The Research of Time Efficiency in Adaptive Content Delivery

Haiping Gong[†], Xiang Li[††], Xiang Lin[†††] and Jianhua Li,[††††]

School of Information Security
Shanghai Jiaotong University, Shanghai, P.R.China

Summary
With different kinds of terminal clients coming up, adaptive content delivery is a hot research topic. The proxy, or the intermediary server, is often chosen to do the function of producing "adaptive" contents in real time. Admittedly, those resolutions addressed the adaptive aspect of content delivery very well. However, they did not make full use of another proxy's important character, which is caching. In this paper, we concern mainly about caching the adaptive content. The object in our consideration is JPEG2000. The proxy in our paper support storing partial object in the cache and realize adaptive increment of the quality of copy in the cache for the subsequent requests. After testing the object perception time, we conclude that the performance can be enhanced after making partial object caching supported.

Key words:
adaptive delivery, content negotiation,jpeg2000,web cache.

1. Introduction

People rely much on Internet in their daily life. They wish to have access to Internet wherever and whenever they want. Therefore, wireless network came up. The development of communication methods and manufacturing techniques challenge application on the Internet a lot. In the past few years, new devices such as small palm computers, smart phones, pocket PCs became common components of the computing infrastructure. According to some estimates cited by the W3C, by the near future, 75% of the web content access will be soon generated by these devices rather than by desktop PCs[1]. Different device has its own capability to process the data from Internet. If we still send the same data to every device. A lot of data make no sense because the device cannot display them to the clients. This is not only a waste of bandwidth, but also a waste of extra burden on client devices. In a word, we should provide appropriate quality of service to different devices. For instance, a mobile phone often has a little LCD, the resolution of its screen is normally 320*240. Even if it received a large, clear image with the size of 640*480, it can only show a quarter of it once on the screen.

Under such environment, clients using particular devices can include the particular information about their own characteristics and preferable requirements for the content they are requesting. Whenever an original server receives such request, it will make its own decision on which content type that will be sent back to the client based on its capability of providing appropriate version of contents.

Different ranges of an object are a common type of client specific requirement. For example, in plain text different range contains different information. Chapter I and Chapter II on different pages talk about different aspects. In Internet, this concept also exists. Traditional multimedia data format such as JPEG and MPEG focuses mainly on better compression ratio, visual effect, and content indexing. Data format for text and html is also defined in terms of the content meaning and semantics. However, after the emergence of new data format which is progressive or layered, the range version becomes practical. For image, JPEG2000[2] is a new data format which is progressive and very meaningful in our consideration. It is possible that we request or deliver certain data set of a complete object in Internet today. We call this mechanism Partial Object Transmission(POT). The object we refer to in this paper is JPEG2000.

In this paper, we will address network transmission and caching issues in POT. Firstly, we will suggest moving the burden on the original server to the intermediary between client and server, e.g. proxy server. Secondly, we will change traditional proxy servers to make them adaptive under the mechanism of POT. Finally, we will demonstrate the performance gain with POT by detailed experiments in an adaptive proxy server.

2. Related works

We consider HTTP[3] here in this paper. Based on HTTP/1.0[4], HTTP/1.1[5] proposes the concept of content negotiation[5], which defines the method used to deliver client specific contents according to different client preferences.

Content negotiation is the basis on which we do this research. In HTTP/1.1, it provides two kinds of negotiation: server-driven negotiation and agent-driven negotiation. As for server-driven negotiation, original server plays an important role in a transmission. It makes the best guess in sending the appropriate version of an object to the client. However disadvantage in this approach is also apparent and clearly described in HTTP

protocol. In a simple word, it put too much burden onto the original server while the burden is sometimes not very cost-effective. As for agent-driven negotiation, it needs a second request from clients to select the best version it finds in the list of initial response. This approach also wastes the resource a lot, like network bandwidth and CPU cycles. The combination of these two kinds of negotiation is called transparent negotiation.[6]

Transparent negotiation uses an intermediary server, or web cache, between original server and clients. The web cache here is referred to a proxy server with caching ability. Proxy caching has been a critical factor in providing large scale access to the web over the Internet. Today's web access patterns show frequent requests for a small number of popular objects at popular sites. Thus a popular object is transferred through the same network link once per request. In the absence of caching, this approach results in server overload, network congestion, high latency, and even the possibility of rejection of a client's request. In essence, this results in *hot-spots* forming in the network. Proxy caching provides an effective solution to all these problems by distributing the load across the network. Thus, moving the burden of content negotiation to proxy servers is a natural idea. Based on this concept, many researches were done.

In the year of 2000, a complete framework of adaptive content delivery was developed [7]. The framework includes content adaptation algorithms, client capability and network bandwidth discovery methods, and a Decision Engine for determining when and how to adapt content. There were many methods for adaptive content delivery as well. Among them, Info Pyramid, scalable coding and transcoding are the most popular methods. Using these methods, many applications were created, especially in the area of video stream. However, in these applications, proxy servers pay too much attention on the coding or transcoding of the objects. They did not mention how to store and manage these versions on the cache. The clients cannot benefit a lot from the cache between them and original server, because the hit ratio is very low under current mechanism of transparent content negotiation.

3. The Challenge and Proposed Resolution

All the problems in the existing proxy resolutions are that they do not store different versions on their local cache. There are several reasons for this situation.

- The management of different version on the cache presses additional burden onto the web cache. To maintain the coherence of the different version of one object is quite resource consuming. When receiving subsequent request, the process of searching appropriate object stored in the cache will suffer additional operation delay.

- How to deliver the most appropriate version to the client is another challenge. If a client cannot get what he really wants, it makes no sense that how fast he perceives the arrival of the object.

- Storing various versions of a particular object means a multiple of the size of one version. This will require a very large disk on the proxy cache. What is worse, some of the versions may never be referred to again in the future. But the version will stay in the cache until the cache is full and swap it out. If there are too many such versions, the efficiency of proxy cache will become low.

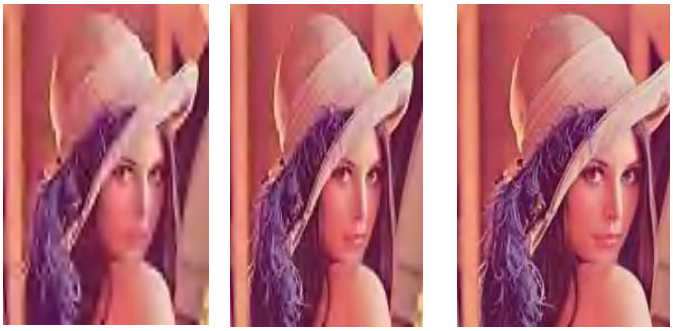
In the area of POT, proxy cache can keep only one version to achieve the effect of adaptive content delivery.

In the first place, we take a look at the algorithm of JPEG2000 and explain why it is an appropriate coding technology in the area of POT.

JPEG2000 is a bit-streaming, layered data format, with the basic layer in the front and refinement layers in the following part of the object. As a result, the quality of an image presentation under JPEG2000 is proportional to the number of first N bytes taken. This makes the delivery of JPEG2000 easy because it is simply a byte range retrieval with negligible overhead. Reusability of data among various transcoded versions of an image is also possible, as a lower quality version can be used to build a higher quality version by simply retrieving the "delta" difference in the byte range of the image from the web server. This implies a cumulative effect of image quality when it is deployed in a proxy cache. Given a fixed bandwidth and latency tolerance, the first client user gets a lower quality version of an image while later users can get better ones by combining the lower quality version in the local cache with the additional delta byte range retrieved from the server. However, traditional algorithm like JPEG did was not progressive and cannot have such an effect. The comparison of the visual effect between JPEG and JPEG2000 can be shown below:

0-1KB 0-2KB 0-4KB

Fig. 1 The visual effect of JPEG when different parts have been retrieved.



0-1KB 0-2KB 0-4KB

Fig. 2 The visual effect of JPEG2000 when different parts have been retrieved.

However, traditional proxy cache does not support partial object to be stored. On the other hand, if a proxy retrieved a response from an original server, it will check the status code of the response. If it finds a 206(partial object) code, it will sign the response as not cacheable. After forwarding the response to the client, there will be no copy on local cache of the partial object.

Actually it is not defined by the HTTP protocol, as in HTTP 1.1 all the cache related control mechanism is defined by cache control parameters. HTTP 206 is cacheable and should be able to benefit the subsequent visitors. The reason why the partial object can not be cached in most of the web caches today is that when receiving subsequent request from client, the proxy will have to forward the request to the original server once more, even though it has done the same action only seconds before. The function of cache storing on the proxy is undermined in the area of POT.

In this paper, we are trying to modify the caching mechanism and operations in traditional proxy to support partial object. Before that, we give the delivery mode and some definitions below.

The data stream delivered in HTTP protocol is in the mode of chunk-by-chunk. A browser is designated to display one chunk at once after it receives it. Although the whole object may not be received completely, the received part can be perceived by clients. In order to accurately define the retrieval time of a particular object, we give some definitions below:

Definition1: Object Retrieval time

The object retrieval time is defined as the time from the sending of a request for the object to the receiving of the last data chunk of the object.

Definition2: Object Perceived time:

The objective perceived time is defined as the time from the sending of a request for the object to the receiving of the first data chunk of the object.

It is evident that perception time is always shorter than retrieval time; at most equal in the case that only one chunk is delivered because the size of the object is very small. In the first sight, it seems that retrieval time plays an important part in measuring the performance of a content delivery system. However, actually, perception time is also important.

Firstly, from the definition of object perceived time, we can get the idea that it means the time passed by before a client perceives that he has got response from server. That is very important because if a client is forced to wait for a long time without knowing whether the server would response, he might think that something is wrong with the server, then stop the current request.

Secondly, if a client can get the first chunk data of an object as soon as possible, it is beneficial that he can sometimes get enough information contained in this object without having to retrieve the complete object, like in viewing a JPEG 2000 coded image.

From the definitions given above, we can then easily focus on the shortening of object perception time in the procedure of content delivery. The framework and improved cache we put forward here is like following:

- The client requests different parts of a jpeg2000 image from original server, without knowing the proxy server between it and the server.
- The framework is set up, namely the delivery speed between the proxy and server is almost the same as the one between proxy and client. That means the enhanced performance of cache in proxy to the delivery speed is very limited. But if we can get a little performance improvement in such an environment, we can predict that the improvement will get greater in real Internet for sure.

- The proxy is responsible of choosing the appropriate range version from the request from client. If necessary, it will forward the request to the server for the remaining part of the image.

The proxy cache is designated like this:
When a client with a low-resolution display requests the image, he can tell the server about his limitation or preference in getting this image. In particular, he can tell the range of byte he wishes to get. Then the server responds with a partial content to client. In traditional cache, this partial content will not be cached. Now we modify it to cache such a version.

When a subsequent request comes from another client, he also requests the same JPEG2000. There are two scenarios:

- Client requests a less range of bytes than the cached copy. Then the proxy can give the appropriate range to client instead of forwarding the request to the original server.

- Client requests a larger range of bytes than the cached copy. Then the proxy first gives the copy in the cache immediately. After that, it requests the remained part from server, and then gives it to the client. Meanwhile, it will append it to the end of cached copy.

This is the operational mode of our new proxy cache in the area of POT. We can see the effect in the content delivery in the next section. The whole system is built upon a P4 1G redhat linux server. The traditional proxy cache platform which used to build our modifications of caching method is SQUID.

4. Experiment and Result

4.1 Experimental Environment

The experimental environment is described in Fig. 3.

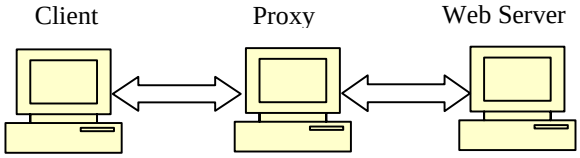


Fig. 3 Experimental Environment

The Web server installs the software of Apache, opens port 80 and publishes a JPEG2000 image on it.

The proxy is the redhat linux server with SQUID installed on it.

The client requests the JPEG2000 image on the web server via the protocol of HTTP/1.1.

4.2 Result

In the first step, we set up the proxy server to directly forward all the requests from the client to the web server. Therefore, the proxy is just acting like a router. And the cache is not used. The collected data of object perceived time is shown in Table 1 .

Table 1 Average object perceived time when client requests different part without cache. (Time Unit: us)

	100K	200K	300K	400K	463K
Average object perceived time	23075	39976	57729	76073	86803

Then, we modify the configuration on the proxy server to redirect the requests from the client to port 3128(SQUID's default port) before forwarding them. We get the data of object perceived time again and shows them in Table 2.

Table 2 Average object perceived time when client requests different part with cache. (Time Unit: us)

	100K	200K	300K	400K	463K
Average object perceived time	14995	28687	41390	57882	62191

We put the data results all together in Fig. 4 and make some comparisons.

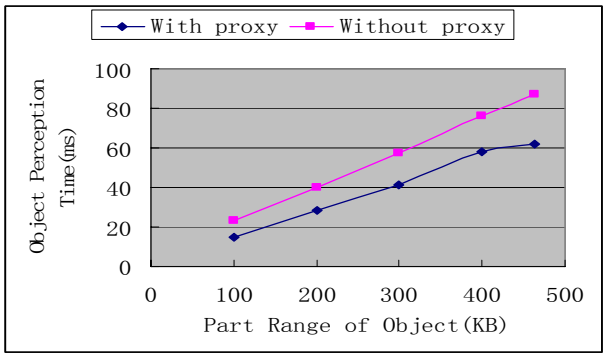


Fig. 4 Object Perceived Time when retrieving different range of an object

From Fig. 4, we can see that the object perceived time is shortened after storing the partial object on the cache. Therefore, the additional burden we give to the proxy to manage partial object can be counteracted with the benefit for the client to perceive the object more quickly.

5. Conclusion

From the experiment, we can conclude that the modified web cache can definitely enhance the performance of the proxy in POT. The increased hit ratio plays an important role. Because the subsequent requests are judged as hit in the cache, the delay of forwarding the request to the original server can be saved. From the log of SQUID, we can get enough information about this.

6. Future Works

With the development of data format and encoding algorithms, the proxy cache can work better in the future. As for the area of POT, if more progressive, layered data format can be designated, the proxy will have great effect on the content delivery efficiency.

Presently, the tendency of storing copy on a cache is one version. Although under the content negotiation mechanism, some work on the Vary Header of HTTP has been done, the efficiency cannot be guaranteed, because it seems to press too much burden on the cache and the reuse of data cannot be predicted.

References

1. Lemlouma, T. and N. Layaida. *Adapted content delivery for different contexts*. 2003.
2. Charilaos Christopoulos, A.S., Touradj Ebrahimi, 2000.
3. Thomas, S., *HTTP Essentials*. 2001.
4. T. Berners-Lee, R.F., H. Frystyk (1996) *Hypertext Transfer Protocol -- HTTP/1.0*.
5. R. Fielding, J.C.M., H. Frystyk,L. Masinter,P. Leach,T. Berners-Lee (1999) *Hypertext Transfer Protocol -- HTTP/1.1*.
6. K. Holtman, A.M., *Transparent Content Negotiation in HTTP*. 1998.
7. Wei-Ying Ma, I.B., Grace Chang, Allan Kuchinsky, and HongJiang Zhang, *A framework for adaptive content delivery in heterogeneous network environments.*

Haiping Gong received B.S. degree in Electrical Engineering in 2004, and now pursuing M.S. degree in Communication and Information System in Shanghai Jiaotong Univ., P.R.China. During his graduate study, he works as a research assistant in Prof. Li Jianhua's lab. His research interests include computer network, network security.

Xiang Li Associate Professor from Shanghai JiaoTong Univ. Major research area includes content security, network security, adaptive content delivery and data encryption. He is now leading the content security lab in Shanghai JiaoTong Univ., focusing on algorithm designing and system implementation.

NOTE:
This paper is supported by following projects:

i. Research and Application of Core Techniques in the Network Media Information Content Security Management and Control Engine(The project of Shanghai Science Committee), No.: 065115020
ii. Research of ODIE(The project of National Nature Science Fund), No.:60502032
iii. Research of Techniques in the Concept Analysis Based Text Information Monitoring on Internet(The project of National Nature Science Fund), No.:60402019