrieve or update a set of tags for a given cache object. External interfaces that use this service could also search for cache objects that match a given set of tags. We have developed a web interface that demonstrates the tag based search capabilities with the ability to download matching Cached Objects using a generated URI. Figure.4 shows how content tagging and searching happens through the layers within CMS.

To minimize the possibility of Denial Of Service (DOS) attacks by malicious external sources when browsing the cache, we have implemented a set of algorithms which return a limited number of results instead of all the results. The procedure we developed is highlighted by the following algorithms. Algorithm 1 is used by the client when making a request and Algorithm 2 used by the server when responding.

B. RSS-TV

RSS-TV is a standard for distributing videos such as podcasts over the Internet using Really Simple Syndication (RSS). RSS-TV has been implemented on PC software and devices like set top boxes, portable video players, game consoles, broadband-connected digital video disc (DVD) players, digital video recorders (DVRs), portable video players and mobile phones [15] [14]. If all these devices try to access video resources, chances are high that the same video file will be downloaded multiple times over the Internet, resulting in wasted bandwidth, which could impact both origin server and the local network.

RSS-TV application could offer a RSS feed of videos that were downloaded and cached via the Bassa server as well

```

n ← getObjectCount();
lastObj: where 0 < lastObj < n ;
offset ← getOffset();
S1: where 0 ≤ S1 ≤ lastObj ;
i ← 1;
while i ≤ lastObj do
  S1 ← S2 + 1;
  if lastObj - i < offset then
    | S2 ← S2 + lastObj - i;
  else
    | S2 ← S2 + offset + 1;
  end
  O ← getObjectList(S1, S2);
  i ← i + O.getListLength();
end

```

Algorithm 1: Client Process

```

while true do
  R ← getRequest();
  S1 ← getStartPos();
  S2 ← getEndPos();
  if S1 - S2 > Offset then
    | S2 ← S1 + offset;
  else
    | L ← readList();
    | R.setList(L);
    | R.sendResponse();
  end
end

```

Algorithm 2: Service Process

as provide a web based interface for users to subscribe to RSS-TV feeds from other sources on the web. By using cached files as a source for the video repository, Bassa's RSS-TV modules can generate RSS feeds that complies with the RSS-TV standard. The RSS-TV application on Bassa is implemented as a service running in the Cache Services Layer in CMS. This service acts as both a server and client to maximize the benefits of RSS-TV with large object caching. Access rights to these video files will be governed by policies defined by those users who downloaded the video. This was done in order to ensure the privacy of the proxy user.

C. Video Blogging

Video blogging has become popular with the introduction of web sites such as YouTubeTM [7] [19]. Video blogging is a mix of text blogging and Flash Videos (FLV) that could be played inside the browser through a FLV player. The video blogging application within Bassa is able to generate FLVs of those videos and offer it to the user, embedded in a web interface along with tagging and user comment features. Similar to the RSS-TV application, the video blogging application resides in the cache services layer of the CSM. By consuming the web service provided by the video blogging module consumer can

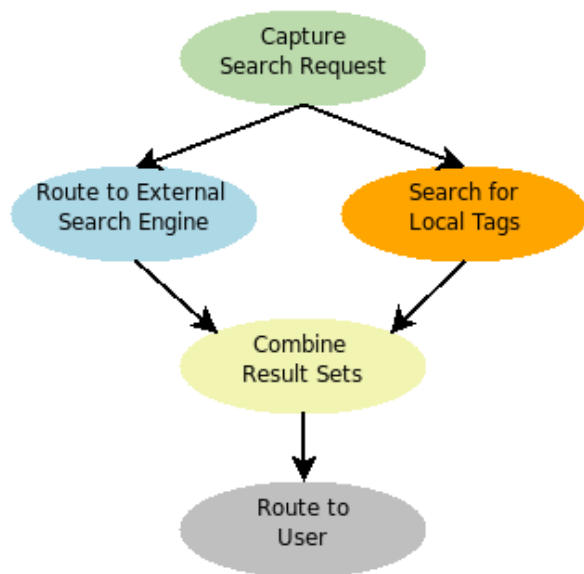


Figure. 5. Search Engine Integration

find out various meta data related to the video such as URL of the original video, the FLV formatted preview and tags (if the clip has already been tagged), etc.

In the future, video blogging sites could easily be created for local communities using their own caching server acting as a media server. This could become quite useful as community mesh networks start becoming popular, thereby enabling local communities to easily share, comment and review content consumed by others in their community. We plan on developing for FaceBookTM, an application that could interface with the video blogging service provided by Bassa.

D. Potential Applications

1) *Search Engine Integration*: Searching the cache using popular search engines is another idea that would be useful and interesting. It would be quite convenient and interesting if we can integrate this search mechanism with search engines such as GoogleTM. One way this could be done is to optimize for search engines using tags that were assigned to a particular cache object in place of keywords. We also propose a novel technique of capturing a search request sent to a search engine through the proxy and modifying the search engine result by inserting local search results that came from tag based search. This could be done by separating the proxy search results visually from original search results when rendering them on the search results page. Tags associated with an object describes the content and context of the object, and therefore we believe that this technique can become very effective for searching files on the Internet.

Figure.5 describes the above mentioned process of capturing an external search engine request, routing it to the external search engine, while searching within local tags and combining the two result sets before sending back to user.

2) *UPnP Support*: UPnP which is an abbreviation of Universal Plug and Play, is an emerging technology for smart

spaces [8] and a generic protocol for device communication, used by portable media devices and media servers. Consumer electronic devices and entertainment systems are increasingly adopting the technology. Therefore we propose to implement a UPnP module to make the cached audio and video content available for UPnP devices. This type of a module would be ideal for a caching server owned by a small community (i.e. university, a community mesh network, etc).

UPnP has two basic types of nodes that are defined as device control points and devices in it's specification [8]. Among the set of standardized UPnP DCPs, there is a specification for media server and media renderer devices [9] [10]. In this specification the device control point is referred to as a media renderer device which is a portable media player in most cases. Unfortunately implementation of this is out of our scope, and therefore we concentrate on the media server specification instead. To develop these features the cache services layer of the CMS requires a module that implements this part of the UPnP specification. Once this module has been loaded it will broadcast it's availability to renderer devices, manage connections established through cache message protocol layer and manage content directory which will be implemented using both the cache object store and cache object index layer [11]. The content transporter will be an out of band protocol other than UPnP or CMS as required [12].

E. Policy Framework

Next generation caching servers will require a comprehensive policy for handling caching of content. For instance it may be required to address copyright issues arising by sharing of content. Therefore we introduced a simple, experimental policy framework, which at the moment due to lack of time, omit to address copyright issues.

1) *Caching Policy*: Depending on the Caching Policy a Cache Object will be downloaded in an online or offline mode. If an online download occurs then the content is downloaded from the origin server and cached while simultaneously serving it to the user. On the other hand the offline mode when a user tries to download a file larger than the maximum file size allocated during the peak time, the offline mode will kick in and Bassa will present the user with a message notifying that the download request has been queued. Queued requests will be downloaded during off peak time and user will be notified with via RSS or email.

In low bandwidth networks large file downloads (i.e. DVD ISO images) done during peak hours, may result in the network crawling to a halt. But for smaller files like web pages users expect a fair response time even on low bandwidth networks, during peak hours of the network. Hence the offline policy module shifts large file downloads to off-peak hours of the network to save bandwidth for responsive web browsing during peak hours. Figure.6 shows the bandwidth utilization graph taken from MRTG [20] before offline policy was introduced, while Figure.7 shows bandwidth utilization after introduction of offline policy.

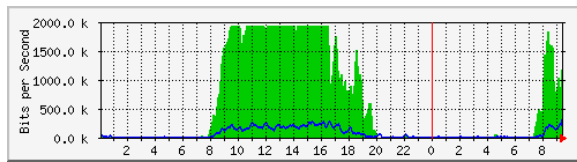


Figure 6. MRTG Graph Before Offline Policy

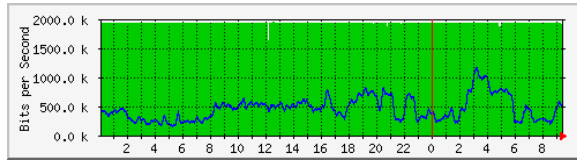


Figure 7. MRTG Graph After Offline Policy

Currently configuring the policy module requires human intervention and it works as follows. First of all, the off-peak and peak hours of the network should be identified and stored in the configuration database. Secondly the maximum file size allowed to be downloaded during peak hours should be identified and stored in the configuration database. In the future these tasks could be automated so that they will be dynamically calculated depending on usage patterns.

This feature seems to be quite useful in developing regions like Africa, some parts of Asia and Latin America where bandwidth is scarce and expensive. This feature could also be useful in community mesh networks where limited network bandwidth is shared among users of the community.

IV. CONCLUSION

The design and implementation of Bassa initially focused on handling of cached content, both in terms of policies and mechanisms. Even though there are a lot of on going research in the area of distributed caching, content pre-fetching and cache replacement policies, the area in which we operate did not seem to have been widely explored. The experience we gained by deploying Bassa at our university, The University of Colombo School of Computing (UCSC), was insightful. It proved us with an opportunity to work with a user community that was ready to experiment with sharing, viewing, tagging, searching and blogging of cached content and provide feedback for content that was otherwise neglected before the advent of Bassa.

We would like to see more research in the area of Next Generation Proxy Servers. We strongly believe that distributed caching and content pre-fetching are key ingredients of Next Generation Proxy Servers and we are looking forward to extending Bassa further in those areas as well. In the future we are keen on developing Bassa as a distributed and self organizing caching server [1] [2]. Content pre-fetching will also be addressed, to make caching process more intelligent [3]. Improvements for the policy frameworks will also be extended to handle copyright issues of content, implement cache replacement policies and to automate the manual configuring of offline policy.

REFERENCES

- [1] Scott Michel, Khoi Nguyen, Adam Rosenstein and Lixia Zhang, "Adaptive Web Caching: Towards a New Caching Architecture", *Computer Networks and ISDN Systems*, vol.30,pp.2169-2177, November 1998.
- [2] M. J. Kaiser, K. C. Tsui, J. Liu, "Self-organized Autonomous Web Proxies", in *Proceedings of the First International Joint Conference on Autonomous Agents & Multi-agent Systems*, pp. 1397-1404, IEEE, 2002.
- [3] G. Dias, G. Cope, R. Wijayaratne, "A Smart internet caching system", http://www.isoc.org.ar/inet96/proc/a4/a4_3.htm
- [4] L. Rizzo, L. Vicisano, "Replacement policies for a proxy cache", *IEEE/ACM Transactions on Networking*, vol.8,pp.158-170, April 2000.
- [5] Jim Gettys, Tim BL, and Henrik Frystyk Nielsen, "Replication and Caching Position Statement" *W3C position statement*, <http://www.w3.org/Propagation/>, 1997.
- [6] Alan Smeaton, Paul Over, Cash J. Costello, Arjen P. de Vries, David Doermann, Alexander Hauptmann, Mark E. Rovrig, John R. Smith and Lide Wu, "Indexing, Browsing and Searching of Digital Video and Digital Audio Information". *Lecture Notes in Computer Science*, vol.1980,pp.93, January 2001.
- [7] C. Parker and S. Pfeiffer, "Video Blogging: Content to the Max ", *Multimedia, IEEE*, vol.12,pp4-8, April 2005.
- [8] UPnP Working Committee, "UPnP Device Architecture 1.0", *UPnP forum*, July 2006.
- [9] UPnP members, "Media Renderer:2", *UPnP forum*, May 2006.
- [10] UPnP members, "Media Server:2", *UPnP forum*, May 2006.
- [11] UPnP members, "Connection Manager:2", *UPnP forum*, May 2006.
- [12] UPnP members, "AV Transport:2", *UPnP forum*, May 2006.
- [13] Tim O'Reilly, "What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software", September 2005, <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>.
- [14] RSS Advisory Board, "RSS 2.0 Specification", October 2007, <http://www.rssboard.org/rss-specification>.
- [15] rss-tv.org, "Simple navigation protocol for broadband media services", <http://www.rsstv.org>.
- [16] OpenID, "What is Open ID?", <http://openid.net/what/>.
- [17] Google, "The web is better when it is social", , <http://code.google.com/apis/opensocial/>.
- [18] FaceBook, "FaceBook", , <http://www.facebook.com>.
- [19] YouTube, "YouTube", , <http://www.youtube.com>.
- [20] MRTG, "Multi Router Traffic Grapher", , www.mrtg.org.
- [21] Wikipedia, "Sinhalese language", http://en.wikipedia.org/wiki/Sinhalese_language.