CloudEdge: a content delivery system for storage service in cloud environment

Chi-Huang Chiu*

Department of Computer Science,
National Chiao Tung University,
Hsinchu, 300, Taiwan
E-mail: chchiu@cis.nctu.edu.tw
*Corresponding author

Hsien-Tang Lin

Department of Computer Science
and Information Engineering,
Tahwa Institute of Technology,
Chulin, 307, Taiwan
E-mail: rogerlin@thit.edu.tw

Shyan-Ming Yuan

Department of Computer Science,
National Chiao Tung University,
Hsinchu, 300, Taiwan
E-mail: smyuan@cis.nctu.edu.tw

Abstract: With the trend of cloud computing, more and more web applications move their content to external storage services to reduce the cost of hardware and maintenance. In this paper, a novel architecture called CloudEdge for Content Delivery Network (CDN) 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 such as 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; Amazon Web Service; simple storage service; edge network; web applications; access control; storage service; web information system; ubiquitous computing.

Reference to this paper should be made as follows: Chiu, C-H., Lin, H-T. and Yuan, S-M. (2010) 'CloudEdge: a content delivery system for storage service in cloud environment', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 6, No. 4, pp.252–262.

Biographical notes: Chi-Huang Chiu received his BS Degree in Computer and Information Science from National Chiao Tung University in 1998, his MS Degree in Computer Information Science from National Chiao Tung University in 2000. Currently, he is pursuing PhD in the Department of Computer Science, National Chiao Tung University. His current research interests include web services, distributed systems, internet technologies, and e-learning.

Hsien-Tang Lin received his Diploma from National Kaohsiung Institute of Technology in 1983, his MS Degree in Electrical Engineering from National Taiwan University in 1991, and his PhD Degree in the Department of Computer Science from National Chiao Tung University in 2007. Currently, he is an Associate Professor at the Department of Computer and Information Science, Tahwa Institute of Technology, Hsinchu, Taiwan. His current research interests include web services, intelligent systems, internet technologies, and e-learning.

Shyan-Ming Yuan received his BSEE Degree from National Taiwan University in 1981, his MS Degree in Computer Science from the University of Maryland, Baltimore County, in 1985, and his PhD Degree in Computer Science from the University of Maryland College Park in 1989. He joined the Electronics Research and Service Organisation, Industrial Technology Research Institute as a Research Member in October 1989. Since September 1990, he has been an

Associate Professor at the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan. He has been a Professor since June 1995. His current research interests include distributed objects, internet technologies, and software system integration. He is a member of ACM and IEEE.

1 Introduction

1.1 Background

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 the internet, managing web content objects like images, photos, and 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 1.

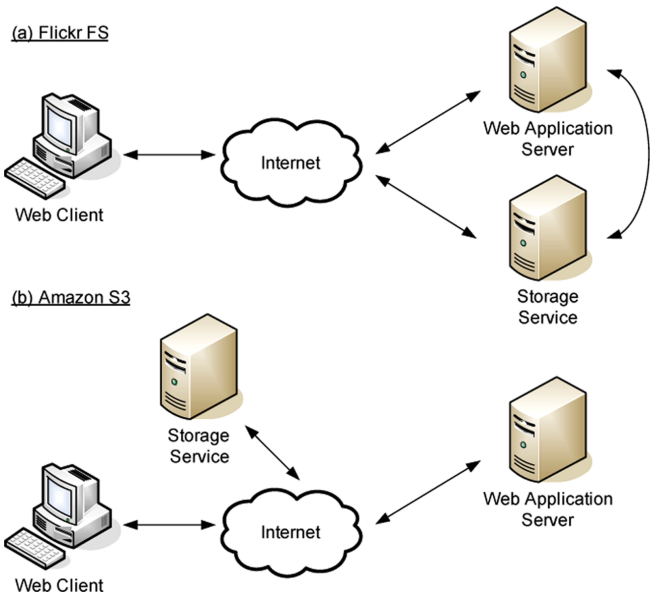
Figure 1 The architecture of a typical web system with external storage service (see online version for colours)

A web application may use a network file system as the storage service like Network Attached Storage (NAS), Storage Area Network (SAN), and Direct Attached storage (DAS). 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 (Jain, 2005), a proprietary file storage service, is applied to handle photo files. Similarly, Amazon S3 (Simple Storage Service), the first commercial online cloud services, is an online storage web service offered by Amazon Web Services. Amazon S3 has been providing unlimited storage through a simple web services interface since March 2006.

Figure 2 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 (1999) which is the most common protocol on the internet. Such design could reduce the overhead of application server and decrease the latency of the web request. Figure 2(a) shows a common architecture from flick that the storage server and web application server are deployed in the same local area network.

Figure 2 The architecture of web systems with storage service connected to the internet directly (see online version for colours)



In Figure 2(b), Amazon S3, the storage service is not deployed near by the web application server. The storage service is located in the 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.

1.2 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 very simple in the traditional architecture without storage service in cloud.

With regard to access control, the architecture with external storage service shown in Figure 1 could manage the rights of access in the web application server because all content accesses are processed in web applications. Unfortunately, such a design could not be applied to storage service in cloud because transferring the content to web applications is not economical and the bandwidth bottleneck

of the local network will increase the response time for content access.

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 3.

Figure 3 A sample signed URL to access private content in Amazon S3 Service

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

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. This sample demonstrates a way to do the access control but it is not good enough to fit all kinds of applications to handle private content. For example, a photo sharing website may intend to block any request without a correct Referral HTTP header which means that the photos are embedded in other website. In such cases, the 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 a 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 that 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.

1.3 Related works

The Amazon Web Services (2002) is a collection of web services offered over the 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) (2006) 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 kinds of content; and each content object in a bucket has a unique ID for reference. Palankar et al. (2008) even tried to apply the storage in a science grid.

On the other hand, Amazon Elastic Compute Cloud (EC2) (2006) 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 (2007) and Simple Queue Service (SQS) (2006) 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.

1.4 Objectives

In order to deliver 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.

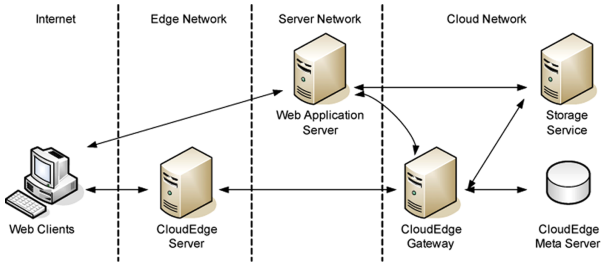
The rest of this paper is organised as follows. The next section, system design, introduces concepts and overall architecture of our system. The section entitled component design gives design details of all major components. Section 4 discusses the implementation of our experimental service. Some discussions and comparisons on system design are presented in Section 5. Conclusions and some future directions are in the last section.

2 System design

2.1 Architecture overview

CloudEdge is a content delivery system for existing storage services. Its whole architecture and major components are shown in Figure 4. A brief description of these components is as follows.

Figure 4 The architecture of CloudEdge System (see online version for colours)



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

The Edge Network, a term in Content Delivery Network (CDN) (Dilley et al., 2002), is a network which has high-quality connections 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 services to web clients. In practice,

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 the 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 the web application server
- transforming the content into the 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 the cloud network, is an interface to manipulate content in the storage service. Web applications import their content to storage service with the help of CloudEdge Gateway instead of accessing the storage services directly. It means that the Gateway makes the storage service transparent and different storage services could be chosen according to the requirement of the 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.

The architecture for multiple server instances is shown in Figure 5. In this system, only one storage service and one CloudEdge Meta Server are available in the centre of the system. Several CloudEdge Gateways 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, Table 1 shows the main functions of major components in CloudEdge system.

Figure 5 The architecture of multiple server instances in CloudEdge system (see online version for colours)

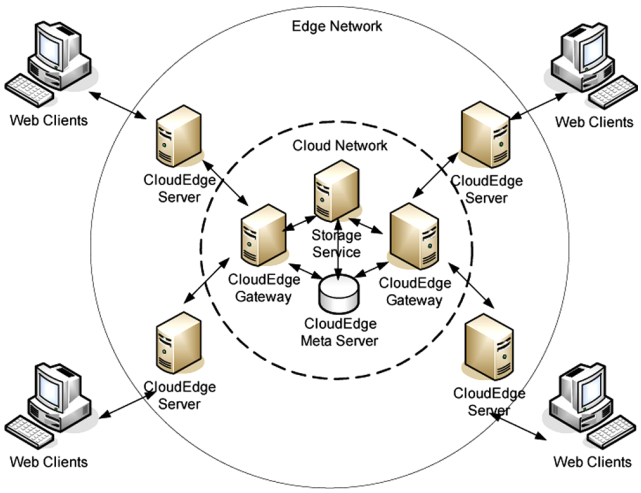


Table 1 Main functions of major components in CloudEdge

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
CloudEdge 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

2.2 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.

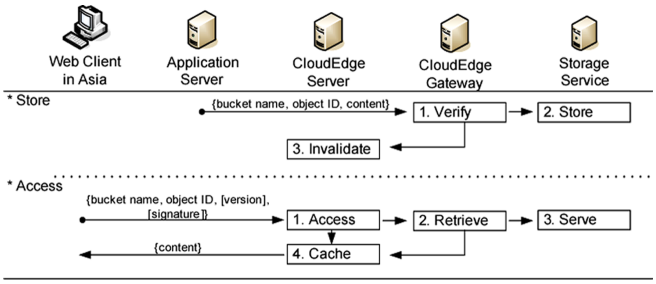
All content objects belong to a content bucket which is a collection of content objects with the same type. Each content bucket has a unique name for reference and each object in the content bucket has a unique Object ID too. A pair of bucket name and content id could be as a reference for a content 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.

As shown in Figure 6, 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.

Figure 6 The process to store and access a content object (see online version for colours)



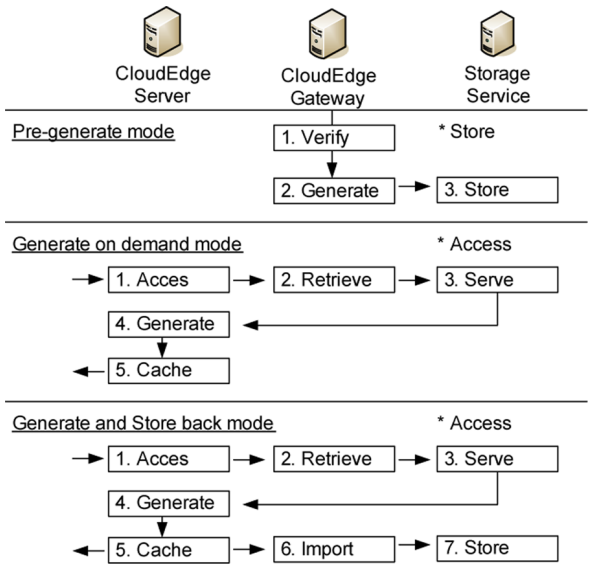
For content access, a client could use a bucket name and content ID to access a public content object and a signature is required if the bucket is private. Sometimes, requests are accomplished with version number to control the cache in clients from applications. Because the cache in the CloudEdge Server 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.

2.3 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: 75×75, 100×100, 240×240, 500×500, and 1024×1024.

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 7.

Figure 7 The processes of different variation generation mode (see online version for colours)



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.

2.4 Access control and 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 the web application server without any interaction with CloudEdge Gateway, 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. An access limiter is similar to the 'expired time' in Amazon S3 mentioned 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 section, the detailed design of the access limiter is explained.

3 Component design

3.1 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 8.

Figure 8 The syntax and samples of content access URL

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

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 four 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 CloudEdge

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

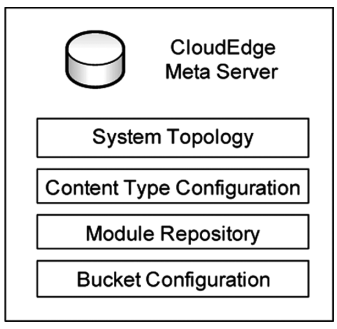
A post-processing modifier provides detailed information to perform post-processing. The module 'range' in sample #5 is applied to get the first 10 s of the video. Finally, the access limiter modifier, similar to the post-processing modifier, is the notation to limit content access. In sample #6, an 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 the content type defined in the 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.

3.2 CloudGate Meta Server

The 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 9. 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 9 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 derive new configurations 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 environment*	*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 module	CloudEdge Server	These modules control the access rights for each request to CloudEdge Server

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

1 add the information of the new node into the network topology table

2 synchronise all content type configuration and bucket configuration between node and meta server

3 synchronise the extensible modules which may be executed in this node.

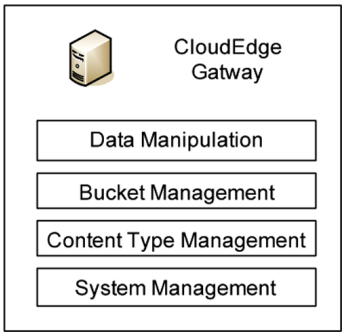
All information exchanged in the above process has a version number. The Meta Server will notify all nodes according to the network topology to resynchronise information to the latest version if any update is made.

3.3 CloudEdge Gateway

The CloudEdge Gateway is the main interface for web applications to manage the whole CloudEdge System. It provides interfaces based on the web service for the following operations which are shown in Figure 10.

- *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 the 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 a 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.

Figure 10 The interfaces in CloudEdge Gateway (see online version for colours)



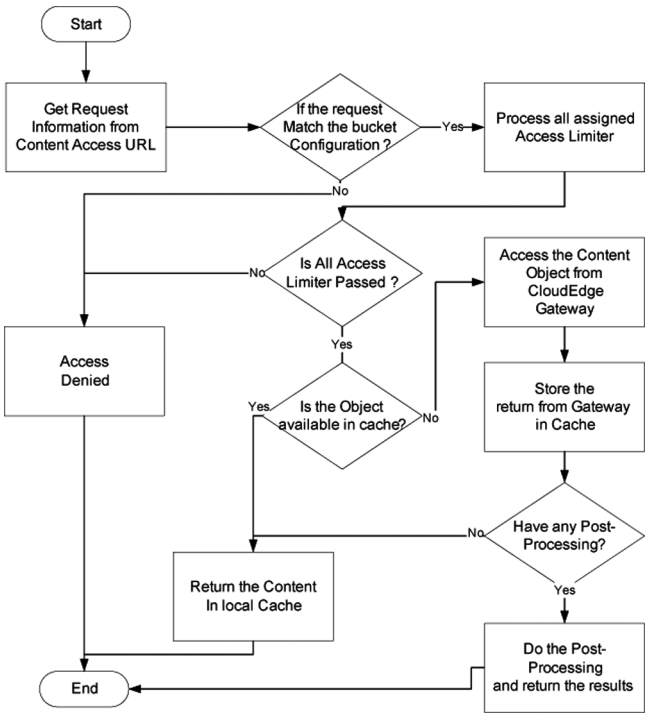
Any update operation in CloudEdge Gateway may trigger the Meta Server to invalidate the cache in CloudEdge Server or issue a request to synchronise the configuration in all CloudEdge Servers and Gateways. In order to avoid the race condition of the cache data or configuration, a Server Notification Queue 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 processing the next item in the 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.

3.4 CloudEdge Server

A CloudEdge Server serves the request from the web client and works like an edge server in a CDN. 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 11 shows the process of handling a request in CloudEdge Server.

Figure 11 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.

4 Implementation

4.1 Environment of implementation

The implementation of the CloudEdge system is based on Java platform (Java Virtual Machine 1999). The reason is portability since a Java program (Java Language Specification 2005) can be executed in all kind of operating systems with a Java Runtime Environment. In some cloud computing environments, the available operating systems are limited, so the portability of the CloudEdge System is increased with such implementation. Besides, the extensible modules are dynamic linked libraries and could be loaded and unloaded 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, and the ordinary POSIX file system.

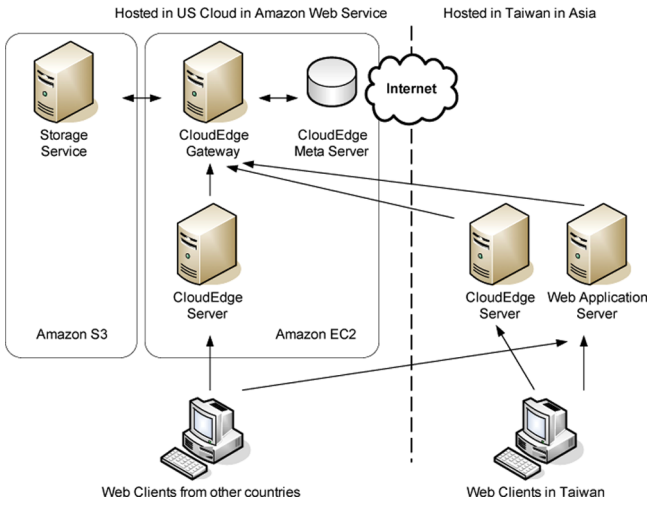
First of all, the Amazon S3 is the first commercial storage service in cloud computing and the CloudEdge implementation could be installed in the Amazon EC2 platform to accommodate the whole system in a cloud environment. Second, the MogileFS from Danga Interactive is a popular distributed file system for LiveJournal developed by Fitzpatrick (2004). 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.

4.2 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.

The architecture of the funPhoto System is shown in Figure 12. First, the funPhoto uses 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 does not 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.

Figure 12 The architecture of funPhoto system (see online version for colours)



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 recognise supported photo format and convert all photo to PNG format
Post-Processing	
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
Expired time	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

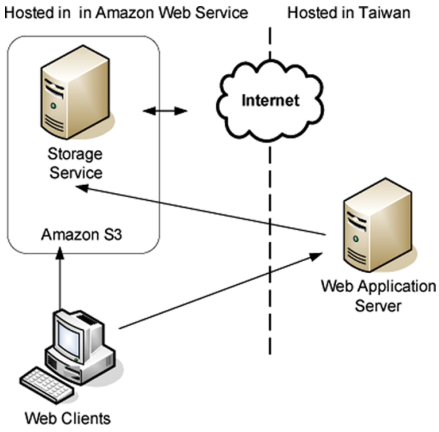
5 Experiments and discussions

5.1 The experiment

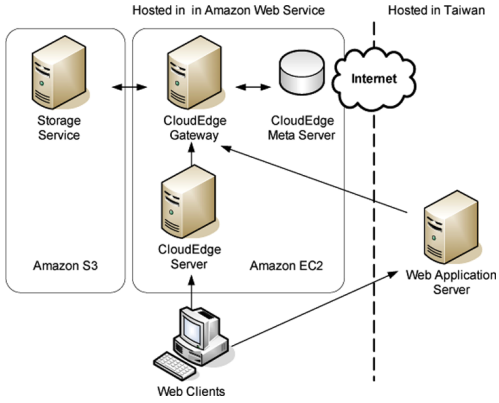
To measure network latency in different configurations, several experiments were conducted. The scenario for these experiments is 'post-processing' and in that case, watermarks were embedded into photos by web application before returning to clients. The test case was executed in three configurations: '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 13. 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.

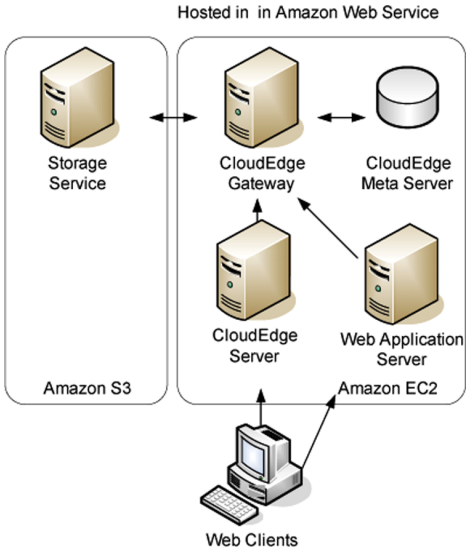
Figure 13 The deployment environment of these three configurations: (a) Configuration (a): Traditional Cloud; (b) Configuration (b): CloudEdge in Cloud and (c) Configuration (c): CloudEdge in Local (see online version for colours)



(a)



(b)



(c)

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).

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 650 KB sample photo was accessed ten 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* (ms)	*Total time* (ms)
	Accessing CloudEdge gateway	*Storage service* (ms)		
(a)	–	4587	601	5189
(b)	<1 ms	812	457	1269
(c)	<1 ms	906	460	1366

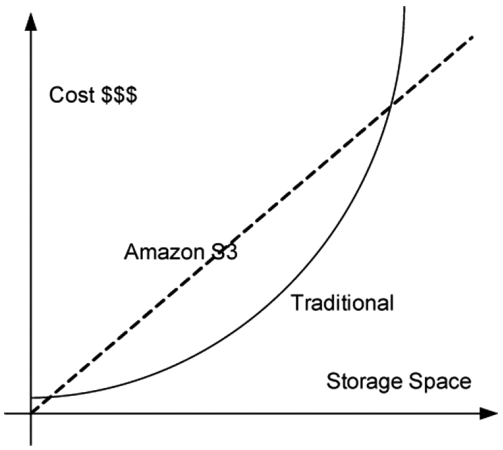
By comparing the result of Configuration (a) and (b), it shows that the CloudEdge System can reduce the cost of communication when the web applications and the storage services are not in the same network. In configuration (a), the cost of retrieving content objects from storage service in cloud is expensive,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

5.2 CloudEdge: a combination of content delivery network and storage service in cloud

Cloud Computing provides a new way to develop the 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 14. Therefore, the use of the storage in cloud is the trend for new system development.

Figure 14 The relationship between storage space and cost in different type of storage



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. Of late, more and more large-scale web applications have been moving 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.

5.3 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 their growth. CloudEdge has 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 the 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.

6 Conclusions and future works

The CloudEdge is a content delivery system for Storage Service in Cloud Environment. It is 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 kinds of applications. Besides, CloudEdge leverages the computing power in edge network to manipulating the content according to the request from the server. Content objects could have variations or be processed according to the assignment from web applications. It could totally reduce the communication cost between web applications and storage services. To sum up, the CloudEdge system provides a perfect combination between CDN 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.

The manipulations on content objects are always for handling requests; 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 needs to download the photo, add the frame, and upload it 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 (2007) or REST APIs. With such improvement, managing content the storage service could be enhanced to a content management system in cloud environment.

References

Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R. and Weihl, B. (2002) 'Globally distributed content delivery', *Internet Computing*, IEEE, September–October, pp.50–58.

Fitzpatrick, B. (2004) *Inside LiveJournal's Backend*, Open Source Convention 2004, Portland, OR, USA.

Jain, M.R. (2005) *Flickr File System*, URL: http://manishrjain.googlepages.com/flickrfs

Palankar, M.R., Iamnitchi, A., Ripeanu, M. and Garfinkel, S. (2008) 'Amazon S3 for science grids: a viable solution?', *Proceedings of the 2008 International Workshop on Data-Aware Distributed Computing*, Boston, MA, USA, pp.55–64.

Bibliography

Amazon.com (2002) *Amazon Web Services*, URL: http://aws.amazon.com/

Amazon.com (2006c) *Amazon Elastic Compute Cloud*, URL: http://aws.amazon.com/ec2/

Amazon.com (2006b) *Amazon Simple Queue Service*, URL: http://aws.amazon.com/sqs/

Amazon.com (2006a) *Amazon Simple Storage Service*, URL: http://aws.amazon.com/s3/

Amazon.com (2007) *Amazon Simple DB*, URL: http://aws.amazon.com/simpledb/

Fielding R., Mogul, J., Frystyk, H., Masinter L., Leach P. and Berners-Lee, T. (1999) *Hypertext Transfer Protocol – HTTP/1.1*, World Wide Web Consortium.

Gosling, J., Joy, B., Steele, G. and Bracha, G. (2005) *The Java Language Specification*, 3rd ed., Addison-Wesley, Boston, USA.

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau J-J., Nielsen H., Karmarkar, A. and Lafon, Y. (2007) *SOAP Version 1.2*, W3C Recommendations, URL: http://www.w3.org/TR/soap/

Hayes, B., (2008) 'Cloud computing', *Communications of the ACM*, Vol. 51, No. 7, July, pp.9–11.

Lindholm, T. and Yellin, F. (1999) *Java Virtual Machine Specification*, Addison-Wesley, Boston, MA, USA.

Murty, J. (2008) *Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB*, O'Reilly Media, 25 March.

Sacks, D. (2001) *Demystifying DAS, SAN, NAS, NAS Gateways, Fibre Channel, and iSCSI*, IBM Storage Networking.

Saroiu, S., Gummadi, K.P., Dunn, R.J., Gribble, S.D. and Levy, H.M. (2002) 'An analysis of internet content delivery systems', *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, USA, pp.315–328.