國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

博 士 論 文

CloudEdge:一個架構在雲端計算環境的內容傳遞系統

CloudEdge: A Content Delivery System in Cloud Environment

研究生: 邱繼弘
指導教授: 袁賢銘 博士

中 華 民 國 九 十 九 年 六 月

CloudEdge:一個架構在雲端計算環境的內容傳遞系統

研究生: 邱繼弘　　指導教授: 袁賢銘 博士

國立交通大學資訊科學與工程研究所

摘要

隨著 Internet 上 Web 應用程式的成長，操作網路上的影像、照片、聲音、與影片變成越來越複雜。一個以提供內容分享為主的網站系統需要大量的便宜、高效能、與高可用性的儲存空間。而因應雲端計算的趨勢，越來越多的應用程式將他們的資料內容搬遷至外部的雲端服務，以降低硬體與維護的成本。Amazon S3 就是一個專門以儲存內容為主的服務，網站開發者能將它的媒體內容資訊，交由這樣雲端服務的系統來運作。隨著大型網站應用全球化服務，如何透過在全球各地所建構的網路節點的合作，提供全球網路用戶低成本、即時、高效能的內容存取，是從單純資料儲存衍生出內容傳遞網路的課題。

在本篇論文中，將提出一種稱為 CloudEdge 的創新架構，他是一個在雲端環境中的內容傳遞系統。這個架構能夠提供遠端的內容管理系統與網站應用程式一種鬆散耦合的整合，並且比 Amazon S3 或類似服務，有如內容更新通知、邊緣網路內容遞送、內容快取、加密資料存取、版本管理、因應需求的內容變化產生器、與內容的後製處理等先進的功能。

CloudEdge: A Content Delivery System in Cloud Environment

Student: Chi Huang Chiu Advisor: Dr. Shyan Ming Yuan

Institute of Computer Science and Engineering

National Chiao Tung University

Abstract

With the growth of web applications on Internet, managing web content objects such as image, photo, audio and video files is becoming more complicated. A web system providing content sharing may need large volume of data storage which must be cheap, high-performance, and high availability. With the trend of cloud computing[1], more and more web applications move their content to external storage services like Amazon S3(Simple Storage Service) to reduce the cost of hardware and maintenance. In the other side, the access to large web applications is from nodes around the world. Providing low cost, real-time, and high efficiency content access method with the help of controlled nodes deployed in different locations is a new issue of content delivery network instead of simply content access.

In this paper, a novel architecture called CloudEdge for content delivery network with the storage service in cloud computing is introduced. This architecture could not only keep the loosely coupled integration between Storage Service and Web Applications but also provide better content manipulation features than Amazon S3 such as change notification, edge network content delivery, caching, secured access control, the variations of content objects generated on demand, and post-process on content objects.

Keywords: cloud computing; CDN; content delivery network; storage service; distributed file system; distributed system; edge network; web applications; storage service; web information system.

ACKNOWLEDGEMENTS

誌謝

在交大就學超過十五年，最要感謝的是我的指導老師 袁賢銘教授從我高中生時參加推薦甄選到博士畢業，持續的給我指導、鼓勵、與訓練。而袁老師對各種事物的看法與理解，深深的影響著我，也是我在遇到困難時，解決問題的最好方式。

此外，我也要感謝這份博士論文的口試委員：施仁忠教授、曾憲雄教授、楊竹星教授、廖婉君教授、林華君教授、與彭文志教授，你們給我的建議與指導，讓我的這份論文能更盡善盡美，也讓我學習了不少新的觀點與想法。由其是施仁忠與曾憲雄老師，你們不但在研究上給我了指導，在交大的求學過程中，也不時的給我提攜與指導，也都是我最該尊敬的恩師。

至於資訊科學系辦的楊秀琴、余美珠、陳小翠、與紀佩詩小姐，你們是我博士班求學的一個大家族，沒有你們對我的照顧與叮嚀，求學的過程會更加的艱辛。而我也要感謝分散式系統實驗室的好伙伴們，特別是林獻堂學長、吳瑞祥、鄭明俊、與蕭存喻學長，有了你們在一起，這幾年的博士班生涯過的特別充實且熱鬧，許許多多的點子，不都在我們的一同激盪下，有了火花。

更要感謝的是我的家人，爸爸與媽媽從小對我的教導、栽培、期待、與支持，是我這博士學位最大的依靠，希望從今以後我能代替你們，肩負著這家庭的重擔。最後一定要感謝的是我的摯愛 嘉慧，沒有你的愛與陪伴，這求學的過程會更加的艱辛與困難。

僅將我這論文，獻給我的家人，謝謝你們。

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1. BACKGROUND

1.1 THE EVOLUTION OF WEB INFRASTRUCTURE

The web infrastructure is the most important invention for internet. Nowadays, we use web applications to share files, read mails, buy merchandises, and even take lessons every day. These web applications in web servers use the Hyper Text Transfer Protocol (HTTP)[11] protocol to serve information and content to web browsers in clients. The design of HTTP is client-server architecture and the loading of these servers are depending on the concurrent access clients.

To avoid the repeat access on the same content that changes rarely, some cache mechanisms are provided in HTTP to help web servers control the cache on web browser. The cache on web browsers could not only reduce the access time of the client but also decrease the loading of web server. In late '90s, the bandwidth of internet is narrow especially on the backbone between different countries or campuses. The design of web proxy helps the browsers to share the cache of web content between different clients in the same local area network. All browsers use the same proxy server could share the cache in the proxy server and it could improve the user experience by decreasing the loading time of web pages.

In general, web proxy servers are deployed in client side and managed by the network of web clients. On the other side, the reverse web proxy is another kind of web proxy which is deployed in the same network of the web server to reduce the overhead and response time of web servers. These kinds of servers are widely applied in large-scale web application servers to increase the throughput of the whole system.[1]

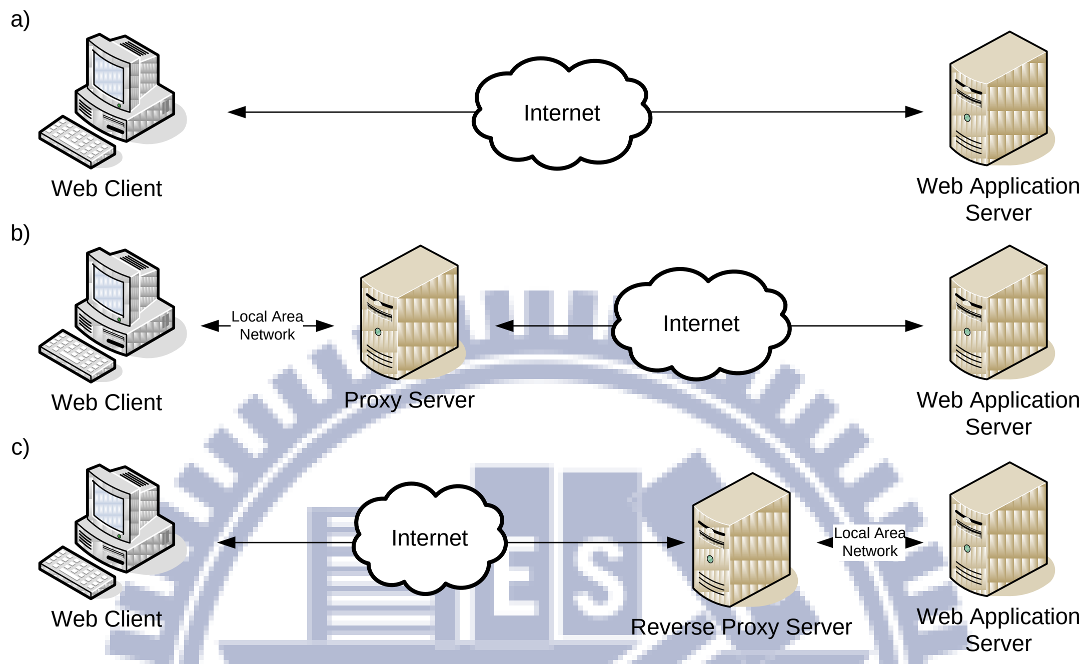


Figure 1. The proxy servers and reverse proxy servers in web architecture.

In Figure 1, the web architecture and position of proxy server and reverse proxy server are shown. The proxy server is deployed by the client-side network and could not be controlled directly from web application provider. In the design of proxy server, a simple cache and invalidate mechanism is applied, but some content in web application is dynamic or generated from different kind of sources. Therefore, some kinds of content composition in the reverse proxy in server side like Server Side Include (SSI) are introduced to support dynamic document caching.

When a web application is popular and accessed from clients distributed in different global locations, the bandwidth between the clients and the reverse proxies are the bottled neck. For instance, when there is a 10 Gbit/s network backbone and 10 Gbit/s central server capacity, only 1 Gbit/s can be delivered. But when 10 reverse proxy servers are moved to 10 different strategy locations which are near the client, total capacity can be 10*10 Gbit/s. Therefore, a client accesses a copy of the data near to the client, as opposed to all clients accessing the same central server, so as to avoid bottleneck near that server.

Such deployment of reserve proxies could be a kind of content delivery network or content distribution network (CDN). CDN is a system of computers containing copies of data, placed at various points in a network so as to maximize bandwidth for access the data from client throughout the network. The strategically placed reverse proxy servers are edge servers in CDN and the network connected between edge servers and web servers are edge networks.

The HTTP protocol for web applications is over TCP connection. The TCP throughput between two different nodes is impacted both by latency and packet loss. CDNs place servers as close to the edge network that users are on as possible. Although network distance may not be the only factor that leads to best performance. End Users will experience less jitter, fewer network peaks and surges, and improved stream quality.

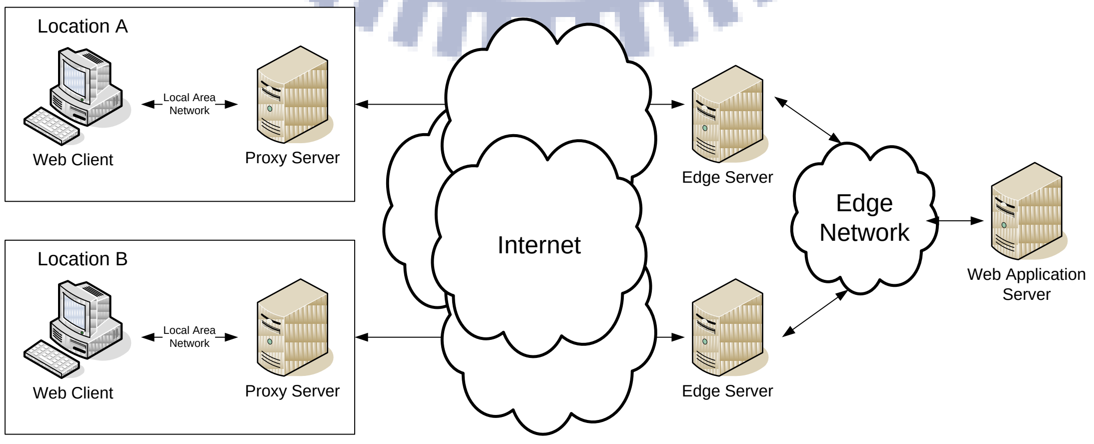


Figure 2.   Edge Servers and Edge Network

As shown in Figure 2, web clients in different locations will be redirected to nearby edge server through internet. Generally, the redirection is done by a customized DNS server which makes the decision according to the IP address of client and the availability of edge servers around the world. Since the edge server and edge network are both in private network, the communication protocols between servers are not limited to HTTP and some proprietary protocols such as multicasting could be applied. Since the edge server is managed by the web application, the storage and computing resources could also be leveraged to serve web clients.

1.2 Storage Service and Cloud Computing

Traditional web applications use a file system for both application scripts and resource files and some critical data may be stored in database servers separately. With the growth of web applications on Internet, managing web content objects like image, photo, audio and video files is becoming more complicated. In order to handle the issues of performance, availability, management and capacity, more and more web applications replace ordinary local file system with external storage services. The architecture of a typical web system is shown in Figure 3.

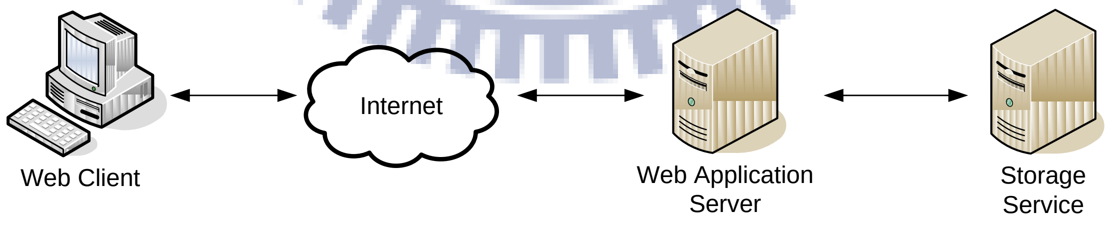


Figure 3. The architecture of a typical web system with external storage service

A web application may use a network file system as the storage service like NAS (Network Attached Storage), SAN (Storage Area Network), and DAS (Direct Attached storage)[9]. In these cases, web application servers may be the bottleneck of the performance because all published content would be accessed through web application servers. Moreover, these commercial solutions are expensive when the volume of the storage servers is huge.

Therefore some tailor-made file systems designed for web application are introduced for large-scale web applications. For example, flickr.com is a photo sharing website operated by Yahoo. It has over billions of photos and all of them are available online. To manage these huge photos, flickrFS[10], a proprietary file storage service, is applied to handle photo files. Similarly, Amazon S3 (Simple Storage Service), the first commercial online cloud services[1][3][4], is an online storage web service offered by Amazon Web Services. Amazon S3 provides unlimited storage through a simple web services interface since March 2006.

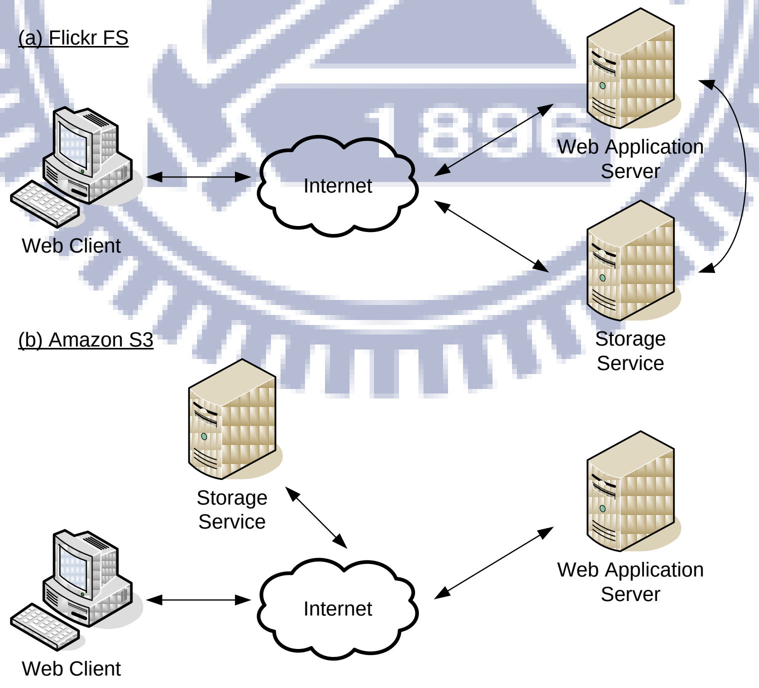


Figure 4. The architecture of web systems with storage service connected to Internet directly.

Figure 4 shows the architecture for these storage services for web applications. These storage services could not only handle the storage accesses from application servers but also serve the requests from web client directly through the HTTP protocol [11] which is the most common protocol on Internet. Such design could reduce the overhead of application server and decrease the latency of the web request. Figure 4(a) shows a common architecture from flick that the storage server and web application server are deployed in the same local area network.

In Figure 4(b), Amazon S3, the storage service is not deployed near by the web application server. The storage service is located in Internet and all operations on storage services are done in cloud. Such design could reduce the cost of maintenance for local storage service and the bandwidth and availability could be guaranteed by their huge investment.

1.3 Loosely-Coupled Integration

Conventionally, the meta information of the content is stored in the database near the web server. Web applications will use this information during generating web pages to clients, and memory cache is much important to reduce the response time on generating a HTML page. The memory cache avoids the bottleneck on I/O access but accessing the resource in cloud computing environment has long latency.

For instance, if a web page wants to embed 10 photo files in the storage service and limit the access of these photo URLs to the current user. The web application needs to send 10 requests to storage server to generate 10 unique URL links for each photo. If the storage server is in a cloud environment, the access time of each request should over 100 milliseconds. 10 photos in a pages need at least 1 second to handle the request, and 1000 photos means more than 100 seconds. In most web client, a request which take more than 100 seconds to process will be recognized as a time-out request. Generally, a web site with good user experience usually processes a request within 1 second.

Therefore, a loosely-coupled integration between web application server and storage service are required to reduce the overhead of waiting response of requests. Some mechanisms need to apply to avoid requests on generating web pages. For example, the Amazon S3 storage service uses a simple signature method in URL to solve above issue. All URLs for private content require a signature signed by a shared key to make sure the access is granted from web applications. A sample URL to access private content object in Amazon S3 is shown in Figure 5.

```
http://quotes.s3.amazonaws.com/nelson?AWSAccessKeyId=44CF9590006BF252F707
&Expires=1177363698&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D
```

Figure 5. A sample signed URL to access private content in Amazon S3 Service.

In the above sample, a signature is included and the edge server can verify it to grant the access right. In order to avoid the illegal access by sharing the URL, an expired time is assigned to avoid illegal access of that object. Moreover, the owner of the content need to pay the request fee and bandwidth cost on handling a request in cloud services like Amazon S3. The method of signature is a demonstration of loosely-coupled integration between web application servers and cloud computing services.

1.4 RELATED WORKS

CONTENT DELIVERY NETWORK

In 1998, Akamai Technologies [12] provide the first commercial content delivery service with Apple to provide media streaming service. After that, a global distributed content delivery architecture is introduced to provide a Akamai distributed content delivery system to fights service bottlenecks and shutdowns by delivering content form the Intenet's edge.[8]

In Akamai's edge platform, the product of HTTP Content Delivery provides the features of access control, cache control, content targeting, secure content delivery, and site failover. In Akamai edge platform, the Edge Side Include (ESI) is the language to do the content composition on edge server. [12] Besides, to manage the cached data in edge servers, the ESI Invalidation Protocol [13] is applied to avoid the access of expired content.

The Edge Platform provide a complete solution to help web applications to distribute existing content on web servers through the edge network and edge servers provide by Akamai Technologies. In addition, Coral CDN [14] is a similar content delivery network service but it use the P2P technologies [15][16][17]. Because Akamai and Coral CDN do not store the content in their system and work like a reverse proxy to deliver the content identified by a URL located in the original web server.

STORAGE AND CLOUD SERVICES

The Amazon Web Services [19] is a collection of web services offered over Internet by Amazon.com; and it is the most popular commercial cloud computing services. From the point view of storage service, Amazon S3 (Simple Storage Service) [20] is an online storage for web applications and it has the definition of bucket and content object. An application could create several buckets to handle different kind of content; and each content object in a bucket has a unique ID for reference.

On the other hand, Amazon Elastic Compute Cloud (EC2)[21] provides computing power to host the images of virtual server in cloud. With the combination of Amazon S3 and EC2, web applications could be deployed in Amazon Web Service to reduce the cost of deployment and prepare for uncertain capacity in the future. Besides, Amazon SimpleDB [22] and Simple Queue Service (SQS) [23] are also the related services for cloud environment. For the whole architecture of its development, Amazon Web Service would host whole web applications in its cloud environment. In CloudEdge, the design focuses on the integration between the existing web applications and external storage services in cloud environment.

INTEGRATED CDN AND STORAGE SERVICE

Amazon CloudFront[24] is a web service for content delivery. It integrates with other Amazon Web Services to give developers and businesses an easy way to distribute content to end users with low latency, high data transfer speeds, and no commitments. With the help of the CloudFront, the content stored in the Amazon S3 service could be accessed via the edge servers located in US, European, Hong Kong, Singapore, and Japan. Since the origin server and the edge server are all from Amazon, some protocols and access methods provided in Amazon S3 could also be applied in Amazon CloudFront, too.

In addition, some fully distributed network file system like Google File System (GFS) [25] or Hadoop HDFS[26] are also applied the concept of CDN architecture. These systems could store files in different globally located servers. The cache of a file in a node may be converted to the replica of the file in the whole system. Since the behaviors of these systems are the same, these file systems still could not provide HTTP accesses.

Chapter 2. PROBLEMS & OBJECTIVES

2.1 THE PROBLEMS OF STORAGE SERVICE IN CLOUD

Since the storage service in cloud is located in the Internet, the network latency and bandwidth between storage service and application server may increase the difficulty for developers to deploy such services in their system. Therefore, some loosely coupled integration designs are applied to solve some issues like access control which may be much simple in the traditional architecture without storage service in cloud.

The sample of Amazon S3 in section 1.3 demonstrates a way to do the access control but it is not good enough to fit all kind of applications to handle private content. For example, a photo sharing website may intend to block any request without correct Referral HTTP header which means that the photos are embedded in other website. In such case, current version of Amazon S3 service does not have any solutions to do such access control.

On the other head, if a client wants to access rotated version of a photo in storage service, the web application server needs to download the whole photo, rotate it and then return to web clients. This scenario shows the capability of content delivery in current storage service in cloud is not good enough and some operations should be done in storage service to reduce the cost of transmission between the service and application server.

To fulfill all the features need by web applications discussed above, there are two issues that are required to be solved in the storage services deployed in cloud environment:

- **More Complicated Access Control Mechanism**

  The signature and expiration mechanism is easy but the URL link of the files are available for all clients in the internet before the expiration of the link determined by the set expired time in the URL. More complicated access control mechanism like IP address, HTTP Cookies, and Referral Headers are required to limited the secured data and prevent the illegal access.
- **Content Manipulation in Edge Network**

  For example, a video sharing service needs to prepare different quality of video files for different devices and configurations. The encoding types of video files are different to view in a High-Definition TV, a desktop computer or a mobile phone. The storage service should have a feature to handling the different version of content object to fulfill the requests from different kind of clients. In addition, a water mark or a hidden signature is also need to be placed in the content object to avoid the copyright issues.

2.2 OBJECTIVES

In order to delivery content from storage services, the CloudEdge is proposed as a system to provide content delivery features in cloud environment. The major goals of CloudEdge are:

- To reduce the communication cost between web application and storage service.

- To provide a secured access control mechanism and make sure private content could be available only for predefined condition.
- To process the content and generate various kinds of variations based on the requests of applications.

The rest of this paper is organized as follows. The next chapter, system design, introduces concepts and overall architecture of our system. The chapter entitled component design gives design details of all major components. Chapter 5. discusses the implementation of our experimental service. Some discussions and comparisons on system design are presented in chapter 6. Conclusions and some future directions are in the last chapter.

Chapter 3. SYSTEM DESIGN

3.1 ARCHITECTURE OVERVIEW

The CloudEdge is a content delivery system for existing storage services. Its whole architecture and major components are shown on figure 6. The brief description of these components is as follows.

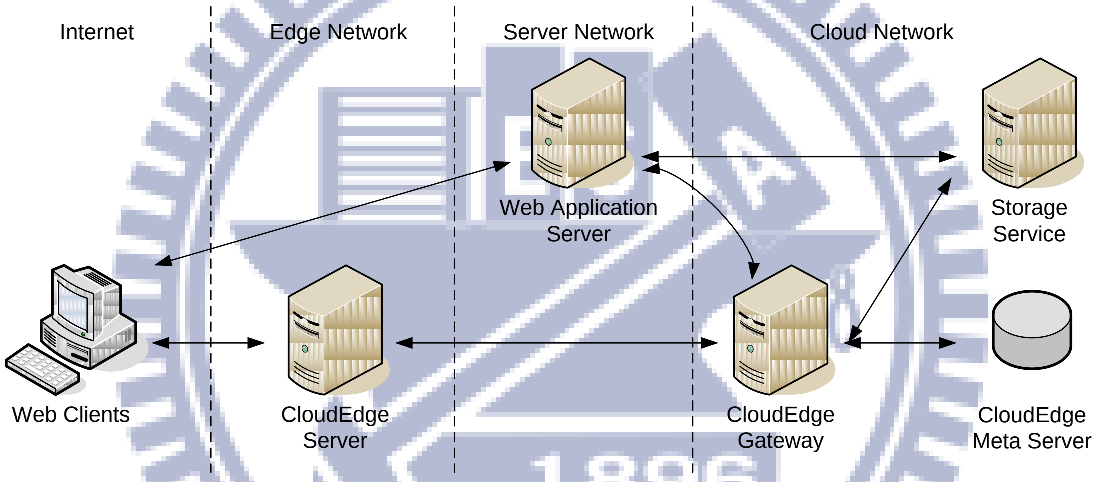


Figure 6. The Architecture of CloudEdge System

Three major components of CloudEdge system are CloudEdge Server, CloudEdge Gateway and CloudEdge Meta Server which help the system to delivery content to web clients.

The Edge Network, a term in CDN (Content Delivery Network), is a network which has high-quality connection to both cloud network and clients in some areas. On the architecture shown in figure 4, the CloudEdge Server is deployed in Edge Network to provide content from backend service to web clients. In practical, several instances of CloudEdge Server may be deployed in several locations according to geographical location and network topology. Theoretically, for web clients, the closer the content the faster the delivery. End users will likely experience less jitter, fewer network peaks and surges, and stream quality improvement- especially at remote areas.

On a web system using the CloudEdge architecture, all content requests from web client would be handled by CloudEdge server which accomplishes the following tasks for each request:

- Retrieving requested content from Storage Services through the CloudEdge Gateway and caching it in local storage to reduce the bandwidth cost.
- Performing access control to check whether the access is allowed based upon the guidance from web application server.
- Transforming the content into requested format like different size, quality or encoding format.
- Processing the content to add one-time signatures like water mark, logo, or texts.

CloudEdge Gateway, located within cloud network, is an interface to manipulate content in storage service. Web applications import their content to storage service with the help of CloudEdge Gateway instead of access the storage services directly. It means that the Gateway make the storage service transparent and different storage service could be chosen according to the requirement of application. The CloudEdge Gateway has the following services for the system:

- To verify and import the content to bound storage service.
- To handle the request from edge servers and access the storage service according to defined mapping.

- To provide the Meta information for each content bucket, a collection of content files.

Finally, the CloudEdge Meta Server stores the Meta information for Content Bucket, and it also has the program library to process the content in the CloudEdge Server. The Meta Server works as a central database of the system because all CloudEdge nodes will share the information from same CloudEdge Meta Server.

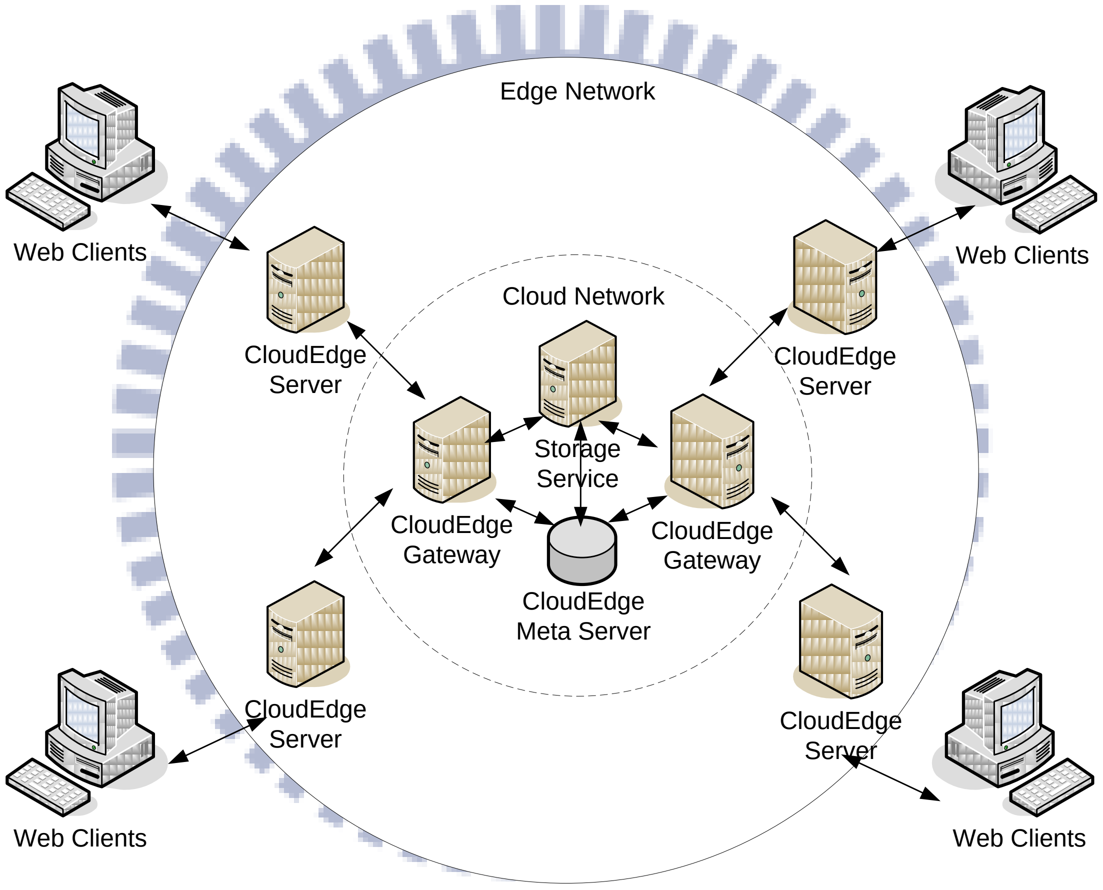


Figure 7. The architecture of multiple server instances in CloudEdge system

The architecture for multiple server instances is shown in figure 7. In this system, only one storage service and one CloudEdge Meta Server are available in the center of the system. Several CloudEdge Gateway could be deployed to increase the throughput and availability of the system. In addition, Several CloudEdge Servers are deployed in several places of Edge Network to provide better connection between the CloudEdge System and local clients. To sum up, the table 1 shows the main functions of major components in CloudEdge system.

Table 1. Main functions of major components in CloudEdge

Component	**Function Description**
CloudEdge Meta Server	The CloudEdge Meta Server has the configuration of all CloudEdge system and make sure that all nodes of the system are all consistent if the configuration is changed.
Cloud Edge Server	CloudEdge Server, located within edge network, is the access point for clients and applications.
CloudEdge Gateway	CloudEdge Gateway, located within cloud network, is an interface to manipulate content in storage service.

3.2 Object, Content Bucket and Version Control

In the CloudEdge system, the object is a minimal unit for the content which could be a photo, video, audio or other media file. Besides, in the storage service, an object would be mapped to a file or an item according to its design. In the system, each object could be accessed from a web client through the CloudEdge Server directly using the HTTP protocol.

All content objects belong to a content bucket which is a collection of content objects with the same type. Each content bucket has a unique name for reference and each object in the content bucket has a unique Object ID too. A pair of bucket name and content id could be as a reference for a content object. In addition, a content bucket could be set either public or private. All requests to content objects in a private bucket should be associated with a singed signature, otherwise, the requests would be denied.

All content objects in CloudEdge are not controlled by version but the cached objects in the CloudEdge Server and would be invalidated immediately if the update is received in the Cloud Edge Gateway. Although the cache could be invalidated, the content object could still be cached in clients or proxy servers. To avoid accessing expired content, the cache in client side could be disabled for specified bucket using the response headers in HTTP protocol. Moreover, the web application could send a version number for each content object with the bucket name and object ID to generate different URL and avoid the access of cached objects. In that case, web applications manage the cache of client side by themselves.

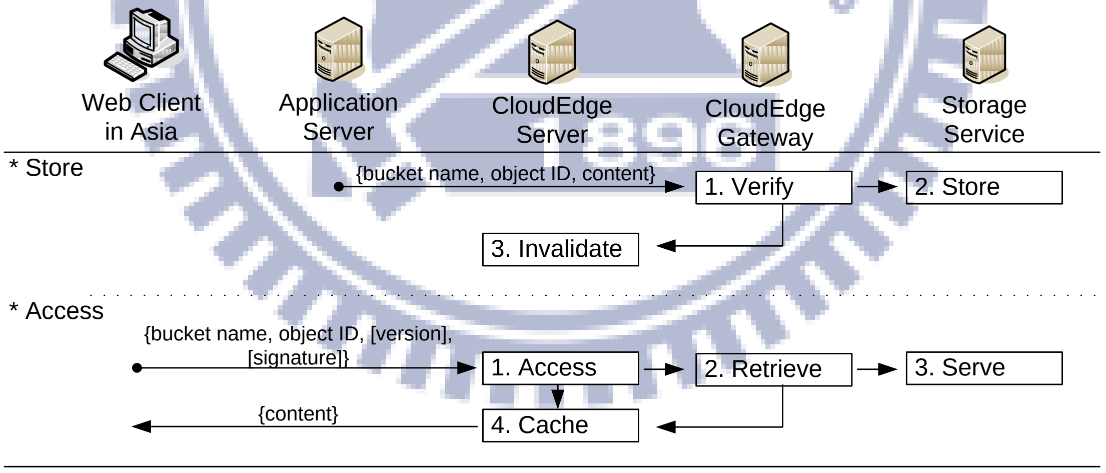


Figure 8. The process to store and access a content object.

As the figure 8 shown above, the application stores a content object into the storage service through the help of CloudEdge Gateway with a bucket name and the object ID as the identifier. After storing the content in the storage service, the related cached content in the CloudEdge Server will be invalidated immediately.

For content access, a client could use a bucket name and content ID to access a public content object and a signature is required if the bucket is private. Sometimes, requests are accomplished with version number to control the cache in clients from applications. Because the cache in the CloudEdge Server could be invalidated after modification, all request to the CloudEdge Server will check the cache first and store the result in the cache according to the configuration of the bucket.

3.3 Content Type, Variations, and Post Processing

A content bucket could have a content type to enable the features of variations and post processing on content in this bucket. A variation for a content object means a new format of this object such as different size, quality, or encoding format. For example, a video may have different encoding format for different devices like desktop, mobile phone, and portable media player; a photo may have different size for application like full size for printing, large size for slide show, and thumbnail size for previewing. In the flickr.com, an uploaded photo will be resized to different variations in the following dimensions: 75x75, 100x100, 240x240, 500x500, and 1024x1024.

For a bucket with a defined content type, all content objects imported to CloudEdge Gateways require a verification process to make sure these objects could be manipulated correctly at CloudEdge Servers. Besides, each variation of a content type has to set a generation mode which may be “pre-generated mode”, “generate on-demand mode”, or “generate and store back mode”. The difference processes for these variation generation modes is shown in figure 9.

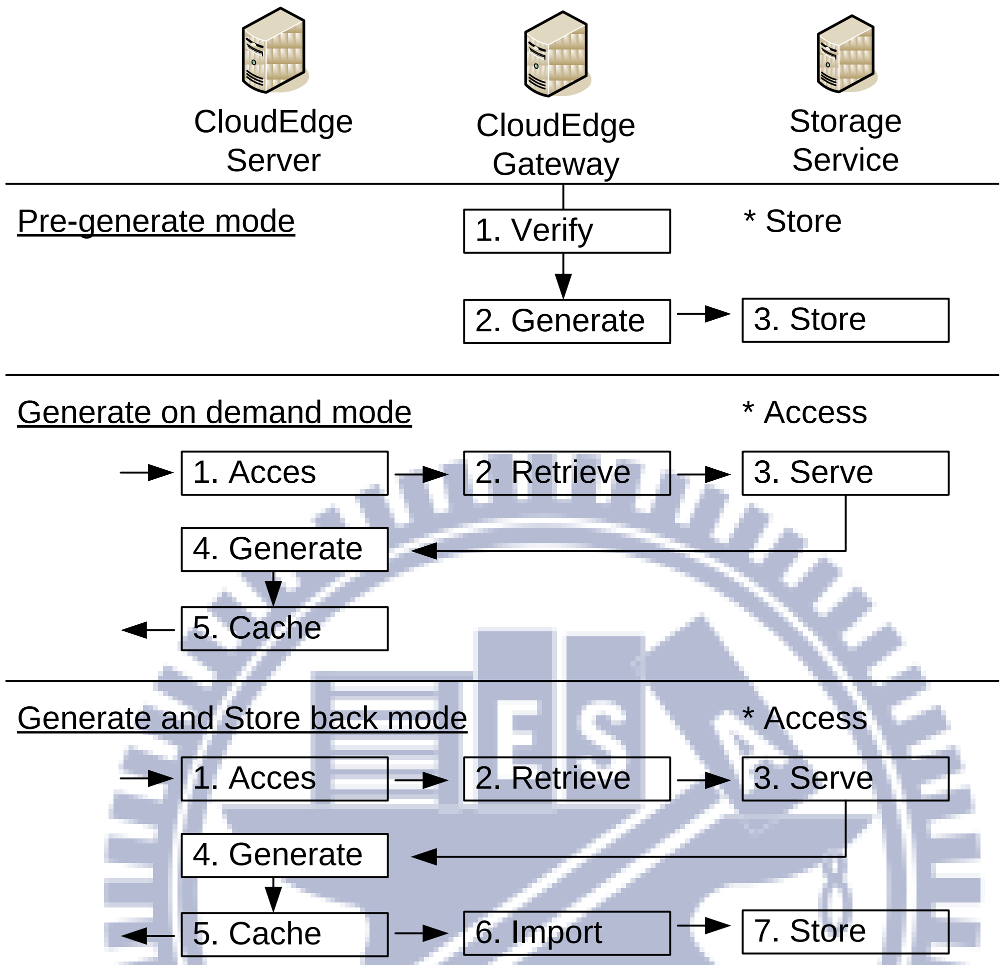


Figure 9. The processes of different variation generation mode

In "Pre-Generate Mode", a variation is pre-generated right after verifying the content object; the storage service has one item for original format and the other item for the variation. This mode is suitable for situation which the variation is accessed frequently. The "Generate on demand mode" generates a variation when a request is sent to the CloudEdge Server. This mode is designed for the variations which require few computing efforts. The third mode is "Generate and store back mode" and it is similar to the combination of previous two modes which store the generated variation in the storage service. An environment which has several cloud edge gateways could use this mode to share the generated variation to reduce the efforts for generating the same variation again. If a variation type is rarely accessed and the generation cost is high, then “Generate and Store back mode” is a better choice.

Post-Processing is a way to modify the content at the CloudEdge Server before returning the result to web clients. Applications could use this design to add some signatures, such as water marks, logos and texts, on content objects. In addition, some operations like get a range of the video or rotate the photo could be done by the post-processing mechanism. The difference between post-processing and variation is caching. In addition, the result of post-processing could not be shared with other request.

3.4 Access Control & Access Limiter

Content objects stored in CloudEdge could be public or private, and illegal accesses will be blocked at CloudEdge Server. In web applications, a web page consists of a HTML file and related content objects in which each item is processed on different requests. The relationships are the URL of each content object. To secure private content objects, a signature using a private key is applied and all URLs for secured content are signed in web applications and verified in CloudEdge Server. With the signature, any malicious change of URL would be blocked and only assigned content objects are accessible. Since the signature is signed in web application server without any interaction with CloudEdge Gatway, it does not increase the processing time except for calculating the signatures.

In the CloudEdge system, a design of “Access Limiter” is applied to secure the access of content objects. Access Limiter is an extensible module in CloudEdge Servers. Once the parameters of Access Limiter are included in the content object URL, the access will be limited according to its parameters. A Access Limiter is similar to the “expired time” in Amazon S3 metioned in section 1.2 but has more options according to the implementations like limiting the client IP, verifying the existence of HTTP cookies and even the bandwidth control. In the next chapter, the detail design of the Access Limiter is explained.

Chapter 4. COMPONENT DESIGN

4.1 CONTENT ACCESS URL

A Content Access URL is a reference to access the content in CloudEdge. A URL consists of bucket name, object ID, access modifier and signature. The syntax of the content object URL is shown in figure 10.

```
http://{CloudEdge Hostname}/{bueckt name}/{object ID}{/access modifier}*{//signature}+

#1. http://edge.cloudedge.com/publicVideo/12765
#2. http://edge.cloudedge.com/privateVideo/12765//ZnVucAop
#3. http://edge.cloudedge.com/privateVideo/12765/v3//ZnVucAop
#4. http://edge.cloudedge.com/publicVideo/12765/Vmp4
#5. http://edge.cloudedge.com/privateVideo/12765/Prange:0-10s//ZnVucAop
#6. http://edge.cloudedge.com/privateVideo/12765/Lip:140.113.23.3//SmlNeU1E
#7. http://edge.cloudedge.com/privateVideo/12765/Lonce//U21sTmVV
```

Figure 10. Syntax and samples of Content Access URL

In a Content Access URL, the CloudEdge Hostname is the server FQDN (Fully Qualified Domain Name) for CloudEdge Server. With regard to samples shown in figure 8, the #1 is the access URL for a public content in bucket “publicVideo” and object ID “12765”. Sample #2 is assigned for private content object in bucket “privateVideo”. A BASE64 encoded string after the double slashes is the signature for the whole URL.

There are 4 kinds of access modifiers for Content Access URL: version, variation, post-processing, and access limiter. The syntax of each modifier is shown in table 2. A version modifier adds version information of the content in the Access URL. The version numbers in CloudEdge are ignored. Sample #3 shows that its signature is same as #2 because the ignored version number is not included. The change of version number in URL avoids the access of cached content in proxy servers or clients. A variation modifier is a selector for different variation of content. In the #4 sample, it asks the CloudEdge server to provide the “mp4” encoding format for the assigned video content.

Table 2.   Syntax of all available access modifiers in CloedEdge

Access Modifier	**Syntax**	**Sample**
Version	/v{versionString}	#3
Variation	/V{variationString}	#4
Post-Processing	/P{module}{:{parameter}}?	#5
Access Limiter	/L{module}{:{parameter}}?	#6, #7

A post-processing modifier provides detail information to perform post-processing. The module “range” in sample #5 is applied to get the first 10 seconds of the video. Finally, the access limiter modifier, similar to post-processing modifier, is the notation to limit content access. In sample #6, an IP Address limiter is applied to control the access only from the IP parameter. The last sample #7 shows the “once” access limiter without parameter. A module limits the access only once per session.

The available modules or variations are defined in each bucket according to its content type defined in CloudEdge Meta Server. Besides, a content access URL could have several post-processing or access limiter modifiers and the order of the access limiter modifiers could be ignored.

4.2 CLOUDEDGE META SERVER

CloudEdge Meta Server is the central database of the CloudEdge System; it has the runtime information and program module repository. The information stored in the Meta Server is shown in figure 11. First, the system topologies about the access information of other CloudEdge Servers are stored. A CloudEdge system could have multiple CloudEdeg Servers or Gateways which are operated independently but all these servers need to register themselves in the CloudEdge Server to guarantee the changes of the system could be notified.

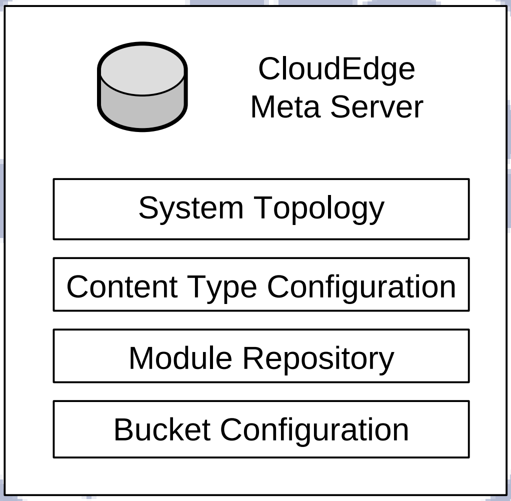


Figure 11. Information stored in CloudEdge Meta Server

Next, the content type configuration includes all available content type and the base setting of verification, variation, and post-processing for these types. Since some default configurations are defined, web application developers could also derived new configuration for their needs from them. Furthermore, the bucket configuration includes all registered bucket and its arrangements including content type, access control setting, available post-processing modules, and variation setting.

The Meta Server provides not only the runtime information and configuration but also a module repository providing the library of extensible modules. There are four kinds of extensible modules which could be executed in CloudEdge Servers and Gateways. All kind of these modules and their execution environment are explained in table 3.

Table 3. The available type of extensible modules in CloudEdge

Extensible Module Type	**Execution Envionment**	**Description**
Verification Module	CloudEdge Gateway	These modules verify the content when the web application import new content into the CloudEdge Gateway
Variation Generation Module	CloudEdge Gateway, CloudEdge Server	These modules generate required variations of content object according to the variation generation mode of the bucket.
Post-Processing Module	CloudEdge Server	These modules modify the content object in relation to the assigned parameters.
Access Limiter Moudle	CloudEdge Server	These modules control the access rights for each request to CloudEdge server.

The registration process for a CloudEdge Serversor Gateway node is as follows:

1. Add the information of the new node into the Network Topology table.
2. Synchronize all Content Type configuration and Bucket Configuration between node and Meta Server.
3. Synchronize the extensible modules which may be executed in this node.

All information exchanged in above process has version number. The Meta Server will notify all nodes according to the network topology to resynchronize information to the latest version if any update is made.

4.3 CloudEdge Gateway

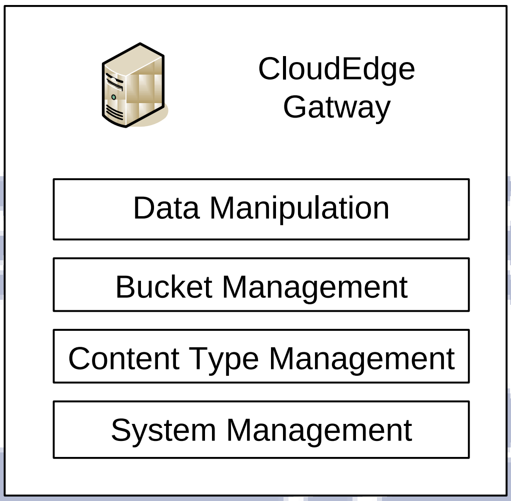


Figure 12. The interfaces in CloudEdge Gateway

The CloudEdge Gateway is the main interface for web applications to manage the whole CloudEdge System. It provides interfaces based on web service for the following operations:

- **Data Manipulate Interface**

  Like the interfaces for most storage services, the data manipulate interface could get, put and delete objects directly. It also supports the “range” options to get content in specified range.

- **Bucket Management Interface**

  This interface provides operations to create or remove a bucket, get the information and configuration of all buckets, or change the setting of specified bucket. Besides, the access control of a specified bucket including ACL and keys could be managed by the CloudEdge Gateway via this interface.

- **Content Type Management Interface**

  This Interface manages the Content Type Configuration in CloudEdge Meta Server including the relationships between content types, modules, variations, and post-processing.

- **System Management Interface**

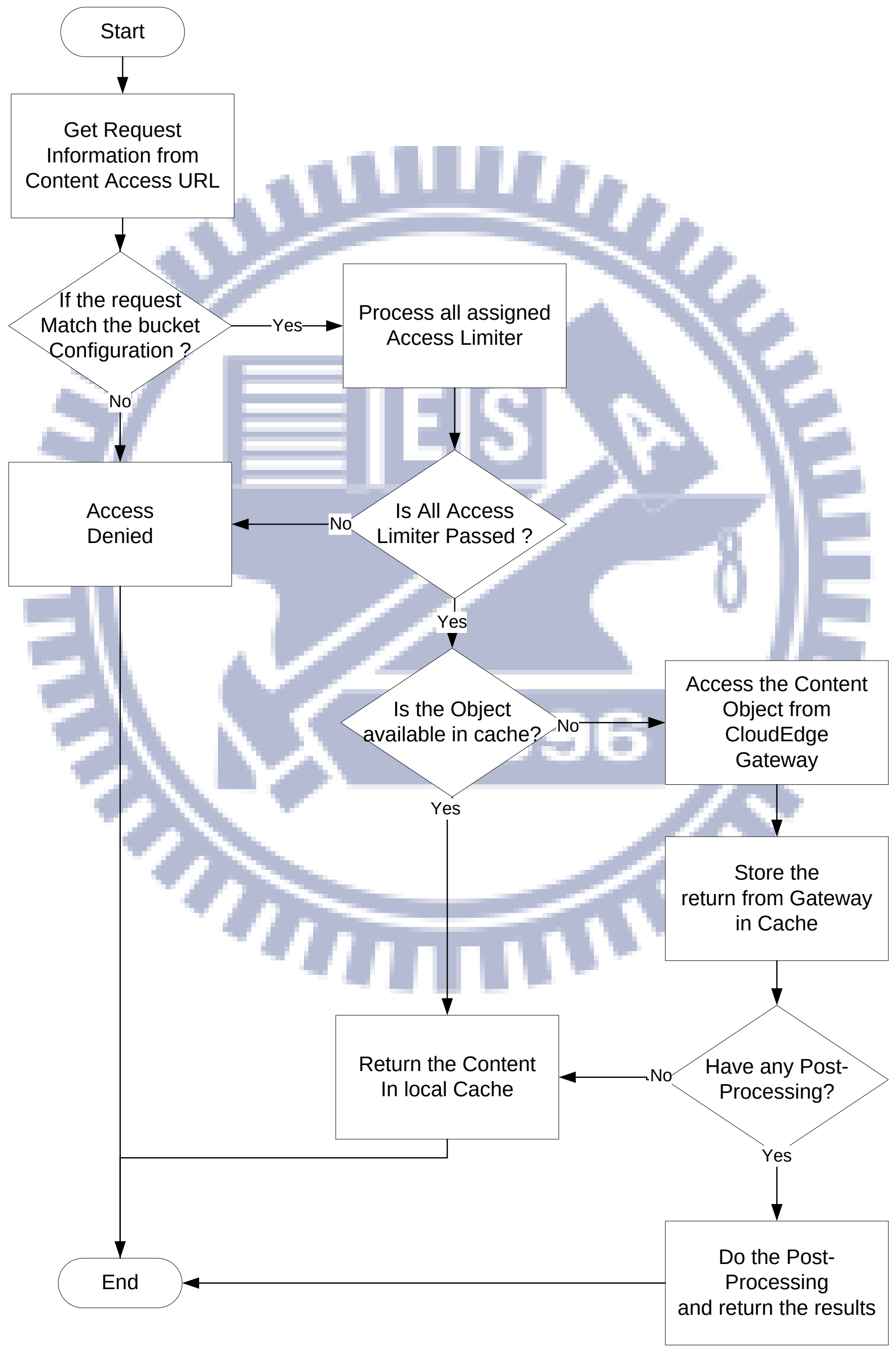
  This interface not only provides the network topology of the online system but also has the operations to get the statistics information of all nodes, buckets, and objects. It could help the web applications to monitor the performance and availability of the systems and get notification if any node is corrupted.

Any update operation in CloudEdge Gateway may trigger the Meta Server to invalidate the cache in CloudEdge Server or issue a request to synchronize the configuration in all CloudEdge Servers and Gateways. In order to avoid the race condition of the cache data or configuration, a Server Notification Queue is implemented in the CloudEdge Meta Server. Any update of content or configuration creates an item in the queue to trigger the update on all servers.

A queue is processed by a signal-thread worker and the Meta Server would ensure that acknowledgments of all servers are received before process the next item in queue. Besides, a serious object changes may generate a lot of items in queue to invalidate cache and the worker will send the all consequent update items in the queue together to reduce the cost of cache invalidation.

4.4 CLOUDEDGE SERVER

A CloudEdge Server serves request from web client and works like an edge server in a Content Delivery Network. The CloudEdge Server not only caches the data but also has a secured access control mechanism and post-processing features. The flow chart in figure 13 shows the process of handling a request in CloudEdge server.



Start
Get Request Information from Content Access URL
If the request Match the bucket Configuration ?
Yes
No
Process all assigned Access Limiter
Access Denied
Is All Access Limiter Passed ?
No
Yes
Is the Object available in cache?
No
Yes
Access the Content Object from CloudEdge Gateway
Store the return from Gateway in Cache
Return the Content In local Cache
Have any Post-Processing?
No
Yes
Do the Post-Processing and return the results
End

Figure 13. The Flow Chart for CloudEdge Server to process a Content Access URL

In the architecture of CloudEdge, there are multiple CloudEdge Server and Gateway instances. For web clients, the nearest CloudEdge Server is chosen by the result from Domain Name System according to the geographical information. For example, the DNS server will return a CloudEdge Server instance if the request is sent from a network which has the cheapest cost to connect the server. On the other head, CloudEdge Gateways have multiple instances designed for load balancing and better system availability. A CloudEdge Server or Web Application Server could choose any online gateway to reach the system.

Chapter 5. IMPLEMENTATION

5.1 ENVIRONMENT OF IMPLEMENTATION

The implementation of the CloudEdge system is based on Java platform [26][28]. The reason is portability since a Java program can be executed in all kind of operating system with a Java Runtime Environment. In some cloud computing environment, the available operating systems are limited, so the portability of CloudEdge System is increased with such implementation. Besides, the extensible modules are dynamic linked libraries and could be load and unload online. In this system, each module is implemented as a JAR file and the ClassLoader in Java Platform could manage these modules as dynamic libraries. The storage services in CloudEdge are also flexible and the system implementation has several configurations for different applications including Amazon S3, MogileFS [29], and ordinary POSIX file system.

First of all, the Amazon S3 is the first commercial storage service in cloud computing and the CloudEdge implementation could be installed in Amazon EC2 platform to accommodate the whole system in a cloud environment. Second, the MogileFS from Danga Interactive is a popular distributed file system for web. The implementation could be leveraged if a large-scale local storage is needed. Finally, the ordinary POSIX file system could also be the backend storage service of a CloudEdge system. It is the basic configuration of a small web application and it could still take the advantage of the CloudEdge like variations conversion, post-processing and access control.

5.2 FUNPHOTO: THE EXPERIMENTAL SYSTEM

The experimental system of CloudEdge is “funPhoto “a web-based photo sharing system with large volume of photos. The system shows the benefits of the CloudEdge design with verification, variation, post-processing, and access control.

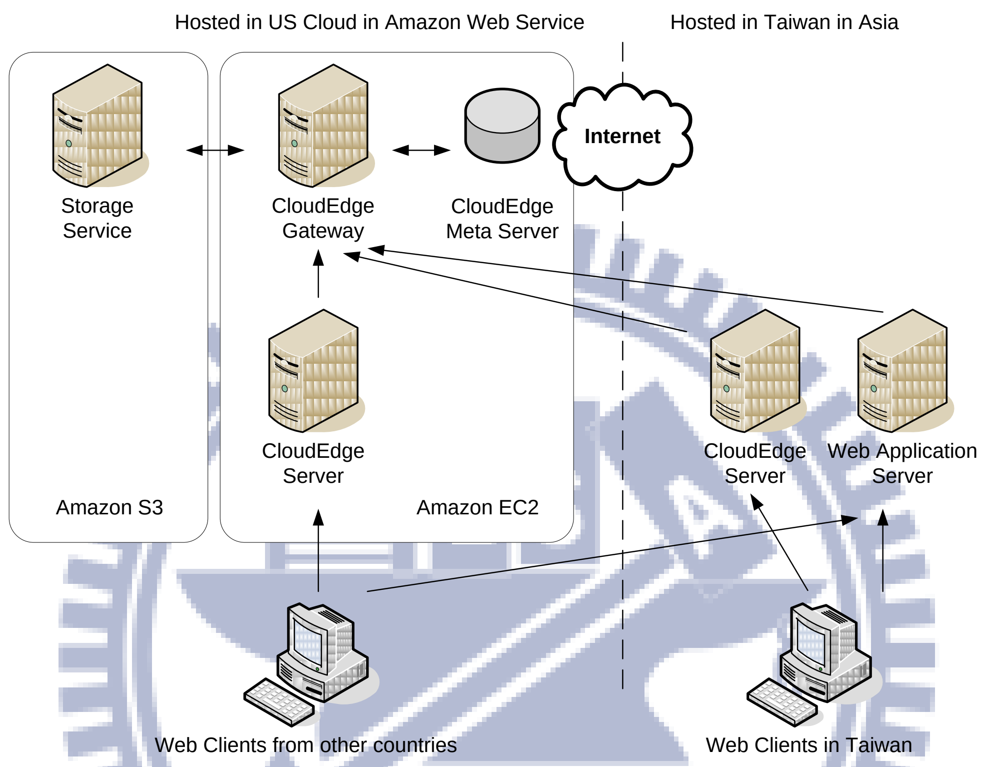


Figure 14. The architecture of funPhoto System

The architecture of the funPhoto System is shown in figure 14. First, the funPhoto use the Amazon S3 Services for photo storage; and CloudEdge Systems are also deployed in Amazon EC2, the first commercial cloud computing Platform. The funPhoto web application system is deployed in our hosting service in Taiwan because the major users of the service come from Taiwan. The novel design of the funPhoto is that it doesn’t store the photo in its machine. The funPhoto system uses a storage service in cloud. Besides, an instance of CloudEdge Server is also deployed in Taiwan to serve photo content objects and reduce the cost of bandwidth and access latency.

In funPhoto, only a content type “Photo” is applied and it has several variations including thumbnail size, small size, normal size, large size, and original size (the default content object). The implemented modules in the latest CloudEdge System comes with the funPhoto Service are shown in table 4:

Table 4. The implemented extensible modules in funPhoto service

Module	**Description**
	Verification Module
Verification for Photo	A Photo Verification Module can recognize supported photo format and convert all photo to PNG format.
	Post-Proccessing
Rotate	Rotate the photo in 90∘ CW, 180∘ CW, or 270∘ CW
Rectangle	Return the assigned portion of the photo.
Resize	Resize the photo to assigned size.
Watermark	Add invisible watermark in the photo.
Text	Add a visible text in assigned position of the photo.
	Access Limiter
Once	Let the client could access the content only in according to the session id in cookie.
ExpiredTime	Block the access after the expired time.
IP	Only allowing the assigned IP to access the photo.
HTTP Cookie	Allow the access when a valid HTTP cookie is found.
Bandwidth	Limit the daily bandwidth usage for same IP.
Referral	Check the referral header of HTTP request to avoid external usage of the photo.

Chapter 6. EXPERIMENTS & DISCUSSIONS

6.1 THE EXPERIMENT

To measure network latency in different configurations, several experiments were conducted. The scenario for these experiments is “post-processing” and in that case, watermarks were embedded into photos by web application before returning to clients. The test case was executed in three configurations: “Traditional Cloud”, a web application with an external storage service in cloud; “CloudEdge in Cloud”, a CloudEdge Server is deployed near the storage service in Cloud but the web application is deployed in the other network; and “CloudEdge in Local”, a configuration has all servers in the same network.

The environments for these three configurations are shown in figure 15. The Amazon S3 storage services are leveraged to store the content objects. The difference is that the two “CloudEdge” configurations have CloudEdge servers deployed in Amazon Elastic Compute Cloud (EC2) servers.

In the experiment environments, all local servers located in Taiwan are Dell R300 Servers with Xeon 2.4 GHz CPU & 8 GB Memory. All Amazon EC2 nodes are the Standard Reserved Instances in default (Small) configuration: 1.7 GB of memory, 1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit).

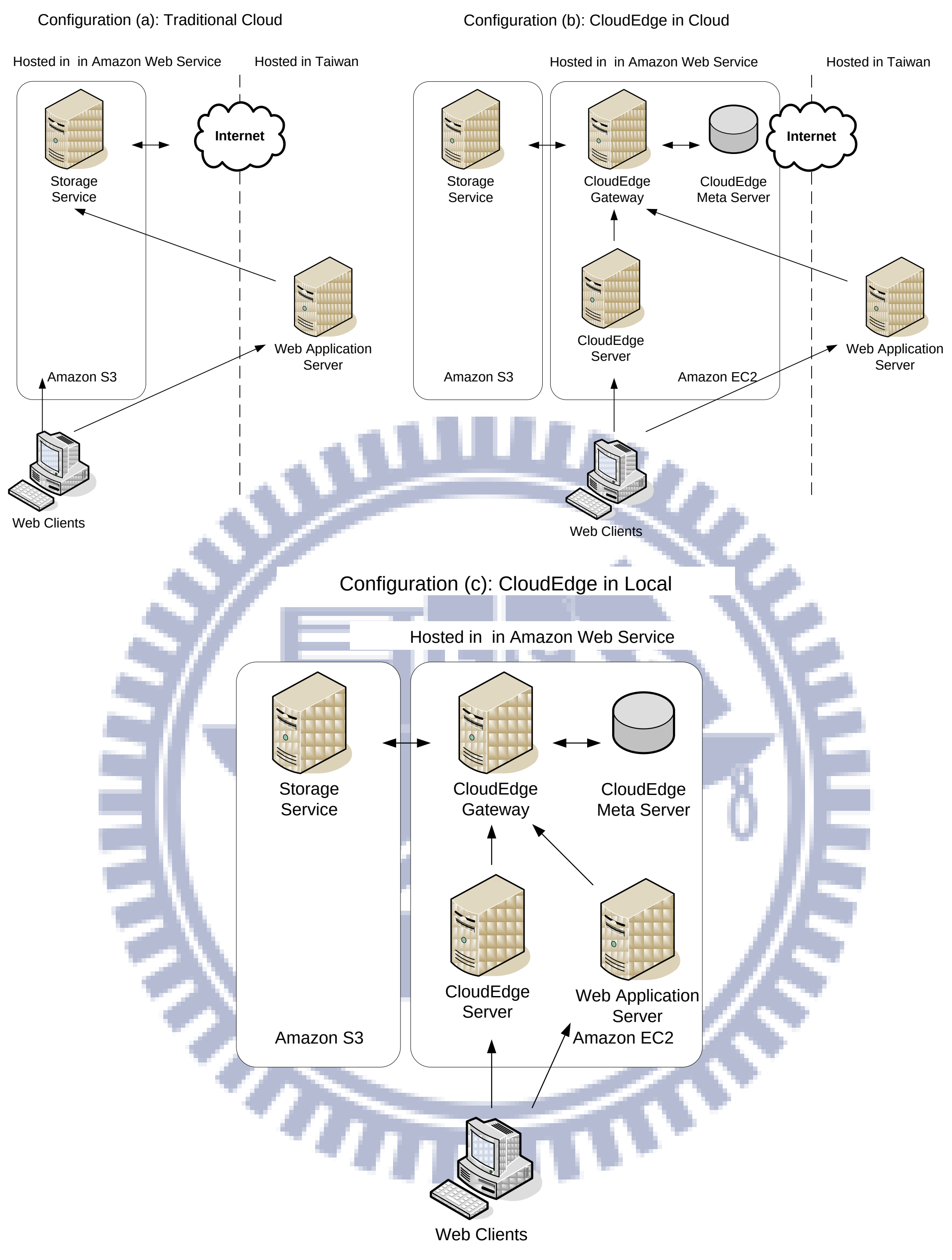


Figure 15. The deployment environment of these three configurations

In the experiment, a photo in storage service was retrieved and processed; and an invisible watermark was added to trace the illegal distribution of the photo. In the configuration (a), the web application server manipulated the photo by itself. In the rest configurations, the photos were manipulated by edge server and the details of the requested photo operations are sent by the web applications via the Content Access URL.

To measure the processing time in each step, a 650KB sample photo was accessed 10 times in these three configurations. The result is shown in table 5.

Table 5. The result of the experiment

	Retrieving the Photo from Storage Service		**Adding Water Mark**	**Total Time**
	Accessing CloudEdge Gateway	Storage Service		
(a)	-	4587 ms	601ms.	5189ms.
(b)	< 1 ms	812ms	457ms	1269 ms
(c)	< 1ms	906ms	460ms	1366 ms

By comparing the result of Configuration (a) and (b), it shows that the CloudEdge System can reduce the cost of communication when the web applications and the storage services are not in the same network. In configuration (a), the cost of retrieving content objects from storage service in cloud is expensive, but the CloudEdge could reduce it significantly since the CloudEdge Server is near the Storage Service.

For the configuration (c), it shows that the total time to process the request is similar to configuration (b). In fact, accessing the Content objects according to the Content Access URL does not require the access of web applications. It means that the CloudEdge Solution could help Web Applications to handle the content object regardless of the communication quality between storage service and web application servers.

6.2 CloudEdge: A Combination of Content Delivery Network and Storage Service in Cloud

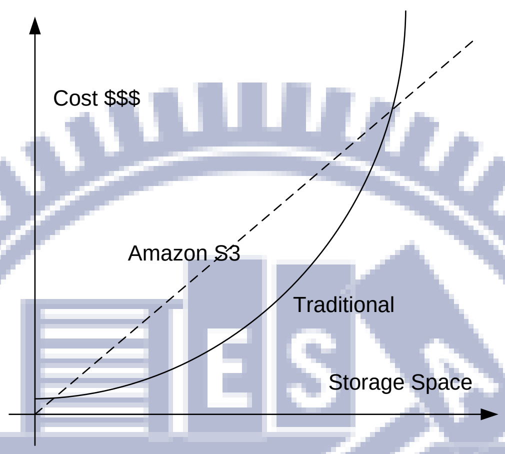


Figure 16. The relationship between storage space and cost in different type of storage.

Cloud Computing provides a new way to develop system and it will change the logic of resource management. For a storage service in cloud computing, the professional services could reduce the cost of uncertainly and maintenance. Taking Amazon S3 as an example, the cost of the storage is according to your storage space, but it costs much more in a traditional approach. With respect to huge storage, traditional approach requires high-end machines and more maintenance efforts as the chart shows in figure 16. Therefore, the use of the storage in cloud is the trend for new system development.

Not all systems could use both the Amazon S3 storage service and computing clustering service like Amazon EC2 because some data may be sensitive and the system infrastructure may be difficult to be deployed in a general environment. Recently, more and more large-scale web applications move their content objects into the storage service in cloud and the CloudEdge could be a better content delivery method than existing solutions. A comparison between CloudEdge and the Amazon Web Services, the most popular commercial cloud service, is shown in table 6.

Table 6. Comparison Between CloudEdge and Amazon Web Service

Item	**CloudEdge**	**Amazon Web Services**
Access Control	Access Limiter could limit the access according not only to signatures and expired time but also the client IP, HTTP cookies, HTTP referral header, and used bandwidth.	Only signatures and expired time.
Edge Network Support	CloudEdge Server could be deployed in any locations as the edge server in CDN.	Amazon CloudFront service has several edge services deployed worldwide.
Post-Processing	Provide Post-Processing operations on content objects.	No
Variations Control	Any content object could have several variations and be generated on demand.	No
Loosely Coupled Integration	Yes	Yes

In summary, CloudEdge is a perfect combination of Content Delivery and Cloud Computing. It could keep the loosely coupled integration between web application and storage service but provide more features required in content protection and manipulation.

6.3 Deploy CloudEdge in Local environment.

The CloudEdge system is designed to be deployed with a storage service in Cloud environment, but most small or middle-scale websites do not have such deployment architecture before its growth. The CloudEdge have a kind of configuration to use an ordinary POSIX file system as its storage service and the system could still have the benefits of content management, access control, variation on demand, and post-processing on content objects.

With the growth of the web applications, the system using the ordinary file system could be migrated to storage service in cloud environment or local distributed file systems. Moreover, the CloudEdge Server could also have multiple instances deployed in local area network to increase the throughput of the system. Once the service has heavy loading from different places around the world, the deployment of CloudEdge Server in edge network could be considered to reduce the latency of content access and bandwidth cost.

Chapter 7. CONCLUSIONS & FUTURE WORKS

7.1 CONCLUSION

The CloudEdge is a Content Delivery System for Storage Service in Cloud Environment. It's loosely coupled integration through the query string of URL could reduce the communication cost between web application server and storage services. For private content, CloudEdge has an extensible access control mechanism to meet the requirement for all kind of applications. Besides, the CloudEdge leverages the computing power in edge network to manipulating the content according to the request from server. Content object could have variations or be processed according to the assignment from Web applications. It could totally reduce the communication cost between web application and storage services. To sum up, the CloudEdge system provides a perfect combination between content delivery network and storage service in cloud environment. It could help the web application developers use an external storage in cloud with ease, and let the applications manipulate these content objects like local disk access.

7.2 FUTURE WORKS

The manipulations on content objects are always required for web applications after receiving a content object from web browser. For a photo or video uploaded from clients need to convert to the same size and data format. But currently, the web application server could not send a command to do such operations on content objects.

For example, if a web application wants to add a photo frame on a photo stored in the storage service, the web application need to download the photo, add the frame, and upload back. In the next step, the content type framework of CloudEdge will be extended to various kinds of Class Library to provide content operations in storage service from web application through SOAP[30] or REST APIs. With such improvement, managing content the storage service could be enhanced to a content management system in cloud environment.

REFERENCES

[1] Brian D. Davison, A Web Caching Primer, IEEE Internet Computing, vol. 5, no. 4, pp. 38-45, July/Aug. 2001.

[2] Brian Hayes, Cloud computing, Communications of the ACM, Volume 51, Issue 7, July 2008, p. 9-11..

[3] Armbrust, Michael and Fox, Armando and Griffith, Rean and Joseph, Anthony D. and Katz, Randy H. and Konwinski, Andrew and Lee, Gunho and Patterson, David A. and Rabkin, Ariel and Stoica, Ion and Zaharia, Matei, Above the Clouds: A Berkeley View of Cloud Computing, EECS Department, University of California, Berkeley, Feb. 2009.

[4] A. Weiss, Computing in the clouds. netWorker 11, 4, Dec. 2007. p 16-15.

[5] James Murty, Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB, O'Reilly Media, Inc., March 25, 2008

[6] Mayur R. Palankar, Adriana Iamnitchi, Matei Ripeanu, Simson Garfinkel, Amazon S3 for science grids: a viable solution?, Proceedings of the 2008 international workshop on Data-aware distributed computing, Boston, MA, USA, 2008

[7] S Saroiu, KP Gummadi, RJ Dunn, SD Gribble, HM Levy, An analysis of internet content delivery systems, Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, December 2002, p. 315-328

[8] Dilley, J.　Maggs, B.　Parikh, J.　Prokop, H.　Sitaraman, R.　Weihl, B., Globally distributed content delivery, Internet Computing, IEEE, Sep/Oct 2002, Sep/Oct 2002, p. 50-58.

[9] D. Sacks, Demystifying DAS, SAN, NAS, NAS Gateways, Fibre Channel, and iSCSI, IBM Storage Networking, 2001

[10] Manish Rai Jain. FlickrFS. http://manishrjain.googlepages.com/flickrfs

[11] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext Transfer Protocol - HTTP/1.1., [Publication]. RFC (Request for Comments), June 1999.

[12] M. Tsimelzon, B. Weihl, J. Chung, D. Frantz, J. Basso, C. Newton, M. Hale,　L. Jacobs, and　C. O'Connell, ESI Language Specification 1.0, W3C, August 2001.

[13] L. Jacobs, G. Ling, and X. Liu, ESI Invalidation Protocol 1.0. W3C, August 2001.

[14] Coral Content Distribution Network, http://www.coralcdn.com/

[15] Michael J. Freedman, Eric Freudenthal, and David Mazières, Democratizing Content Publication with Coral, In Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI'04), San Francisco, CA, March 2004.

[16] Michael J. Freedman, Mythili Vutukuru, Nick Feamster, and Hari Balakrishnan, Geographic Locality of IP Prefixes, Proc. 5th ACM SIGCOMM Conference on Internet Measurement, Berkeley, CA, October 2005.

[17] Maxwell N. Krohn, Michael J. Freedman, and David Mazières, On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution, Proc. IEEE Symposium on Security and Privacy, Oakland, CA, May 2004.

[18] Akamai Technologies. http://www.akamai.com/

[19] Amazon Web Services. http://aws.amazon.com/

[20] Amazon Simple Storage Service. http://aws.amazon.com/s3/

[21] Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/

[22] Amazon SimpleDB. http://aws.amazon.com/simpledb/

[23] Amazon Simple Queue Service. http://aws.amazon.com/sqs/

[24] Amazon CloudFront. http://aws.amazon.com/cloudfront/

[25] S Ghemawat, H Gobioff, ST Leung, The Google file system, Proc. of 19th ACM Symposium on Operating Systems Principles, Lake George, NY, October, 2003.

[26] D Borthakur. The hadoop distributed file system: Architecture and design, Hadoop Project Website, 2007. http://hadoop.apache.org/common/docs/r0.18.0/hdfs_design.pdf

[27] T. Lindholm, F. Yellin, Java virtual machine specification, Addison-Wesley Longman Publishing Co., Inc., 1999 Boston, MA, USA

[28] J. Gosling, B. Joy, G. Steele, G. Bracha, Java Language Specification: The Java Series, Addison-Wesley Longman Publishing Co., Inc., 2000 Boston, MA, USA

[29] Brad Fitzpatrick, Lisa Phillips, Inside Live Journal's Backend, November 2004, Danga Interactive, http://www.usenix.org/events/lisa04/tech/talks/fitzpatrick.pdf

[30] Box D, Ehnebuske D, Kakivaya G, Layman A, Mendelsohn N, Nielsen H, Thatte S, Winer D: Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/SOAP/