

**Bringing Into Focus Content Delivery Network
of Amazon Cloud Services Bouquet**

STARLET PUBLISHING

RZ 94, Sector - 6, Dwarka, New Delhi - 110075
Shubham Vihar, Mangla, Bilaspur, Chhattisgarh - 495001

Website: *www.starletpublishing.in*

© Copyright, 2020, Author

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form by any means, electronic, mechanical, magnetic, optical, chemical, manual, photocopying, recording or otherwise, without the prior written consent of its writer.

ISBN: 978-93-89808-21-6

Price: Rs.244.00

The opinions/ contents expressed in this book are solely of the author and do not represent the opinions/ standings/ thoughts of Starlet.

Printed in India

Bringing Into Focus Content Delivery Network of Amazon Cloud Services Bouquet

DR. RAJESH BOSE

DR. SANDIP ROY

About the Author



Dr. Rajesh Bose is an Associate Professor of the Department of Computational Science at Brainware University, West Bengal, India. He was awarded a doctorate degree in Computer Science and Engineering from University of Kalyani, India in 2018. Before completing his PhD, he had graduated with B.E. in Computer Science and Engineering from Biju Patnaik University of Technology (BPUT), Rourkela, Orissa, India in 2004. He completed his M.Tech. in Mobile Communication and Networking from Maulana Abul Kalam Azad University of Technology, West Bengal (formerly known as West Bengal University of Technology or WBUT) in 2007. Dr. Bose has industry experience of over 14 years working for one of the largest civil engineering construction companies in India helping build, modernize and maintain one of its kind Data Center. He holds several certifications on Citrix and has served on several special task groups charged with designing, developing, conducting proofs-of-concept with leading vendors in the field of Cloud Computing and IoT technologies. His research interests span Cloud Computing, Fog Computing, IoT, wireless communication, virtualization and networking. There are currently more than 50 publications to his name and with over 11 published books. A major portion of his research works have been published by leading global journals and conferences worldwide. Prior to joining industry, he held teaching positions at colleges for a number of years. His academic and professional experience accords him a unique position among his peers. In conducting research while writing this book, Dr. Bose has found consistent support from Brainware University and the staff from its Department of Computational Science of which is currently an Associate Professor.



Dr. Sandip Roy is the head of the Department of Computational Science at Brainware University, West Bengal, India and an Associate Professor. Dr. Roy was awarded his PhD in Computer Science & Engineering in 2018 by the University of Kalyani, India. Prior to his PhD, he completed his B. Tech. and M. Tech. from Maulana Abul Kalam Azad University of Technology, West Bengal (formerly known as West Bengal University of Technology or WBUT) in 2008. Dr. Roy has more than 40 research papers published in peer-reviewed journals and conferences. He is also the recipient of the Best Paper Award from ICACEA in 2015. In addition to publishing research papers, he has written several books on Cloud Computing, Fog Computing, Wireless networking, IoT and related technologies. His areas of research interests include Data Science, Internet-of-Things, Cloud Computing and Smart Technologies.

ACKNOWLEDGEMENT

This kind of book is shaped for distributing knowledge. I cannot but trust it that devoid of the benevolence of the benign God, it was perfectly preposterous to shape the bottom line of the book in the first position. I am grateful to the God as He blessed me with the power to write. Moreover, I am thankful to Him as He has gifted diligence to me to execute what this takes to provide this book with the subject-matter, structure in which this has been offered for the readers for whom the book has been connoted.

My co-writers, Dr. Sandip Roy, have been involved to provide form and essence to the book. Devoid of their priceless role, much of the research which has helped shape this book would have been preposterous. In spite of the existence of the connoisseurs in their relevant domains, they have relaxed much reliance and buoyancy in my competence to take part in the most important role in compiling this book.

Without my parents, my aims and ambition would have remained unrealized. My mother, the Late Jharna Bose, whose dream it was that I should devote my energy to inspire those who depend on me for their own academic sustenance, would have found it adequate to see fruits of my labour take shape thus. After my mother passed away, my ailing father, the Late Sisir Kumar Bose, made it a mission of his life to ensure that I never for once deviated from the goal set out before me. With their blessings and kind words which I cherish, I gathered the strength to surge through successive waves of trials and tribulations. They were the ones to realize the value of technology years before. I value their simplicity and far-sightedness which has enabled me to achieve this modest goal.

My personal contributions, however, would not have materialized if it had not been for my family who supported me in my quest and dream to write this book. I owe a great deal to my wife, Swati, who has steadfastly stood beside me in my attempts to put together the initial sketches and flow of the chapters. Without her unflinching support and dedication, it would not have been possible to expand my horizons and push forward my career.

To my son, Pablo, I offer my deepest love for having made me smile all the while. For having understood that his father needed time to write the book instead of playing games with him when he found me hammering away at a laptop keyboard. I hope one day he would realize why I spent countless weekends and holidays on this book.

We are deeply obliged and grateful to the staff, students and research scholars at the Department of Computational Sciences, Brainware University. While the ideas and assistance have been instrumental in infusing much of the foundation of this book, errors and omissions that may have crept in inadvertently are entirely the responsibility of the authors.

Last but not the least; I take this opportunity of thanking Mr. Satadru Sengupta for editing portions of this book. His tireless efforts and devotion in assisting authors pursue academic excellence and authoring books have been nothing short of remarkable.

PREFACE

More than ever, Content Delivery Networks have gained a position of prime importance among application service providers and companies offering enterprise solutions and cloud computing services. A content delivery network (CDN) can be thought of as an intricate network of computers that allows delivery of digital content and media in shortest time possible. While there have been several strategies developed to make CDN grow progressively more effective, traditional CDN approach to strategically place servers as close as possible to end-users is not consistent in terms of meeting Quality of Experience (QoE) under fluctuating network conditions.

This book is an attempt by the authors to present before readers the need and applicability of Content Delivery Networks. Among the more popular services on offer in global markets today, Amazon's cloud computing model and services that the company offers are significant in the present technological context. In the following chapters, readers shall be introduced to the world of Cloud Computing and the elasticity of Amazon's cloud content delivery network services looking at from the point of view of pay-per-use model.

Content

SECTION 1

INTRODUCING CONTENT DELIVERY NETWORK AND UNDERLYING ARCHITECTURE..... 9

CHAPTER 1

INTRODUCING CONTENT DELIVERY NETWORK (CDN) AND ITS FEATURES10

1. THE EVOLUTION OF CDNS	10
2. WHY CDN? WHAT IS A CDN?	12
Content Delivery Networks (CDN)	13
2.1. Features of Content Delivery Networks (CDN).....	13
2.2. A Typical CDN Architecture	14

CHAPTER 2

BENEFITS OF USING CONTENT DELIVERY NETWORKS AND APPLIED SYSTEMS20

1. THREE COMMON DISTRIBUTED SYSTEM.....	20
1.1. Data grids	20
1.2. Distributed databases	21
1.3. Peer-to-peer networks	21
2. USAGE OF CDN.....	22
Media Distribution or Media delivery:.....	22
Website Acceleration/Caching:.....	23
Large File/Software Delivery.....	23
3. APPLICATION OF CDN	24

CHAPTER 3

EXPLORING THE ARCHITECTURE OF CONTENT DELIVERY NETWORK25

1. LAYERED ARCHITECTURE OF CDN	25
2. PHYSICAL ARCHITECTURE CONTENT DELIVERY NETWORKS.....	27

CHAPTER 4

MANAGEMENT AND PLACEMENT OF REPLICAS IN THE CONTEXT OF CDN30

1. CONTENT CACHING TECHNIQUES	30
2. REPLICA PLACEMENT.....	31
3. CACHE CONSISTENCY AND CONTENT FLOW	31

CHAPTER 5

CONTENT DELIVERY NETWORK AND CLOUD COMPUTING32

1. DYNAMIC CONTENT MANAGEMENT	34
2. CONTENT CREATION.....	35
3. CONTENT HETEROGENEITY	35
4. CCDN OWNERSHIP.....	35
5. CCDN PERSONALIZATION	35
6. COST MODELS FOR CLOUD CDNS	35
7. SECURITY	35
8. HYBRID CLOUDS.....	36
9. CCDN MONITORING.....	36
10. CCDN QoS.....	36
11. CCDN DEMAND PREDICTION	36

12. CCDN CLOUD SELECTION	36
13. UBIQUITOUS CONTENT DELIVERY	37
14. FLEXIBLE CONTENT STORAGE, COMPRESSION, AND INDEXING	37
15. OTHER CHALLENGES.....	37

CHAPTER 6

DISCUSSING CLOUD CONTENT DELIVERY NETWORK (CCDN) IN ACADEMIC AND COMMERCIAL SETTINGS	39
1. CLOUD SECURITY:	40
2. CLOUD-BASED DNS	41
3. CLOUD STORAGE	41
4. CLOUD LOAD BALANCER	41
5. CLOUD ORCHESTRATOR.	42
6. EXISTING COMMERCIAL AND ACADEMIC CLOUD-BASED CDNS	42
Rackspace Cloud Files	42
Amazon CloudFront.....	43
MetaCDN.....	44
Limelight Orchestrate: Limelight Networks.....	45
MediaWise Cloud.....	46
SyncCast	48
Netli.....	48
Accellion.....	48
AppStream	49
EdgeStream	49
Globix	50
Akamai	50
Codeen	53
COMODIN.....	53

CHAPTER 7

EXPLORING THE TAXONOMY OF CDN.....	56
1. CDN COMPOSITION	57
2. CONTENT DISTRIBUTION AND MANAGEMENT.....	63

SECTION 2.....79

SERVICES OFFERED BY AMAZON CLOUD79

CHAPTER 8

AMAZON’S BOUQUET OF CLOUD COMPUTING SERVICES	80
1. THE DIFFERENCES THAT DISTINGUISH AWS	81
Flexible	81
Cost-Effective	82
Scalable and Elastic	82
Secure.....	83

CHAPTER 9

AN OVERVIEW OF AMAZON’S WEB SERVICES IN CLOUD.....	85
1. AMAZONE COMPUTE SERVICE	86
2. USE CASES OF AMAZONE SERVICES.....	89
3. AWS LAMBDA	90

4. AMAZON EC2 CONTAINER SERVICE	90
4. VM IMPORT/EXPORT	90
5. LICENSING MODEL.....	91
6. COMMON USES FOR VM IMPORT/EXPORT	92

CHAPTER 10

UNDERSTANDING AMAZON’S NETWORK SERVICE	93
1. AMAZON VPC	93
Features and Benefits	93
Use Cases of Amazon VPC	94
Extend your corporate network into the cloud.....	95
2. AWS DIRECT CONNECT	95
Service Highlights.....	96

CHAPTER 11

ADVANTAGES OFFERED BY AMAZON STORAGE SERVICES.....	98
1. AMAZON S3	98
Use Cases Amazon S3	98
Key Features	99
2. AWS STORAGE GATEWAY	100
The AWS Storage Gateway supports three configurations:.....	100
Benefits AWS Storage Gateway.....	101
Common Use Cases of AWS Storage Gateway	102
Benefits of Using AWS for Disaster Recovery	103
3. AMAZON GLACIER	104
Benefits of Amazon Glacier	105
4. AMAZON ELASTIC BLOCK STORE.....	106
Benefits Amazon Glacier	107
EBS Features.....	107

CHAPTER 12

EXPLORING CLOUDFRONT: AMAZON’S CONTENT DELIVERY NETWORK (CDN) SERVICES.....	110
1. AMAZON’S CONTENT DELIVERY NETWORK (CDN)	110
Cloudfront CDN Benefits	111
Common Use Cases of Cloudfront CDN	112
Distributing software or other large files	113
Delivering media files	113

CHAPTER 13

AN INTRODUCTION TO AMAZON’S DATABASE SERVICES	114
1. AMAZON RDS.....	114
Service Highlights	115
Scalable Database in the Cloud	115
Designed for use with other Amazon Web Services	116
2. AMAZON AURORA	117
3. AMAZON DYNAMODB	117
Benefit.....	117
4. AMAZON REDSHIFT.....	118
Features and Benefits Of Amazon Redshift.....	118
Optimized for Data Warehousing.....	119
No Up-Front Costs	119

Simple.....	119
Fully Managed.....	119
Fault Tolerant	120
Automated Backups	120
Fast Restores	120
Secure.....	120
5. AMAZON ELASTICACHE	121
CHAPTER 14	
AN OVERVIEW OF AMAZON’S ANALYTICS SERVICE	124
1. AMAZON EMR	124
How to Use Amazon EMR.....	124
2. AMAZON KINESIS.....	125
3. AWS DATA PIPELINE	125
CHAPTER 15	
A BRIEF OVERVIEW OF AMAZON’S APPLICATION SERVICES	126
1. AMAZON SQS.....	126
Amazon SWF	126
Amazon AppStream	127
Amazon SES	127
Amazon Elastic Transcoder.....	127
Amazon Cloud Search	128
2. MOBILE SERVICES	128
Amazon SNS	128
Amazon Cognito	128
Amazon Mobile Analytics.....	129
AWS Mobile SDK	129
3. ENTERPRISE APPLICATIONS.....	129
Amazon WorkSpaces.....	129
Amazon Zocalo	130
REFERENCES.....	131
SOME IMPORTANT QUESTION AND ANSWER:	136
INDEX.....	161

Section 1

Introducing Content Delivery Network and underlying architecture

Chapter 1

Introducing Content Delivery Network (CDN) and its Features

1. The evolution of CDNs

Over the last decades, users have witnessed the growth and maturity of the Internet. As a consequence, there has been an enormous growth in network traffic, driven by rapid acceptance of broadband access, along with increases in system complexity and content richness. The over evolving nature of the Internet brings new challenges in managing and delivering content to users. As an example, popular Web services often suffer congestion and bottleneck due to the large demands made on their services. A sudden spike in Web content requests may cause heavy workload on particular Web server(s), and as a result a hotspot can be generated. Coping with such unexpected demand causes significant strain on a Web server. Eventually the Web servers are totally overwhelmed with the sudden increase in traffic, and the Web site holding the content becomes temporarily unavailable.

Content providers view the Web as a vehicle to bring rich content to their users. A decrease in service quality, along with high access delays mainly caused by long download times, leaves the users in frustration. Companies earn significant financial incentives from Web-based e-business. Hence, they are concerned to improve the service quality experienced by the users while accessing their Web sites. As such, the past few years have seen an evolution of technologies that aim to improve content delivery and service provisioning over the Web. When used together, the infrastructures supporting these technologies form a new type of network, which is often referred to as content network.

Several content networks attempt to address the performance problem through using different mechanisms to improve the Quality of Service (QoS). One approach is to modify the traditional Web architecture by improving the Web server hardware adding a high-speed processor, more memory and disk space, or maybe even a multi-processor system. This approach is not flexible. Moreover, small enhancements are not possible and at some point, the complete server system might have to be replaced. Caching proxy deployment by an ISP can be beneficial for the narrow bandwidth users accessing the Internet. In order to improve performance and reduce bandwidth utilization, caching proxies are deployed close to the users. Caching proxies may also be equipped with technologies to detect a server failure and maximize efficient use of caching proxy resources. Users often configure their browsers to send their Web request through these caches rather than sending directly to origin servers. When this configuration is properly done, the user's entire browsing session goes through a specific caching proxy. Thus, the caches contain most popular content viewed by all the users of the caching proxies. A provider may also deploy different levels of local, regional, international caches at geographically distributed locations. Such arrangement is referred to as hierarchical caching. This may provide additional performance improvements and bandwidth saving.

A more scalable solution is the establishment of server farms. It is a type of content network that has been in widespread use for several years. A server farm is comprised of multiple Web servers, each of them sharing the burden of answering requests for the same Web site . It also makes use of a Layer 4-7 switch, Web switch or content switch that examines content request and dispatches them among the group of servers. A server farm can also be constructed with surrogates instead of a switch. This approach is more flexible and shows better scalability. Moreover, it provides the inherent benefit of fault tolerance. Deployment and growth of server farms progresses with the upgrade of network links that connects the Web sites to the Internet.

Although server farms and hierarchical caching through caching proxies are useful techniques to address the Internet Web performance problem, they have limitations. In the first case, since servers are deployed near the origin server, they do little to improve the network performance due to network congestion. Caching proxies may be beneficial in this case. But they cache objects based on client demands. This may force the content providers with a popular content source to invest in large server farms, load balancing, and high bandwidth connections to keep up with the demand. To address these limitations, another type of content network has been deployed in late 1990s. This is termed as Content Distribution Network or Content Delivery Network, which is a system of computers networked together across the Internet to cooperate transparently for delivering content to end-users.

With the introduction of CDN, content providers started putting their Web sites on a CDN. Soon they realized its usefulness through receiving increased reliability and scalability without the need to maintain expensive infrastructure. Hence, several initiatives kicked off for developing infrastructure for CDNs. As a consequence, Akamai Technologies evolved out of an MIT research effort aimed at solving the flash crowd problem. Within a couple of years, several companies became specialists in providing fast and reliable delivery of content, and CDNs became a huge market for generating large revenues. The flash crowd events like the 9/11 incident in USA, resulted in serious caching problems for some site. This influenced the CDN providers to invest more in CDN infrastructure development, since CDNs provide

desired level of protection to Web sites against flash crowds. First generation CDNs mostly focused on static or Dynamic Web documents. On the other hand, for second generation of CDNs the focus has shifted to Video-on-Demand (VoD), audio and video streaming. But they are still in research phase and have not reached to the market yet.

With the booming of the CDN business, several standardization activities also emerged since vendors started organizing themselves. The Internet Engineering Task Force (IETF) as a official body took several initiatives through releasing RFCs (Request For Comments). Other than IETF, several other organizations such as Broadband Services Forum (BSF), ICAP forum, Internet Streaming Media Alliance took initiatives to develop standards for delivering broadband content, streaming rich media content – video, audio, and associated data – over the Internet. In the same breath, by 2002, large-scale ISPs started building their own CDN functionality, providing customized services. In 2004, more than 3000 companies were found to use CDNs, spending more than \$20 million monthly. A market analysis shows that CDN providers have doubled their earnings from streaming media delivery in 2004 compared to 2003. In 2005, CDN revenue for both streaming video and Internet radio was estimated to grow at 40%. A recent marketing research shows that combined commercial market value for streaming audio, video, streaming audio and video advertising, download media and entertainment was estimated at between \$385 million to \$452 million in 2005. Considering this trend, the market was forecasted to reach \$2 billion in four year (2002-2006) total revenue in 2006, with music, sports, and entertainment subscription and download revenue for the leading content categories. However, the latest report from AccuStream iMedia Research reveals that since 2002, the CDN market has invested \$1.65 billion to deliver streaming media (excluding storage, hosting, applications layering), and the commercial market value in 2006 would make up 36% of the \$1.65 billion four-year total in media and entertainment, including content, streaming advertising, movie and music downloads and User Generated Video (UGV) distribution.

Content distribution networks (CDNs) using cloud resources such as storage and compute have started to emerge. Unlike traditional CDNs hosted on private data centers, cloud-based CDNs take advantage of the geographical availability and the pay-as-you-go model of cloud platforms. The Cloud-based CDNs (CCDNs) promote content-delivery-as-a-service cloud model. Though CDNs and CCDNs share similar functionalities, introduction of cloud impose additional challenges that have to be addressed for a successful CCDN deployment.

2. Why CDN? What is a CDN?

The Internet has enabled the delivery of content and information to the far corners of the earth. Transmitting information over long distances involves multiple intermediate steps that can cause delay or latency. In case of audio or video streaming, this delay is more significant, leading to degradation in quality and an unsatisfactory user experience. Peak traffic or demand spikes can place a huge strain on network resources that can affect speed of delivery and performance, and can sometimes bring down servers leading to service outages. CDN provides a way to overcome these challenges.

Content Delivery Networks (CDN)

This is a system of servers deployed in different geographical locations to handle increased traffic loads and reduce the time of content delivery for the user from servers. The main objective of CDN is to deliver content at top speed to users in different geographic locations and this is done by a process of replication. CDNs provide web content services by duplicating content from other servers and directing it to users from the nearest data center. The shortest possible route between a user and the web server is determined by the CDN based on factors such as speed, latency, proximity, availability and so on. CDNs are deployed in data centers to handle challenges with user requests and content routing. CDNs are used extensively by social networks, media and entertainment websites, e-commerce websites, educational institutions, etc. to serve content quickly to users in different locations.

2.1. Features of Content Delivery Networks (CDN)

Organizations by implementing CDN solutions stand to gain in many ways that include scalability, security, reliability, responsiveness and performance.

A. **Scalability** – The main business goal of a CDN is to achieve scalability. Scalability refers to the ability of the system to expand in order to handle new and large amounts of data, users and transactions without any significant decline in performance. To expand in a global scale, CDNs need to invest time and costs in provisioning additional network connections and infrastructures. It includes provisioning resources dynamically to address flash crowds and varying traffic. A CDN should act as a shock absorber for traffic by automatically providing capacity-on-demand to meet the requirements of flash crowds. This capability allows a CDN to avoid costly over-provisioning of resources and to provide high performance to every user.

B. **Security** – One of the major concerns of a CDN is to provide potential security solutions for confidential and high-value content. Security is the protection of content against unauthorized access and modification. Without proper security control, a CDN platform is subject to cyber fraud, distributed denial-of-service (DDoS) attacks, viruses, and other unwanted intrusions that can cripple business. A CDN aims at meeting the stringent requirements of physical, network, software, data and procedural security. Once the security requirements are met, a CDN can eliminate the need for costly hardware and dedicated component to protect content and transactions. In accordance to the security issues, a CDN combat against any other potential risk concerns including denial-of-service attacks or other malicious activity that may interrupt business.

C. **Reliability, Responsiveness and Performance** – Reliability refers to when a service is available and what are the bounds on service outages that may be expected. A CDN provider can improve client access to specialized content through delivering it from multiple locations. For this a fault-tolerant network with appropriate load balancing mechanism is to be implemented. Responsiveness implies, while in the face of possible outages, how soon a service would start performing the normal course of operation. Performance of

a CDN is typically characterized by the response time (i.e. latency) perceived by the end-users. Slow response time is the single greatest contributor to customers' abandoning Web sites and processes . The reliability and performance of a CDN is affected by the distributed content location and routing mechanism, as well as by data replication and caching strategies. Hence, a CDN employs caching and streaming to enhance performance especially for delivery of media content. A CDN hosting a Web site also focuses on providing fast and reliable service since it reinforces the message that the company is reliable and customer focused .

2.2. A Typical CDN Architecture

An overview of a typical CDN architecture is presented in Fig. 1. Depending on application and content type the architecture of CDNs may vary However, all CDN architectures mainly comprise of an origin server, a request redirecting mechanism and a large number of surrogate cache servers namely Point of Presence (POP).

A typical content delivery environment where the replicated Web server clusters they are located at the edge of the network to which the end-users are connected. A content provider (i.e. customer) can sign up with a CDN provider for service and have its content placed on the content servers. The content is replicated either on-demand when users request for it, or it can be replicated beforehand, by pushing the content to the surrogate servers. A user is served with the content from the nearby replicated Web server. Thus, the user ends up unknowingly communicating with a replicated CDN server close to it and retrieves files from that server.

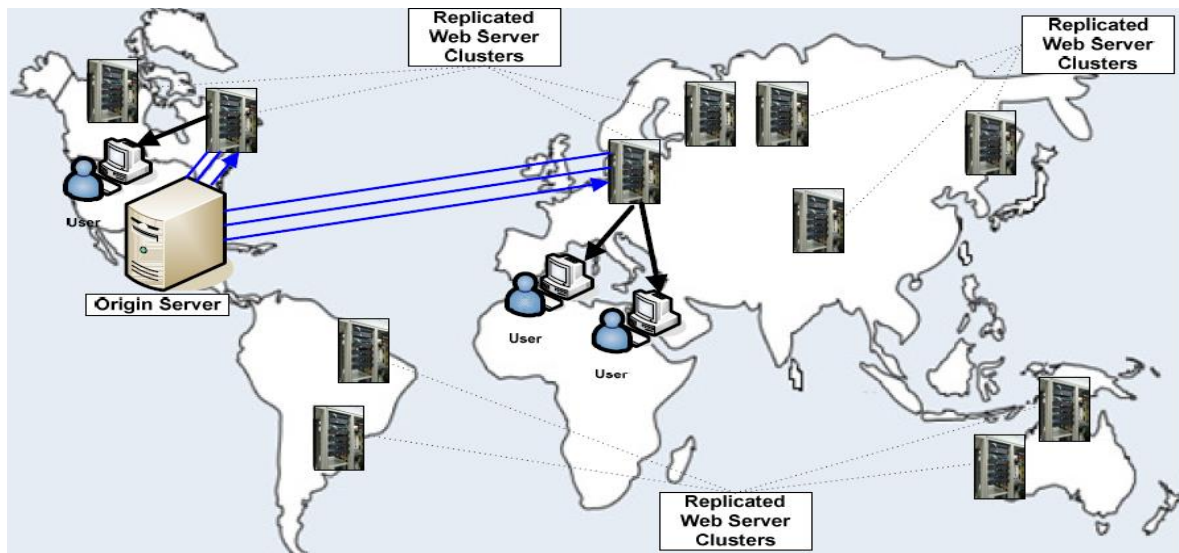


Figure 1: The Architecture of a Content Delivery network

CDN providers ensure the fast delivery of any digital content. They host third-party content including static content (e.g. static HTML pages, images, documents, software patches), streaming media (e.g. audio, real time video), User Generated Videos (UGV), and varying content services (e.g. directory service, e-commerce service, file transfer service). The sources of content include large enterprises, Web service providers, media companies and news broadcasters. The end-users can interact with the CDN by specifying the

content/service request through cell phone, smart phone/PDA, laptop and desktop. Figure 2 depicts the different content/services served by a CDN provider to end-users.

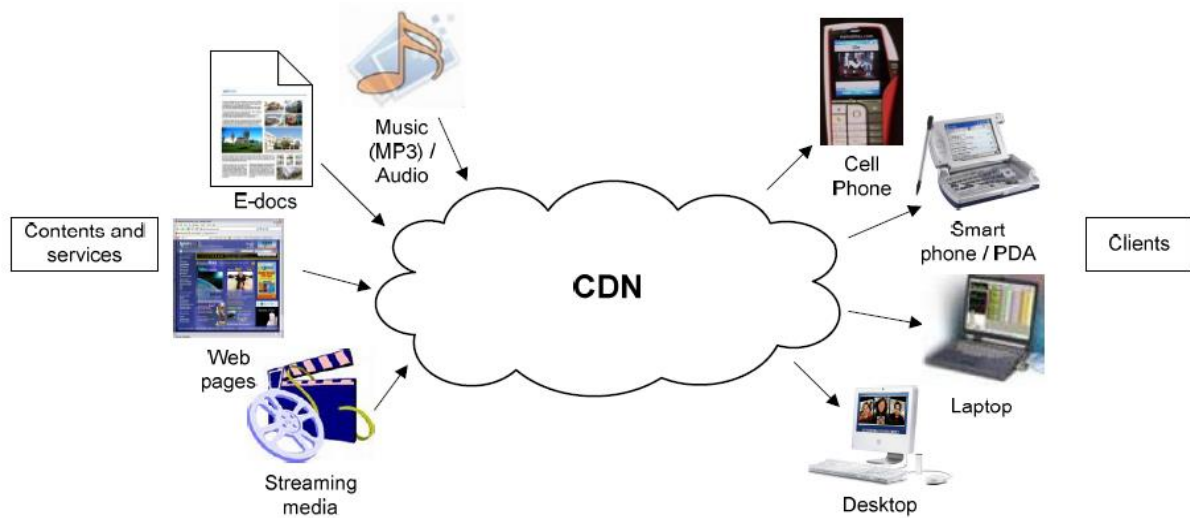


Figure 2: Content/services provided by a CDN

CDN providers charge their customers according to the content delivered (i.e. traffic) to the end-users by their surrogate servers. CDNs support an accounting mechanism that collects and tracks client usage information related to request-routing, distribution and delivery. This mechanism gathers information in real time and collects it for each CDN component. This information can be used in CDNs for accounting, billing and maintenance purposes. The average cost of charging of CDN services is quite high, often out of reach for many small to medium enterprises (SME) or not-for-profit organizations. The most influencing factors affecting the price of CDN services include:

- Bandwidth cost
- Variation of traffic distribution
- Size of content replicated over surrogate servers
- Number of surrogate servers
- Reliability and stability of the whole system and security issues of outsourcing content delivery

Origin server: This is a powerful storage system that contains all the content and/or the meta data of all the content. To achieve high performance of the whole CDN, the content in the origin server are pushed to the POP servers (surrogate servers) that are located at different geographical locations across the globe.

POP servers: This is distributed in a large numbers at diverse areas in a CDN. The main function of pop server is to offer the content based on user request. When the content is not available locally, the pop server should pull it from the origin server and store it for the next probable requirement; as it might be possible that the same/other user(s) in the region will require the content. Prefetching is another important functionality provided by the POP server

where it fetches the content that clients may be interested in from the origin server thereby reducing the chance of traffic congestion especially during the high demand. Needless to say, prefetching needs to predict the users preferential contents by synthesizing and analyzing the historical information such as access logs. It is evident that this kind of prefetching techniques may require statistical data mining algorithms to determine what content to prefetch.

Request Redirecting mechanism: One of the functions of a CDN is to dynamically redirect clients to the most optimal servers based on several QoS parameters such as server load, latency, network congestion, client access networks, and proximity etc. There are a variety of methods that can be used to implement this mechanism as presented in Table 1.

Global Server Load Balancing	Global awareness
	Smart authoritative DNS
DNS-based request routing	
HTTP redirection	
URL rewriting	URL modification
	Automation through scripts
Anycasting	IP anycast
	Application level anycast
CDN Peering	Centralized directory model
	Distributed Hash Table
	Flooded request model
	Document routing model

Table 1: CDN Request Redirecting Mechanisms

Global Server Load Balancing (GSLB): Aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any one of the resources. The capabilities that allow global server load balancing include global awareness and smart authoritative domain name service (DNS). In GSLB, services nodes are aware of information and status of other service nodes. This provides intermediate switching nodes to be globally aware. To make use of the global awareness, intermediate switches act as smart authoritative DNS, each switching between the best surrogate servers.

DNS-based request-routing today commercial content distribution services rely on modified Domain Name System servers to dynamically redirect clients to the appropriate content server. This is the simplest form of redirection, according to which a domain name, e.g. www.cdn.com has multiple IP records attached to it. When a client requests the IP address of the domain, any one of the IP records from the pool will be selected based on the DNS action.

The popularity of DNS based request-routing techniques is mainly due to the ubiquity of DNS as a directory service. They mainly consist in inserting a specialized DNS server in the name resolution process. This specialized server is capable of returning a different set of A,

NS or CNAME records based on user defined policies, metrics, or a combination of both. In the single reply approach, the DNS server returns the IP address of the best surrogate in an A record to the requesting DNS server (client site DNS server's). The IP address of the surrogate could also be a virtual IP address of the best set of surrogates for requesting DNS server. The best surrogate will be selected, in a second step, by a server switching mechanism.

In the multiple replies approach, the request-routing DNS server returns multiple replies such as several A records for various surrogates. Common implementations of client site DNS servers cycle through the multiple replies in a round robin fashion. The order in which the records are returned can be used to direct multiple clients using a single client site DNS server.

A multiple-level resolution approach is also possible according to which multiple request-routing DNS servers can be involved in a single DNS resolution, thus demanding complex decisions from a single server to multiple, more specialized, request-routing DNS servers, disseminated in different points of the Internet .

The most common mechanisms used to insert multiple request-routing DNS servers in a single DNS resolution is the use of NS and CNAME records: NS records allow the DNS server to redirect the authority of the next level domain to another request-routing DNS server; CNAME records allow the DNS server to redirect the resolution request to an entirely new domain.

There are three main drawbacks of using NS records. First the numbers of request-routing DNS servers are limited by the number of parts in the DNS name; second the last DNS server can determine the Time To Live (TTL) of the entire resolution process. The client will cache the returned NS record and use it for further request resolutions until it expires. As a third drawback, a delay is added in the resolution process due to the use of multiple DNS servers.

Request-routing based on CNAME record has the advantage to redirect the resolution process to another domain, and the number of request-routing DNS servers is independent of the format of the domain name. The main disadvantage is the introduction of an additional overhead in resolving the new domain name.

The basic limitations of DNS based request-routing techniques can be summarized as follows below:

- A. DNS allows resolution only at the domain level. However, an ideal request resolution system should serve requests at object granularity (preserving sessions if needed).
- B. A short TTL of DNS entry allows reacting quickly to network outages. This in return may increase the volume of requests to DNS servers. Therefore many DNS implementations do not honor the DNS TTL field.
- C. DNS request-routing does not take into account the IP address of the clients. Only the Internet location of the client DNS server is known: this limits the ability of the request-routing system to determine a client's proximity to the surrogate.

D. Users that share a single client site DNS server will be redirected to the same set of IP addresses during the TTL interval. This might lead to overloading the surrogate during a flash crowd.

HTTP Redirection: takes advantage of the HTTP protocol's redirection feature. This mechanism builds on special Web servers that can inspect a client request and chooses the most suitable surrogate server and redirect the client to those servers. This approach provides the flexibility of managing replication with finer granularity (e.g., at page level). However, it does pose significant overheads due to the introduction of extra messages round trips.

URL Rewriting: can be one of the best and quickest ways to improve the usability and search friendliness. A rewrite engine is software located in a Web application framework running on a Web server that modifies a web URL's appearance. Many framework users have come to refer to this feature as a "Router". This modification is called URL rewriting. For example, request for web sites with images, the router can rewrite the URLs of the images to point to the best surrogate servers.

Anycasting: This solution aims at solving the request-routing problem at the IP packet routing level. The base principle is that a group of servers providing the same service can be addressed using an anycast name and an anycast address. A user willing to access some service, e.g., a given content, from any of the (equivalent) servers issues a request with the anycast name. This is mapped to the anycast address and the request is sent into the network with the anycast address as destination. The role of the anycast service, is to redirect these request to one of the servers, thus selecting the server which will serve the user request. Since the redirection system has the obvious goal of improving clients performance, redirection is based upon some performance criteria, e.g. minimize the user perceived response time. The redirection is therefore performed by a cooperative access router that is capable of selecting the best suited replica from the anycast table. Anycasting is a more elaborate location mechanism that targets network-wide replication of the servers over potentially heterogeneous platforms. A mechanism for request redirection that is based on the anycasting concept must allow for the maintenance of information about the servers state and performance. An anycast service can be implemented at different levels in the network protocol stack. At the network layer, anycasting mechanisms consist in associating a common IP anycast address with the group of replicated servers. The routing protocol routes datagrams to the closest server, using the routing distance metric. Standard intra-domain unicast routing protocols can accomplish this, assuming each server advertises the common IP address. At the network level the implementation of an Anycast service entails the following mechanisms:

- ❖ Anycast request interception, which may be dealt at the network level by the edge routers. Edge routers are set to filter packets with Anycast destination address.
- ❖ Anycast name to Anycast address translation mechanism.
- ❖ Server Selection that maybe dealt by an edge router module (or by an application interacting with the router). This module interacts with the measurement module and keeps and updates Anycast server performance metrics.
- ❖ Anycast address to IP address translation that maybe dealt at the network level by the edge routers. Upon receiving an Anycast address, edge routers perform redirection by translating them into a selected unicast address.
- ❖ Measurements collection to be used for the selection process itself.

At the application level the resolution of the Anycast address is performed by means of a hierarchy of Anycast resolvers that map the Anycast domain name (AND) onto an IP address. To perform the mapping the resolvers maintain two types of information:

- 1) the list of IP addresses that form particular Anycast groups, and
- 2) a metric database information associated with each member of the Anycast group, while authoritative resolvers maintain the definitive list of IP addresses for a group, whereas local resolvers cache this information.

CDN Peering: is a methodology where clients provide resources; the client can also use these resources based on their requirements. This means that unlike client-server systems, the content serving capacity of peer-to-peer networks can actually increase as more users begin to access the content (especially with protocols such as Bit torrent that require users to share). This property is one of the major advantages of using P2P networks because it makes the setup and running costs very small for the original content distributor. To locate the content in CDN peering, a centralised directory model, distributed hash table, flooded request model or document routing model can be used. In centralised P2P file-sharing service, a large server is used to provide directory service. The P2P application contacts the directory service, informing the directory service of its IP address and the names of objects in its local disk that it is making available for sharing. When an active peer obtains a new object or removes one, it informs the directory server, which then updates its database. In a distributed hash table, peers are indexed through hashing keys and are found through complex queries within a distributed system. This approach is good in performing load balancing and offloading loads to less-loaded peers. The flooded request model is simple but scales poorly. When a node wants to find a resource on the network, which may be on a node it does not know about, it could simply broadcast its search query to its immediate neighbours. If the neighbours do not have the resource, it then asks its neighbours to forward the query. This is repeated until the resource is found or all the nodes have been contacted, or perhaps a network-imposed hop limit is reached.

Chapter 2

Benefits of Using Content Delivery Networks and Applied Systems

1. Three Common Distributed System

Data grids, distributed databases and peer-to-peer (P2P) networks are three distributed systems that have some characteristics in common with CDNs. These three systems have been described here in terms of requirements, functionalities and characteristics.

1.1. Data grids

A data grid is a data intensive computing environment that provides services to the users in different locations to discover, transfer, and manipulate large datasets stored in distributed repositories. At the minimum, a data grid provides two basic functionalities: a high-performance, reliable data transfer mechanism, and a scalable replica discovery and management mechanism. A data grid consists of computational and storage resources in different locations connected by high-speed networks. They are especially targeted to large scientific applications such as high energy physics experiments at the Large Hadron Collider , astronomy projects – Virtual Observatories, and protein simulation – BioGrid that require analyzing huge amount of data. The data generated from an instrument, experiment, or a network of sensors is stored at a principle storage site and is transferred to other storage sites around the world on request through the data replication mechanism. Users query the local replica catalog to locate the datasets that they require. With proper rights and permissions, the required dataset is fetched from the local repository if it is present there or otherwise it is

fetched from a remote repository. The data may be transmitted to a computational unit for processing. After processing, the results may be sent to a visualization facility, a shared repository, or to individual users' desktops. Data grids promote an environment for the users to analyze data, share the results with the collaborators, and maintain state information about the data seamlessly across organizational and regional boundaries. Resources in a data grid are heterogeneous and are spread over multiple administrative domains. Presence of large datasets, sharing of distributed data collections, having the same logical namespace, and restricted distribution of data can be considered as the unique set of characteristics for data grids. Data grids also contain some application specific characteristics. The overall goal of data grids is to bring together existing distributed resources to obtain performance gain through data distribution. Data grids are created by institutions who come together to share resources on some shared goal(s) by forming a Virtual Organization (VO). On the other hand, the main goal of CDNs is to perform caching of data to enable faster access by the end-users. Moreover, all the commercial CDNs are proprietary in nature – individual companies own and operate them.

1.2. Distributed databases

A distributed database (DDB) is a logically organized collection of data distributed across multiple physical locations. It may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers. Each computer in a distributed database system is a node. A node in a distributed database system acts as a client, server, or both depending on the situation. Each site has a degree of autonomy, is capable of executing a local query, and participates in the execution of a global query. A distributed database can be formed by splitting a single database or by federating multiple existing databases. The distribution of such a system is transparent to the users as they interact with the system as a single logical system. The transactions in a distributed database are transparent and each transaction must maintain integrity across multiple databases. Distributed databases have evolved to serve the need of large organizations that need to replace existing centralized database systems, interconnect existing databases, and to add new databases as new organizational units are added. Applications provided by DDB include distributed transaction processing, distributed query optimization, and efficient management of resources. DDBs are dedicated to integrate existing diverse databases to provide a uniform, consisting interface for query processing with increased reliability and throughput. Integration of databases in DDBs is performed by a single organization. Like DDBs, the entire network in CDNs is managed by a single authoritative entity. However, CDNs differ from DDBs in the fact that CDN cache servers do not have the autonomic property as in DDB sites. Moreover, the purpose of CDNs is content caching, while DDBs are used for query processing, optimization and management.

1.3. Peer-to-peer networks

Peer-to-peer (P2P) networks are designed for the direct sharing of computer resources rather than requiring any intermediate and/or central authority. They are characterized as information retrieval networks that are formed by ad-hoc aggregation of resources to form a fully or partially decentralized system. Within a peer-to-peer system, each peer is autonomous and relies on other peers for resources, information, and forwarding requests. Ideally there is no central point of control in a P2P network. Therefore, the participating

entities collaborate to perform tasks such as searching for other nodes, locating or caching content, routing requests, encrypting, retrieving, decrypting, and verifying content. Peer-to-peer systems are more fault-tolerant and scalable than the conventional centralized system, as they have no single point of failure. An entity in a P2P network can join or leave anytime. P2P networks are more suited to the individual content providers who are not able to access or afford the common CDN. An example of such system is BitTorrent, which is a popular P2P replication application. Content and file sharing P2P networks are mainly focused on creating efficient strategies to locate particular files within a group of peers, to provide reliable transfers of such files in case of high volatility, and to manage heavy traffic (i.e. flash crowds) caused by the demand for highly popular files. This is in contrast to CDNs where the main goal lies in respecting client's performance requirements rather than efficiently finding a nearby peer with the desired content. Moreover, CDNs differ from the P2P networks because the number of nodes joining and leaving the network per unit time is negligible in CDNs, whereas the rate is important in P2P networks.

2. Usage of CDN

While a CDN can be used for any type of website, it is primarily used in three application areas:

Media Distribution or Media delivery:

- ◆ Usually used for entertainment such as sporting events, television programs, movies as well as by news media.
- ◆ Live broadcasts over the Internet of one-time events such as a swearing-in ceremony of a head of state.
- ◆ Corporate Communications such as corporate videos, weekly meetings, earnings reports.

Media distribution of content over the Internet is fairly common today, with three possible types of media delivery. One is the live streaming of global or one-time events such as award ceremonies or a political leader's swearing-in ceremony. The second and the most common is Video-on-Demand such as YouTube videos in flv, mp4 and m4v formats. A third form is an Internet based 24/7 channel that plays pre-recorded content at scheduled times similar to a TV channel.

An organization first has to decide on the type of media delivery and the criteria relevant to that type of media delivery.

For live delivery, the number of nodes or POPs that can service the intended audience, redundancy and 24/7 customer support are deciding factors. Users can access live broadcasts, weekly meetings and 24/7 channel broadcasts through a variety of devices such as Pcs, laptops, tablets (iOS, Android) mobile devices (iOs, Android, Windows 7) and set-top boxes. The questions to ask would be: Does the carrier have a global reach and a high capacity network that is scalable and can it deliver the live broadcasts instantly without buffering or freezing? With respect to performance and scalability, can streaming be supported for up to hundreds or thousands of simultaneous viewers worldwide? What is

the delay between the live feed and the feed at the end-user? Is there support for DVR-like controls so that users can pause, rewind, restart and jump to any point in a live broadcast? Real-time reporting of user statistics is valuable for live broadcasts.

With on-demand video delivery, videos should start instantly and play uninterrupted. Network reach and the type of architecture available for storage and delivery of videos may be important considerations. Content owners could push files to the CDN provider's origin servers and leverage the provider's distributed architecture, or have the files pulled from their origin server or cloud storage account. Analytics such as most watched videos, geographic location of users and viewing time are valuable statistics to obtain from the CDN provider.

Website Acceleration/Caching:

Usually used for websites whose users are spread over a wide geographic area such as retail, online reservations and ticketing and online newspapers?

Website acceleration speeds up the delivery of content to visitors accessing the website and reduces the need to scale infrastructure when traffic increases or becomes more geographically dispersed, thereby improving user experience. For businesses that run a reservations system, on-line ticketing or have an online market place, website acceleration improves speed of delivery by improving the HTTP or HTTPS performance.

Most CDN providers offer one of four complimentary technologies for website acceleration or caching. These are static site caching, dynamic site caching, web application acceleration or IP acceleration.

Static site caching uses a mechanism called GEO-DNS2 to cache the content in geographically diverse areas. The technology helps CDN point its distributed network at an origin server and cache its content in the correct geographical region. This is also known as 'reverse proxy caching'.

Dynamic site caching is applicable to dynamic data such as pricing or account balances. These change over short periods of time and thus cannot be cached for long durations.

Web application acceleration methods typically focus on improving HTTP and HTTPS performance. The techniques used could be HTTP protocol acceleration and compression or tuning of packet sizes. Browser defaults are most commonly targeted to improve performance.

IP Acceleration focuses on improving Transport Control Protocol (TCP) traffic unlike web application acceleration methods. However since TCP is more suited for accuracy than speed, some CDN providers prefer to work on solutions for accelerated IP.

Large File/Software Delivery

Used by gaming companies or software providers to distribute software. Enterprise application vendors, anti-virus software providers and gaming companies usually distribute large files over the Internet and want this process to go smoothly. Securing the content via

delivery over the SSL (Secure Socket layer), authenticating user access, large file optimization and download analytics are important factors.

3. Application of CDN

The benefits of CDNs are more emphasized by examining its usage in a few real time application areas. Some common application areas include,

- ❖ **E-Commerce:** E-commerce companies make use of CDNs to improve their site performance and making their products available online. According to Computer World, CDN provides 100% uptime of e-commerce sites and this leads to improved global website performance. With continuous uptime companies are able to retain existing customers, leverage new customers with their products and explore new markets, to maximize their business outcomes.
- ❖ **Media and Advertising:** In media, CDNs enhance the performance of streaming content to a large degree by delivering latest content to end users quickly. We can easily see today, there is a growing demand for online video, and real time audio/video and other media streaming applications. This demand is leveraged by media, advertising and digital content service providers by delivering high quality content efficiently for users. CDNs accelerate streaming media content such as breaking news, movies, music, online games and multimedia games in different formats. The content is made available from the data center which is nearest to users' location.
- ❖ **Business Websites:** CDNs accelerate the interaction between users and websites, this acceleration is highly essential for corporate businesses. In websites speed is one important metric and a ranking factor. If a user is far away from a website the web pages will load slowly. Content delivery networks overcome this problem by sending requested content to the user from the nearest server in CDN to give the best possible load times, thus speeding the delivery process.
- ❖ **Education:** In the area of online education CDNs offer many advantages. Many educational institutes offer online courses that require streaming video/audio lectures, presentations, images and distribution systems. In online courses students from around the world can participate in the same course. CDN ensures that when a student logs into a course, the content is served from the nearest datacenter to the student's location. CDNs support educational institutes by steering content to regions where most of the students reside.

Chapter 3

Exploring the Architecture of Content Delivery Network

1. Layered architecture of CDN

The architecture of content delivery networks can be presented according to a layered approach. In Figure 3, we present the layered architecture of CDNs, which consists of the following layers: Basic Fabric, Communication & Connectivity, CDN and End-user. The layers are defined in the following as a bottom up approach.

- ❖ **Basic Fabric** is the lowest layer of a CDN. It provides the infrastructural resources for its formation. This layer consists of the distributed computational resources such as SMP, clusters; file servers, index servers, and basic network infrastructure connected by high-bandwidth network. Each of these resources runs system software such as operating system, distributed file management system, and content indexing and management systems.
- ❖ **Communication & Connectivity** layer provides the core internet protocols (e.g. TCP/UDP, FTP) as well as CDN specific internet protocols (e.g. Internet Cache Protocol (ICP), Hypertext Caching Protocol (HTCP), and Cache Array Routing Protocols (CARP), and authentication protocols such as PKI (Public Key Infrastructures), or SSL (Secure Sockets Layer) for communication, caching and delivery of content and/or services in an authenticated manner. Application specific overlay structures provide efficient search and retrieval capabilities for replicated content by maintaining distributed indexes.
- ❖ **CDN layer** consists of the core functionalities of CDN. It can be divided into three sub-layers: CDN services, CDN types and content types. A CDN provides core services such as surrogate selection, request routing, caching and geographic load balancing, and user specific services for SLA management, resource sharing and CDN brokering. A CDN can operate within an enterprise domain, it can be for academic and/or public purpose or it can simply be used as edge servers of content and services. A CDN can also be

dedicated to file sharing based on a peer-to-peer (P2P) architecture. A CDN provides all types of MIME content (e.g. text, audio, video etc) to its users.

- ❖ **End-users** are at the top of the CDN layered architecture. In this layer, we have the Web users who connect to the CDN by specifying the URL of content provider’s Web site, in their Web browsers.

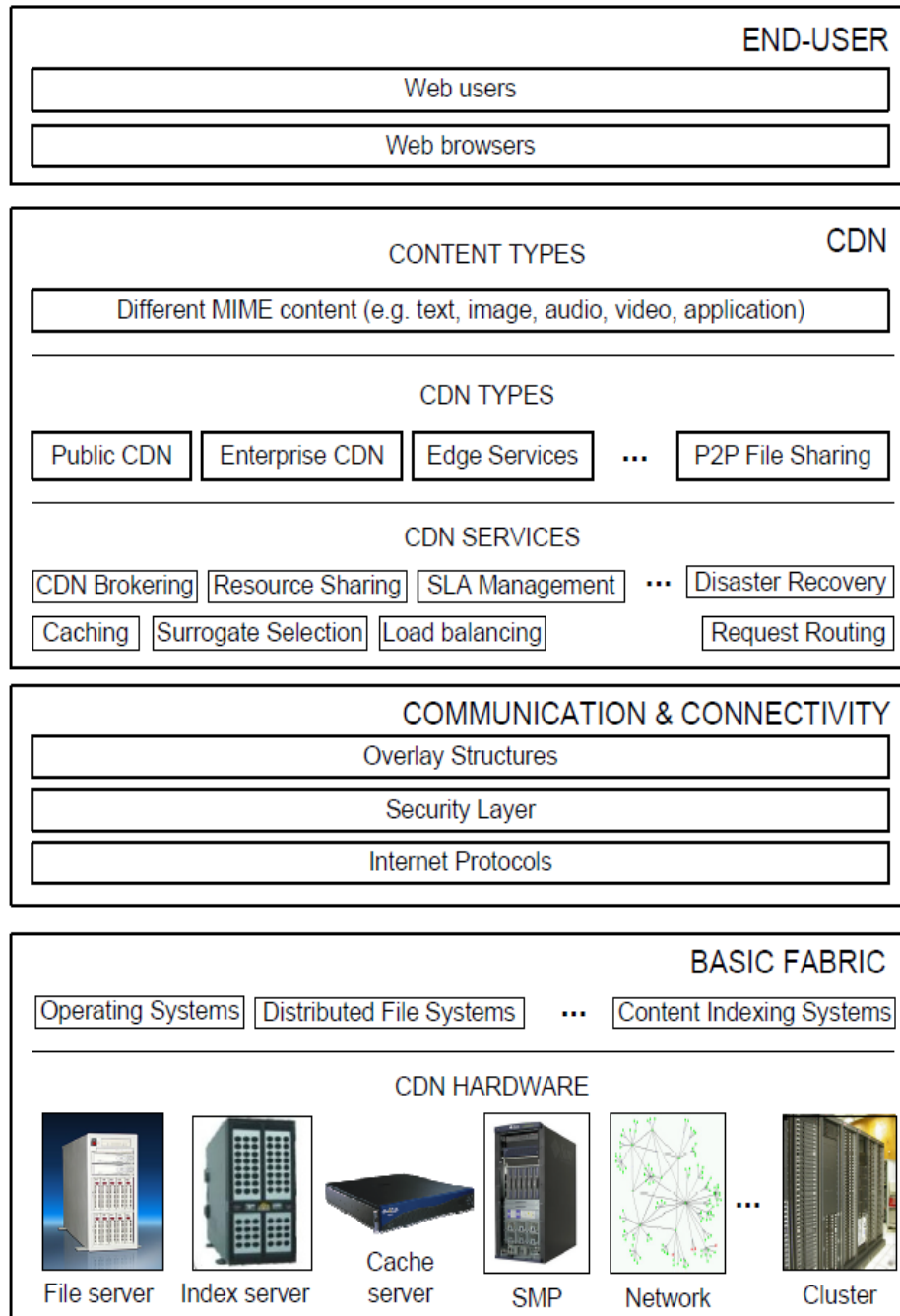


Figure 3: Layered architecture of a CDN

2. Physical Architecture Content Delivery Networks

The main goal of server replication in a CDN is to avoid large amounts of data repeatedly traversing possibly congested links on the Internet. As Figure 4 shows, there are a variety of ways and scale (local area or wide area networks) in which content networks may be implemented. Local solutions are web clusters, that typically host single site, and web farms, typically used to host multiple sites. Wide area solutions include: distributed web server systems, used to host single or multiple sites; cooperative proxy cache networks (a service infrastructure to reduce latency in downloading web objects) and content delivery networks that are the focus of this paper. A typical server farm is a group of servers, ranging from two to thousands, that makes use of a so-called cooperative dispatcher, working at OSI layers 4 and/or 7, to hide the distributed nature of the system, thus appearing as a single origin site. A layer 4 web switch dispatches the requests, among a group of servers, on the basis of network layer information such as IP address and TCP port. A content switch, working at the application layer, examines the content of requests and dispatches them among a group of servers. The goals of a server cluster/farm include: load-balancing of requests across all servers in the group; automatic routing of requests away from servers that fail; routing all requests.

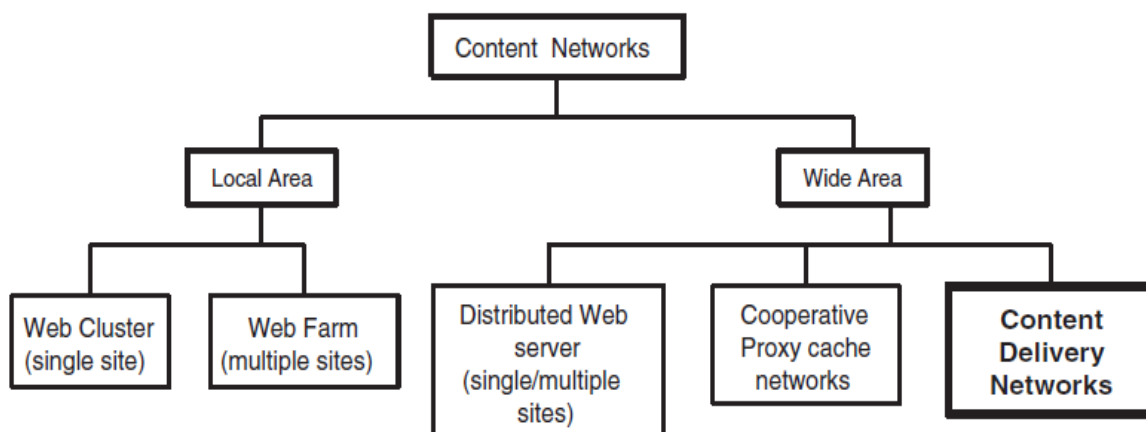


Figure 4: Taxonomy of Content Networks

A type of content network that has been in use for several years is a caching proxy deployment. Such a network might typically be employed by an ISP for the benefit of narrow bandwidth users accessing the Internet. In order to improve performance and reduce bandwidth utilization, caching proxies are deployed close to the users. These users are encouraged to send their web requests through the caches rather than directly to origin servers, by configuring their browsers to do so. When this configuration is properly done, the user's entire browsing session goes through a specific caching proxy. This way the proxy cache would contain the hot portion of content that is being viewed by all the users of that caching proxy. A provider that deploys caches in many geographically locations may also deploy regional parent caches to further aggregate user requests thus creating an architecture known as hierarchical caching. This may provide additional performance improvements and bandwidth savings. Using rich parenting protocols, redundant parents may be deployed such that a failure in a primary parent is detected and a backup is used instead. Using similar

parenting protocols, requests may be partitioned such that requests for certain content domains are sent to a specific primary parent. This can help to maximize the efficient use of caching proxy resources. Clients may also be able to communicate directly with multiple caching proxies.

Though certainly showing better scalability than a single origin server, both hierarchical caching and server farms have their limits. In these architectures, the replica servers are typically deployed in proximity to the origin server, therefore they do not introduce a significant improvement to the performance difficulties that are due to the network congestion. Caching proxies can improve performance difficulties due to congestion (since they are located in proximity to the final users) but they cache objects reactively to the client demand. Reactive caching based on client demand performs poorly if the requests for a given object, while numerous in aggregate, are spread among many different caching proxies.

To address these limitations, CDNs employ a solution based on proactive rather than on reactive caching, where the content is prefetched from the origin server and not cached on demand. In a CDN, multiple replicas host the same content. A request from a browser for a single content item is directed to the replica that is considered the best suited at the moment of the request arrival, and the item is served to the client in a shorter time than the one it would have taken to fetch it from its origin server. Since static information about geographic locations and network connectivity are not sufficient to choose the best replica, a CDN typically incorporates dynamic information about network conditions and load on the replicas, to redirect requests and balance the load among the servers. Operating a CDN is therefore a complex and expensive activity. For this reason a CDN is typically built and operated by a network/service provider that offers a content distribution service to several content providers.

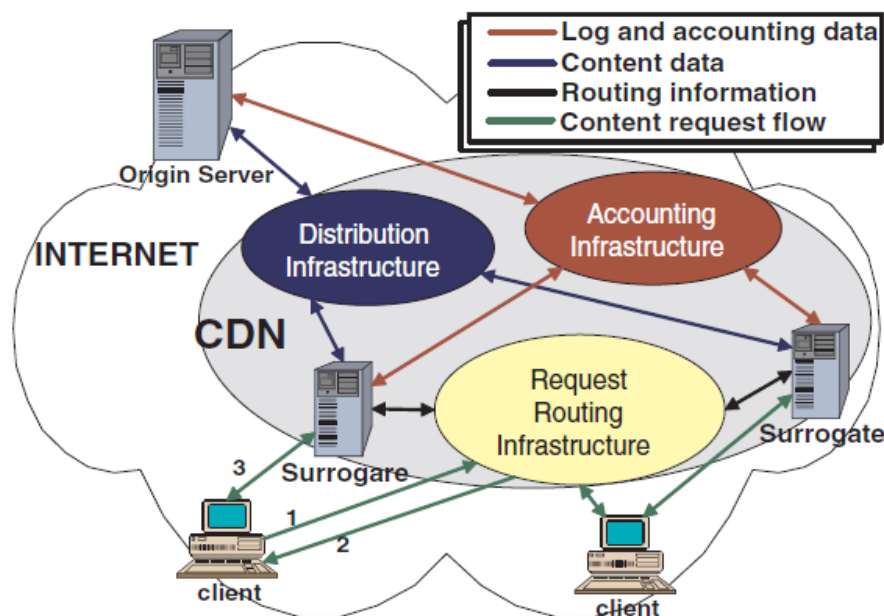


Figure 5: Infrastructure components of a Content Delivery Network

A content delivery architecture consists of a set of surrogate servers that deliver copies of content to the users while combining different activities (see figure 5).

- The request-routing infrastructure consists of mechanisms to redirect content requests from a client to a suitable surrogate.
- The distribution infrastructure consists of mechanisms to move contents from the origin server to the surrogates.
- The accounting infrastructure tracks and collects data on request-routing, distribution, and delivery functions within the CDN creating logs and reports of distribution and delivery activities.

The origin server (hosting the content to be delivered) interacts with the CDN two ways (see figure 5):

- it pushes new content to the replica servers, (the replica themselves request content updates from the origin server through the distribution infrastructure);
- it requests logs and other accounting data from the CDN or the CDN itself provides this data to the origin server through the accounting infrastructure.

The clients interact with the CDN through the request routing infrastructure and surrogate servers. Figure 5 shows one of the possible scenarios of interaction between the clients, the access routers, the replica servers and the origin server.

The user agent sends

- (1) a content request to the routing infrastructure, that redirects
- (2) the client request to a surrogate server, to which the client subsequently asks
- (3) The desired content.

Chapter 4

Management and Placement of Replicas in the context of CDN

1. Content Caching Techniques

The proactive caching infrastructure must be transparent to the end-users that must see no difference with being served directly by the central server. Proactive caching to the edges offers better delivery to the client because the content is located in their proximity. Therefore the requesting users perceive a lower latency, higher availability and lower load on the network links. Such architecture is also inherently protected from sudden burst that can be distributed among many servers so that no single device has to cope with a massive load. This close to the client deployment mode is commonly known as forward proxy caching. Forward proxy implementations can reduce wide area network traffic by 30 to 50 percent (results vary based on the "cacheability" of the requested content). A Web cache monitors Internet traffic, intercepts requests for Web objects and then fulfills those requests from the set of objects it stores (cache hit). If the requested object is not in the cache (cache miss), the cache forwards the request to the origin server, which sends a copy of the object back to the cache. The cache stores the object and sends it back to the requester. Caches in CDN cooperate interacting through the Internet Cache Protocol (ICP). ICP is typically used to build cache clusters or child-parent relationships in hierarchical caching. A cache can react to a cache miss inquiring other cooperative caches, in spite of the origin server, in order to retrieve the content from a closer location. Caching also acts as a point of control and security. Today's caches frequently include support for content filtering, anti-virus, access control and bandwidth management. Anti-virus and content filtering give users an extra level of security across the network. The access control and bandwidth management further assists in the reduction of the overall network utilization by making sure that only approved users get access to bandwidth, and that the bandwidth is being allocated in a way that ensures the best adherence to the signed agreements on quality. Caching activity in a CDN may involve different types of contents and therefore different functionalities.

- ❖ **Static Caching:** to cache and replicate static content, such as html pages, images, documents, audio/video file etc.
- ❖ **Dynamic Caching:** to cache and replicate dynamically generated content. This include application delivery and replication.
- ❖ **Streaming Media Caching:** to store streaming media objects, as well as to serve streaming media to clients. Essentially, the cache acts as a streaming media server, storing media clips for later use.
- ❖ **Live Splitting:** to cache replicated live streams, so that only one copy is pulled down from the upstream server and is then distributed to the subscribing clients.

2. Replica Placement

A hot topic in content delivery design is the replica placement problem: where and how the replica could be distributed across the Internet to minimize the user latency, the number of replica, and the bandwidth used for replica management? The majority of the schemes presented in the literature tackle the problem of static replica placement that can be formulated as follows. Given a network topology, a set of CDN servers and a given request traffic pattern, decide where content has to be replicated so that some objective function is optimized while meeting constraints on the system resources. The solutions so far proposed typically try to either maximize the user perceived quality given an existing infrastructure, or to minimize the CDN infrastructure cost while meeting a specified user perceived performance. Examples of constraints taken into account are limits on the servers storage, on the servers sustainable load, on the maximum delay tolerable by the users etc.

3. Cache Consistency and Content Flow

One of the important problems in CDNs is how to manage the consistency of content at replicas with that at the origin server, especially for those documents changing dynamically. Cached objects typically have associated expiration times after which they are considered stale and must be validated with a remote server (origin or another cache) before they can be sent to a client. Sometimes, a considerable fraction of cache hits involve stale copies that turned out to be current.

These validations of current objects have small message size, but nonetheless, they often induce latency comparable to cache misses. Thus the functionality of caches as latency-reducing mechanism highly depends not only on content availability but also on its freshness. A technique to achieve cache consistency consists in pre-populating, or pushing, content to the cache before requests arrive. When automatically pushing a new, or updated, Web object to a cache, the content in the cache is guaranteed to be always fresh and there is no reason for the cache to initiate a freshness check with the side effect that this technique often generates a large amount of traffic.

Chapter 5

Content Delivery Network and Cloud Computing

CDNs have made a significant impact on how content is delivered via the Internet to the end-users . Traditionally content providers have relied on third-party CDNs to deliver their content to end users. With the ever changing landscape of content types e.g. moving for standard definition video to high definition to full high definition, it is a challenge for content providers who either supplement their existing delivery networks with third-party providers or completely rely on them to understand and monitor the performance of their service. Moreover, the performance of the CDN is impacted by the geographical availability of the third-party infrastructure. A cloud CDN (CCDN) provides a flexible solution allowing content providers to intelligently match and place content on one or more cloud storage servers based on coverage, budget and QoS preferences . The key implication is economies of scale and the benefits delivered by the pay-as-you-go model. Using clouds the content providers have more agility in managing situations such as flash crowds avoiding the need to invest in infrastructure development.

As stated previously, clouds provide the end users with virtually infinite pool of compute and

storage resources with no capital investment in terms of hardware and software. Therefore, CCDN systems can be very valuable in data processing and delivery of content over the Internet. The main advantage of such a system would be that they provide a cheaper means of hosting and deploying multi-tiered applications that can scale based on the usage demands. Further clouds offer not only cheaper content storage and distribution functionality, but also compute functionality such that application and data processing can also be performed on clouds. Lastly, cloud offers pay-as-you-model whereby the end-users can start and terminate the cloud resources based on the amount of money they are willing to spend hosting their services without entering into a complex contract with the cloud provider. Migration from traditional client/server based CDNs to cloud computing model is a major transformation that introduces great opportunities and challenges. The major advantages and opportunities introduced by CCDNs include:

- A. **Pay-as-you-go CCDN model:** CCDN allows the users to consume the delivery content using a pay-as-you-go model. Hence, it would be much more cost-effective than owning the physical infrastructure that is necessary for the users to be the part of CDN.
- B. **Increased point-of-presence:** The content is moved closer to users with relative ease in the CCDN system than the traditional CDN due to the omnipresence of cloud. The Cloud-based content delivery network can reduce the transmission latency as it can rent operating resources from the cloud provider to increase the reach and visibility of the CDN on-demand.
- C. **CCDN Interoperability:** CDN interoperability has emerged as a strategic important concept for service providers and content providers. Interoperability of CDNs via the cloud will allow content providers to reach new markets and regions and support nomadic users. E.g., instead of setting up an infrastructure to serve a small group of customers in Africa, taking advantage of current cloud providers in the region to dynamically host surrogate servers.
- D. **Support for variety of CCDN application:** The cloud can support dynamic changes in load. This will facilitate the CDNs to support different kinds of applications that have unpredictable bursting traffic, predictable bursting traffic, scale up and scale down of resources and ability to expand and grow fast.

However, while cloud-based CDNs have made a remarkable progress in the past five years, they are still limited in a number of aspects. For instance, moving into the cloud might carry some marked security and performance challenges that can impact the efficiency and productivity of the CDN thus affecting the client's business. Further, current CCDNs are more suited to distributing static content such as audio, video and text. They are not well suited to serving dynamic content-based applications such as collaborative audio-video processing and streaming. Moreover, CDNs are usually owned by private and telecommunication companies making these services costly to the end-users as they have to enter in legal contract to use CDN services. A categorical list of technical issues and challenges in CCDN system is presented in Fig. 6 and the following sections.

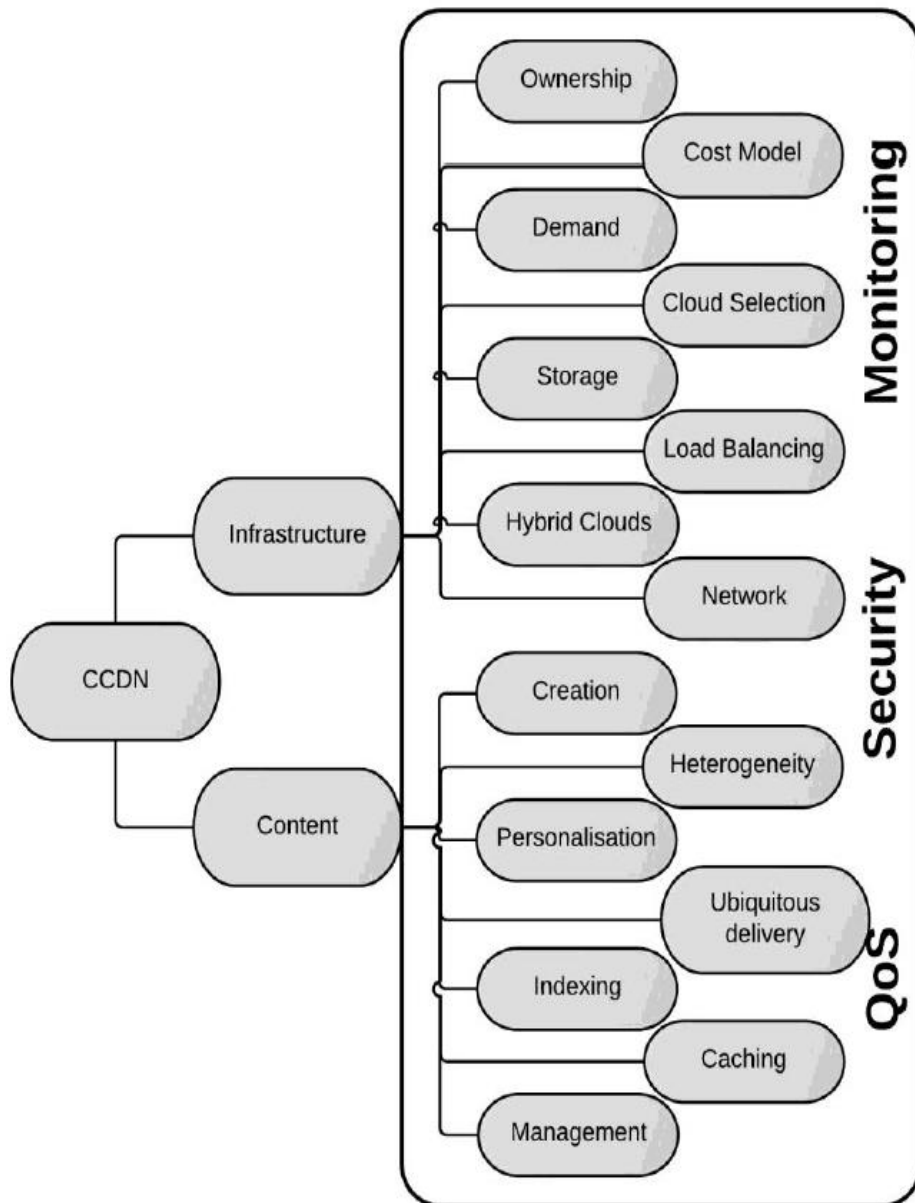


Figure 6: Classification of CCDN Challenges and Issues

1. Dynamic Content Management

CDNs are designed for streaming staged content but do not perform well in situations where content is produced dynamically. This is typically the case when content is produced, managed and consumed in collaborative activities. For example, an art teacher may find and discuss movies from different film archives; the students may then edit the selected movies. Parts of them may be used in producing new movies that will be sent to the students, friends for comments and suggestions. Current CDNs do not support such collaborative activities that involve dynamic content creation.

2. Content Creation

Traditional CDNs are not designed to manage content (e.g., find and play high definition movies). This is typically done by CDN applications . For example, CDNs do not provide services that allow an individual to create a streaming music video service combining music videos from an existing content source on the Internet (e.g., YouTube), his/her personal collection, and from live performances he/she attends using his/her smart phone to capture such content. This can only be done by an application managing where and when the CDN will deliver the video component of his/her music program. With CCDN, the end-user will act as both content creator and consumer. CCDN needs to support this feature inherently. User-generated content distribution is emerging as one of the dominant forms in the global media market.

3. Content Heterogeneity

Existing Web 2.0 technologies currently support the authoring of structured multimedia content (e.g., web pages linking images, sounds, videos, and animations). The CCDNs will need to extend and broaden existing Web 2.0 strengths with a new environment aimed at supporting the creation and consumption of interactive multimedia content (e.g., interactive audio and video), as well as other novel forms of multimedia content (e.g., virtual and augmented reality) that are currently not supported by existing Web 2.0 technologies and tools.

4. CCDN Ownership

Cloud CDN service providers either own all the services they use to run their CDN services or they outsource this to a single cloud provider. A specialized legal and technical relationship is required to make the CDN work in the latter case.

5. CCDN Personalization

CDNs do not support content personalization. For example, if the subscriber's behaviour and usage pattern can be observed, a better estimation on the traffic demand can be achieved. The performance of content delivery is moving from speed and latency to on-demand delivery of relevant content matching end-users interest and context.

6. Cost models for Cloud CDNs

The cloud cost model works well as long as the network consumption is predictable for both service provider and end-user. However, such predictions become very challenging with distributed cloud CDNs.

7. Security

CDNs also impose security challenges due to the introduction public clouds to store, share and route content. The use of multi-vendor public clouds further complicates this problem. Security is the protection of content against unauthorised usage, modification, tampering and protection against illegal use, hack attacks, viruses and other unwanted intrusions. Further,

security also plays an important role while accessing and delivering content to relevant users .

8. Hybrid Clouds

The integration of cloud and CDN will also allow the development of hybrid CCDN that can leverage on a combination and private and public cloud providers. E.g. the content provider can use a combination of cloud service platforms offered by Microsoft Azure and Amazon AWS to host their content. Depending on the pay-as-you go model, the content provider can also move from one cloud provider to another. However, achieving a hybrid model is very challenging due to various CCDN ownership issues and QoS issues.

9. CCDN Monitoring

The CCDNs can deliver end-to-end QoS monitoring by tracking the overall service availability and pinpoint issues. Clouds can also provide additional tools for monitoring specific content e.g. video quality monitoring. However, developing a CCDN monitoring framework is always a challenge.

10. CCDN QoS

With the notion of virtually unlimited resources offered by the cloud, quality for service plays a key role in CCDNs to maintain a balance between service delivery quality and cost. Defining appropriate SLA's to enforce QoS and guarantee service quality is very important and is also challenging. Further, the notion of hybrid clouds further complicate CCDN QoS challenges due to the involvement of multiple cloud providers with varying SLAs. CCDNs must accommodate highly transient, unpredictable users behaviour (arrival patterns, service time distributions, I/O system behaviours, user profile, network usage, etc.) and activities (streaming, searching, editing, and downloading).

11. CCDN Demand Prediction

It is critical that CCDNs are able to predict the demands and behaviours of hosted applications, so that it can manage the cloud resources optimally. Concrete prediction or forecasting models must be built before the demands and behaviours of CDN applications can be predicted accurately. The hard challenge is to accurately identify and continuously learn the most important behaviours and accurately compute statistical prediction functions based on the observed demands and behaviours such as request arrival pattern, service time distributions, I/O system behaviours, user profile, and network usage.

12. CCDN Cloud Selection

The diversity of offering by Cloud providers make cloud section to host CDN components a complex task. A practical question to be addressed is: how well does a cloud provider perform compared to the other providers? For example, how does a CDN application engineer compare the cost/performance features of CPU, storage, and network resources offered by Amazon EC2, Microsoft Azure, GoGrid, FelxiScale, TerreMark, and RackSpace. For instance, a low-end CPU resource of Microsoft Azure is 30% more expensive than the comparable Amazon EC2 CPU resource, but it can process CDN application workload twice

as quickly. Similarly, a CDN application engineer may choose one provider for storage intensive applications and another for computation intensive CDN applications. Hence, there is need to develop novel decision making framework that can analyse existing cloud providers to help CDN service engineers in making optimal selection decisions.

13. Ubiquitous content delivery

Content delivery services will interact with the network and appropriately adjust its QoS as needed to deliver content to a specific user based on content and user requirements for maintaining its integrity, the device the user is using, his/her location, and the service contracts. This is a requirement for CCDNs with the growing complexity in media types, end-user access devices and intermediate network architectures.

14. Flexible content storage, compression, and indexing

Cloud storage resources allow content producers to store content on virtualized disks and access them anytime from any point on the Internet. These storage resources are different from the local storage (for example, the local hard drive) in each CPU resource (e.g., Amazon EC2 instance types), which is temporary or non-persistent and cannot be directly accessed by other instances of CPU resources. Multiple storage resource types are available for building content orchestrator. Naturally, the choice of a particular storage resource type stems from the format (e.g., structured vs. unstructured) of the content. For instance, Azure Blob (<https://azure.microsoft.com/en-us/>) and Amazon S3 (<http://aws.amazon.com/>) storage resources can hold video, audio, photos, archived email messages, or anything else, and allow applications to store and access content in a very flexible way. In contrast, NoSQL (Not Only SQL) storage resources have recently emerged to complement traditional database systems. Amazon SimpleDB, Microsoft Azure Table Storage, Google App Engine Datastore, MongoDB, and Cassandra are some of the popular offerings in this category.

Though cloud environments are decentralized by nature, existing CDN application architecture tends to be designed based on centralized network models. It is worth noting that none of the existing cloud storage resources exposes content indexing APIs. It is up to the CDN application designer to come-up with efficient indexing structure that can scale to large content sizes to help end-users find and retrieve relevant content effectively and efficiently. To facilitate new and better ways of content delivery using CCDNs, advanced distributed algorithms need to be developed for indexing, browsing, filtering, searching and updating the vast amount of information.

15. Other Challenges

Apart from the above CCDN specific challenges, there are also several important factors specific to the CDN in a CCDN that affect the performance of service within the cloud infrastructure. These include:

- Network proximity: reduces the response time for improving the customers' experience about the services offered via the CDN.
- Load balancing: improves the capability of the whole network by decreasing the flash crowd situation i.e., it distributes the load to different nodes in a network such that response times and system throughput improve.
- Local caching: fetches the content for the customer from the origin server and

stores it in a local server closer to the customer. This technique helps in significantly reducing the response time.

— Request redirecting: plays a very pivotal role in the performance of a CDN service as it redirects the customer's request to the nearest cache server.

Chapter 6

Discussing Cloud Content Delivery Network (CCDN) in Academic and Commercial Settings

Technically, architectures of CCDNs in existence are various in terms of the correlation between CDN and Cloud. For instance, some CCDNs adopt the cloud-based store as their origin server. In this kind of CCDNs, the general mechanism is similar as the traditional CDNs". Other architecture includes Master/Slave mechanism . Specifically, in this kind of CCDNs the functionality of master node is managing, monitoring and provisioning slave nodes on demand. The slave nodes combine the functions of POP servers. The data is replicated in the master nodes which act as the origin server. When the slave node has to get some contents users require, it only needs to communicate with the master node to fetch the content. A typical CCDN architecture is presented in Fig 3. As depicted in the Fig. 7, the POPs are distributed across multiple cloud providers while the master node/origin server is responsible to orchestrate the entire CDN functionality. Based on demand from various geographical locations and QoS constraints, the master node will fire new slave POP nodes in close proximity to origin of user requests for a particular user agent's session to the same server, if necessary to preserve session state.

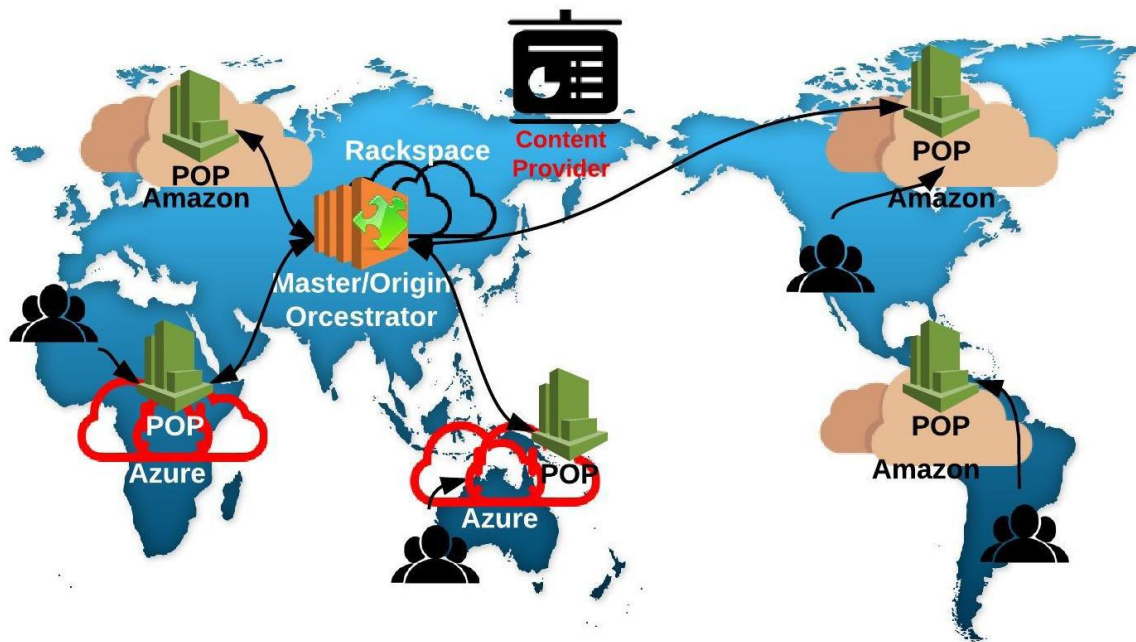


Figure 7: Cloud CDN Typical Architecture

Cloud-based CDNs offer a large number of additional services compared to traditional CDNs. These include:

1. Cloud Security:

The Cloud-based CDN providers can combine CDN security with the performance of cloud-based distributed infrastructures to keep their customers' websites both high performance and secure.

The following aspects are some instances of services the providers may provide in this area:

- a) **Data Security:** The Cloud-based CDN providers can apply and support advanced standards and methods of security like PCI compliance, secure socket layer, and digital rights management to offer the protection of their customers' data.
- b) **High Availability:** The providers offer the Cloud-based delivery of robust website and application functionality in a high-performance manner.
- c) **Cloud DDoS Protection:** It means the protection of websites via DDoS mitigation. The CDN can support proactive monitoring and alerting.
- d) **Regulatory Compliance:** The Cloud-based CDN enhance the CDN infrastructures and services to meet the requirement of industrial and governmental standards for protecting the customers' personal or financial information.

2. Cloud-Based DNS

The Domain Name System (DNS) is a very important Internet infrastructure that enables visitors to reach their website. DNS redirection has a very crucial role to play in a Content Delivery Network, for it enables the users to get the content they want from the available nearest surrogate server to reduce the response time. Fetching the users' preferential content from the optimal cache server can not only improve the performance but reduce the chance of traffic congestion especially during the rush time. Actually, DNS redirection is one of the mainly two techniques the most CDN providers adopt in their architecture to achieve the aim of redirecting the clients to the nearest surrogate server, the other is URL re-writing.

3. Cloud Storage

Taking advantage of Cloud Storage is a significant difference between the Cloud-based CDN and the traditional CDN. For users, to Store, maintain, and deliver a great mess of media, software, documents, or other digital objects is an essential part of ensuring an outstanding online experience. By using the Cloud storage functionality, the clients can effectively store a great amount of data and serve these data to the users who need such data in different locations over the world reliably and fastly. Furthermore, it is a very economical option. Though most Cloud-based CDN providers in today's world allege their Cloud storage designed for reliability, scale or speed, anyway they always mention the advantages of Cloud storage as more as possible, there are also some limitations of the Cloud storage:

- ❖ Due to some defective causes of the machines in the Cloud, the user's data which are stored in a Cloud Storage system can be corrupted and this would lead to the situation that the Cloud Storage system returns incorrect results to the users.
- ❖ The attacker may make a bug in the users programs to steal valuable information, even control the users' client to do Dos attacks or to spam.
- ❖ In some rush time, because of the traffic jam, users might not be able to get access to their data which are stored in the Cloud storage system accidentally.

The above disadvantages of Cloud storage may be extremely impossible, if the Cloud Storage system is robust, well-management, well-designed etc. However, the first importance of the fact is that when these bad events happen, it will be very difficult to find that who should be responded for that among the Cloud provider, the CDN provider and the Customer when something goes wrong. As a consequence, it would be necessary to build an accountable Cloud system which means it is easy to find who's false when some mistakes happen in such kind of Cloud system. Implementing data mining algorithms to analyse the log system is a good choice to address this problem.

4. Cloud Load Balancer

The Cloud load balancer provides the customers the flexibility to manage their content delivery strategy. This service enables customers to specify content delivery policies based on real-time conditions and user targets. The typical cloud load balancing technology manages the customers' application traffic and makes decisions of where to route it. When a node in the Cloud system fails, a health check process will remove it from rotation to keep maximum availability of the whole system. The Cloud load balancer service should follow the pay-as-you-go model as well in term of the hours the customers use, number of current connections

and bandwidth.

5. Cloud Orchestrator.

Cloud orchestration service offers enhanced flexibility and elasticity of CCDN as it manages public and private cloud resources using the pay-as-you-go model. Cloud orchestration operations include: (i) production: create and edit; (ii) storage: uploading and scaling of storage space; (iii) keyword-based content tagging and searching and (iv) distribution: streaming and downloading. At Cloud service level, the orchestrator capabilities span across a range of operations such as selection, assembly, deployment of cloud resources to monitoring their run-time QoS statistics (e.g., latency, utilization, and throughput). The orchestrate supports deployment, configuration and monitoring of content and cloud resources deployed across hybrid cloud platforms using web-based widgets. These widgets hide the underlying complexity related to cloud resources and provide an easy do-it-yourself interface for content management. The cloud orchestration service is also responsible to manage the cloud resources based on service providers SLAs.

6. Existing Commercial and Academic Cloud-based CDNs

The current landscape of CCDNs leverages the flexibility of the cloud to easily and quickly distribute content across the internet. The CCDNs landscape diversifies into two primary forms namely web site content distribution and media distribution. Web site content focuses mostly on serving static pages with a combination of text and other media content while the media delivery CCDNs are dedicated to deliver high speed video form content providers such as Netflix. The majority of the system use the architecture presented in Fig 3 with proprietary implementation of cloud storage architecture, security, DNS, load balancer, CDN orchestrator and indexing mechanisms. In this section, we will analyse the current state-of-the are in commercial and academic/research based CCDN solutions.

Rackspace Cloud Files

RackSpace offers "Cloud Files" as a Cloud-based CDN service where the customers can use virtually unlimited and on-demand cloud storage and high speed content delivery over the Internet all over the world. The high-level architecture of Cloud Files is shown in Fig. 8. As "Cloud Files" is a cloud based system, it offers pay-as-you-go model which means that the users only need to pay for the amount of storage and network bandwidth based on the actual usage. The "Cloud Files" takes advantage of the Akamai Content Delivery Network to deliver the content worldwide. Akamai CDN is one of the largest CDN providers in the world and has a large number of surrogate servers around the world so that the content access latency is significantly minimized even if the customers are far away from the origin server. In terms of content hosting, Cloud Files make use of OpenStack for file storage functionality. The Cloud Files supports API to that cloud and CDN resources can be managed programmatically. The Cloud Files system uses Time-to-Live (TTL) timers to manage content that change dynamically. The dynamic content to be shared using the CDN are associated with a CDN-enabled container. The TTL of the container navigates to each file in the container. When the TTL expires, the edge servers (POPs) will synchronise with the origin server to update the changed content. It is not possible to have a TTL associate with each individual file within a CDN-enabled container.

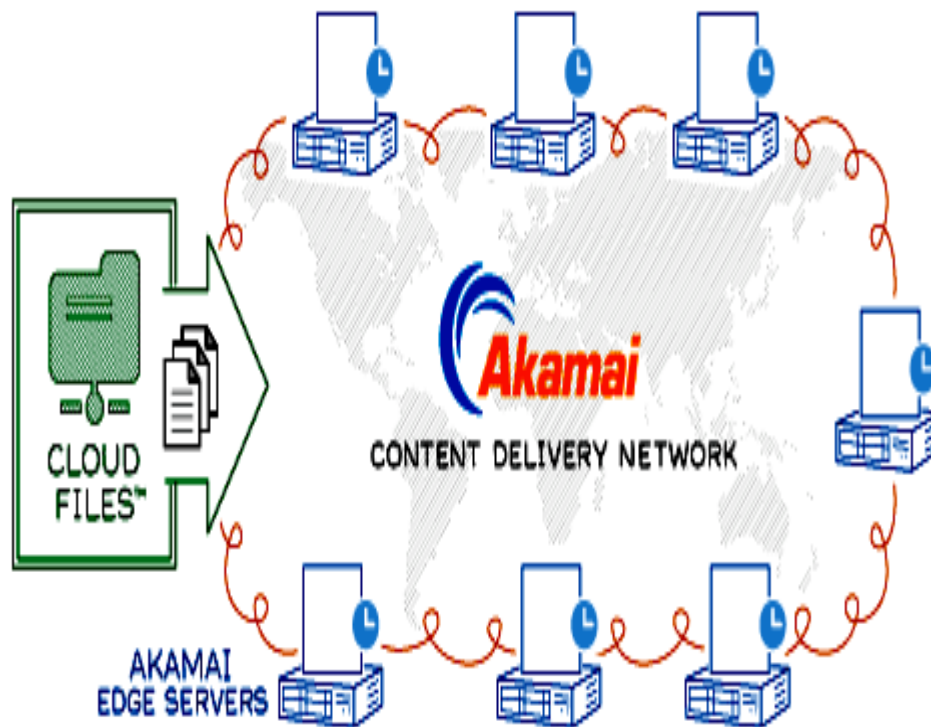


Figure 8: Architecture of Rackspace Cloud Files

Amazon CloudFront

Amazon offers CloudFront as a content delivery service that can be integrated with their widely popular Amazon Elastic Cloud Compute service. Similar to Rackspace, CloudFront is also offered as a pay-as-you-go model and supports both static and dynamic content delivery along with live media streaming functionality. Using CloudFront, the customers can store their content on the origin servers, or use the Amazon's Cloud Store service (Amazon S3). The customers can use simple APIs or the AWS Management Console to register their origin servers with Amazon CloudFront. When the customer has more than one server, he/she can use URL pattern matching to find which origin server has the content, and the customer can assign one of those origin servers as the default server. The most significant feature of Amazon's CloudFront is that it can be co-operated with several other Amazon Cloud Services. The architecture of interactions between Amazon CloudFront and other AWS services is presented in Fig 9. One of the major difference between Rackspace Cloud Files and Amazon CloudFront is that Rackspace utilizes Akamai CDN service that offers 219 CDN edge locations worldwide compared to only 32 CDN edge locations offered by Amazon. The CloudFront enables handing of dynamic content while delivering web content that change for each end-user. It uses the concept of URL pattern matching which has to be defined for the dynamic content being served to control the cache behaviour. When a URL match succeeds for a dynamic content request, the corresponding cache behaviour is invoked.

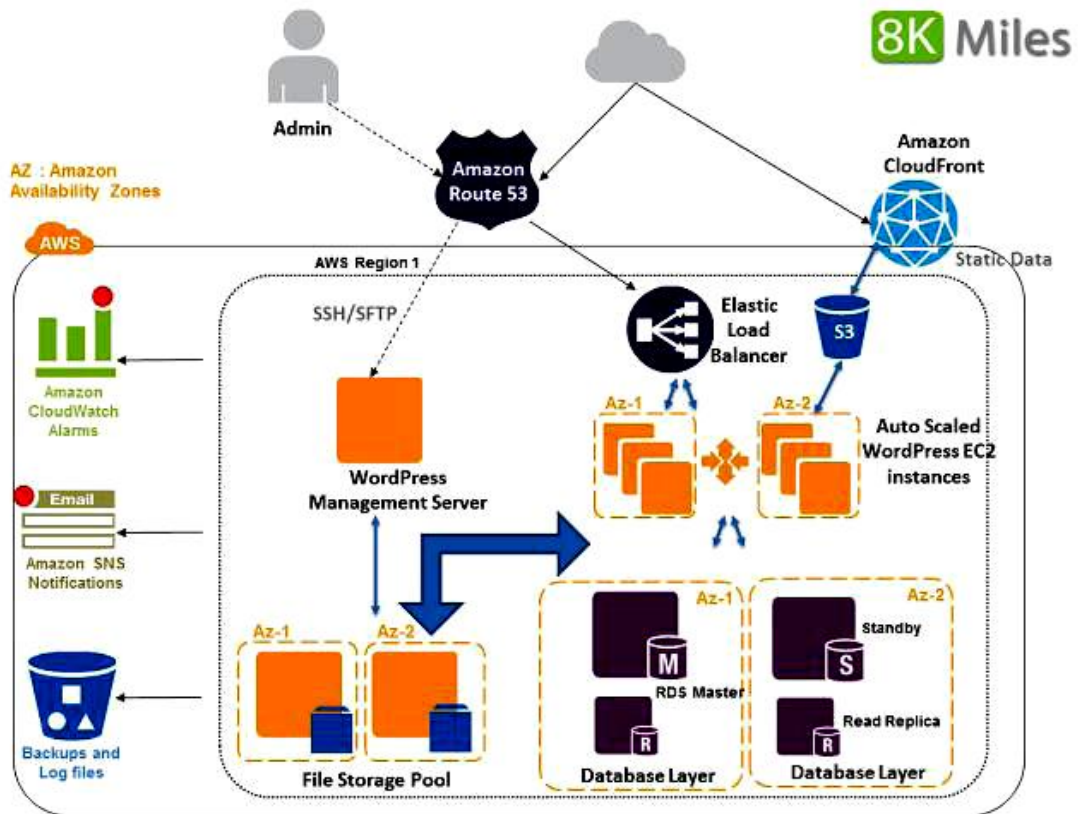


Figure 9: Integration between the CloudFront and other Amazon services

MetaCDN

MetaCDN is another content delivery provider that offers two kinds of CDN services: one for static content (e.g., websites) acceleration, and another for live multimedia streaming. Unlike other CDN providers that have their own global distributed system, MetaCDN take advantage of existing storage clouds and compute technology to support its own services. Contrary to cloud providers such as Amazon and Rackspace that offer diverse kind of additional services using their own infrastructure, MetaCDN offers its services by integrating the offerings from several other public cloud providers worldwide, thereby having in excess of 120 edge locations across the world for static content delivery. In case of live streaming, they also have more than 40 edge servers located around the world. As a consequence, MetaCDN is clearly illustrates the power and value of combining the Cloud with the CDN for optimized content delivery over the Internet. Fig 10 presents an overview of MetaCDN architecture.

The MetaCDN platform uses connectors to interface with public cloud storage providers such as Amzaon S3, Limelight networks. The connector has the basic sets of operations that are supported by most cloud storage providers. The MetaCDN also have a number of core components responsible for functioning of the service. These include the MetaCDN manager, QoS monitor, Allocator, Database and Load redirector. The allocator selects the optimal service provider. The QoS monitor keeps track of cloud storage performance and the CDN Manager tracks each user's current deployments. The database is used to store vital user and

cloud storage mapping information and finally the Load Redirector is responsible for distribute end-user requests to appropriate POP servers. The MetaCDN system also provides user interfaces and APIs to configure system via the web and programmatically.

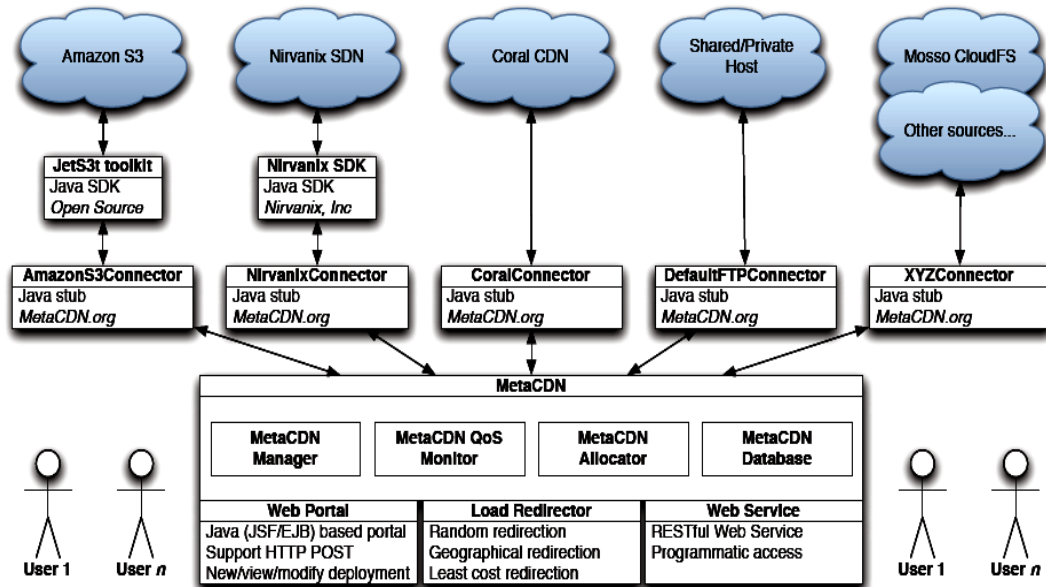


Figure : 10 METACDN Architecture

Limelight Orchestrate: Limelight Networks

Limelight is one of the biggest CDN providers in the world and offers services such as cloud storage, web acceleration and media delivery. There are some typical products the Limelight offer like “Deep Insight” which gives the customers analytic data which would be helpful for them to make business decisions. The Limelight orchestrate is a content delivery network offered by Limelight networks. This service is one of the world’s largest CDN. The Limelight orchestrate service features cloud storage, content control, security, traffic direction and mobile device content delivery. The cloud part of the system take advantage of Limelight networks cloud storage service. Fig 11 presents an overview of the orchestrate service.

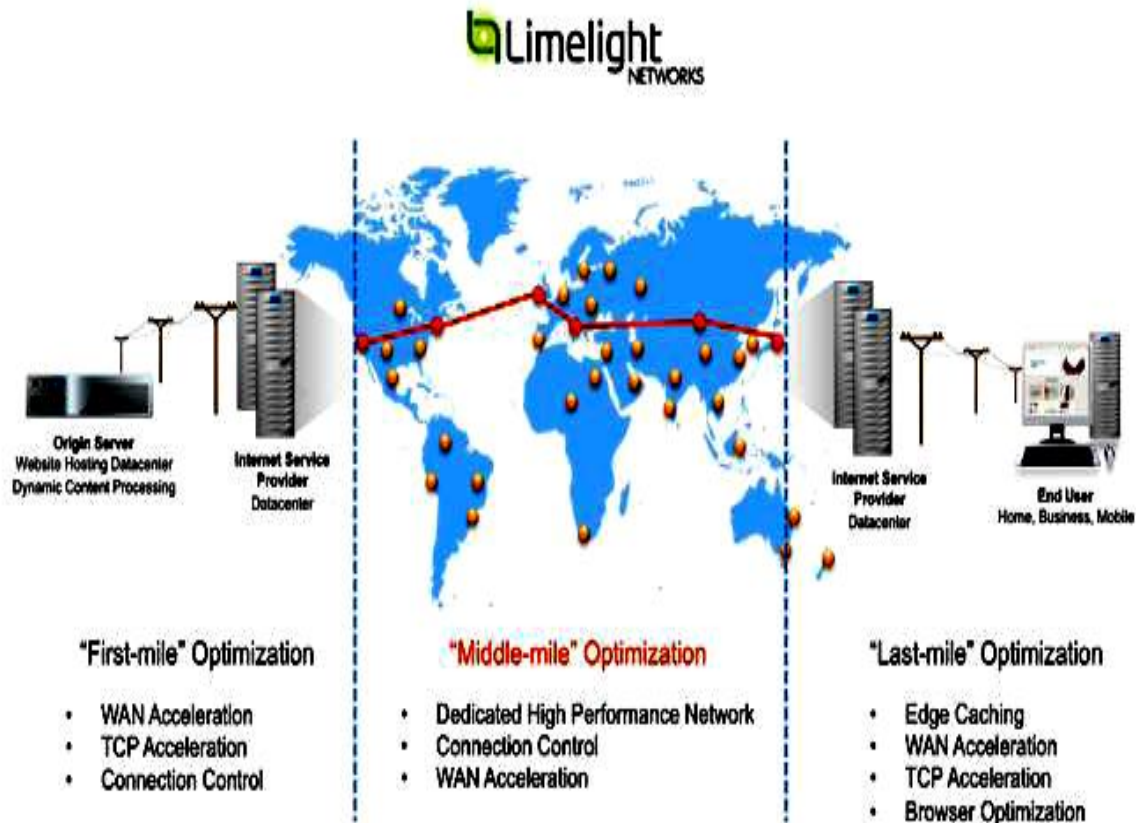


Figure: 11 Limelight Orchestrate – Overview

MediaWise Cloud

The MediaWise cloud offers a novel cloud orchestration framework where any user can become a CDN provider. As in MetaCDN, the MediaWise cloud leverages multiple cloud providers and offers pay-as-you-go model. The end user can select any public cloud provider simultaneously (e.g., Amazon and Rackspace) based on SLA, price and QoS requirements to leverage services such as compute, storage and content distribution at significantly lower costs. The main highlight of this approach is that a customer is not locked to any particular cloud and CDN provider. Compared to other cloud-based CDNs, another major highlight of the MediaWise cloud is that it supports dynamic content delivery to enable collaborative activities such as collaborative content creation, indexing, storage and retrieval.

Fig. 12 shows the reference architecture of MediaWise cloud. As can be seen in this figure, the MediaWise cloud consists of a several components. These include: content orchestrator, hybrid clouds, content access portal and the content management portal. Using the content management portal, the users (content producers) can add, delete or update content on any public clouds (e.g., Amazon and Rackspace). This content is then available to the end users via the content access portal. As mentioned previously, the MediaWise Cloud supports dynamic content creation and delivery. Using this functionality, several users using the

MediaWise clouds create multimedia content together via the content access/management portal. This dynamic content can also be annotated using keywords for efficient indexing, search and retrieval. As soon as the request the content is generated from a user(s), it is forwarded it to the MediaWise Cloud content orchestrator (MCCO) .

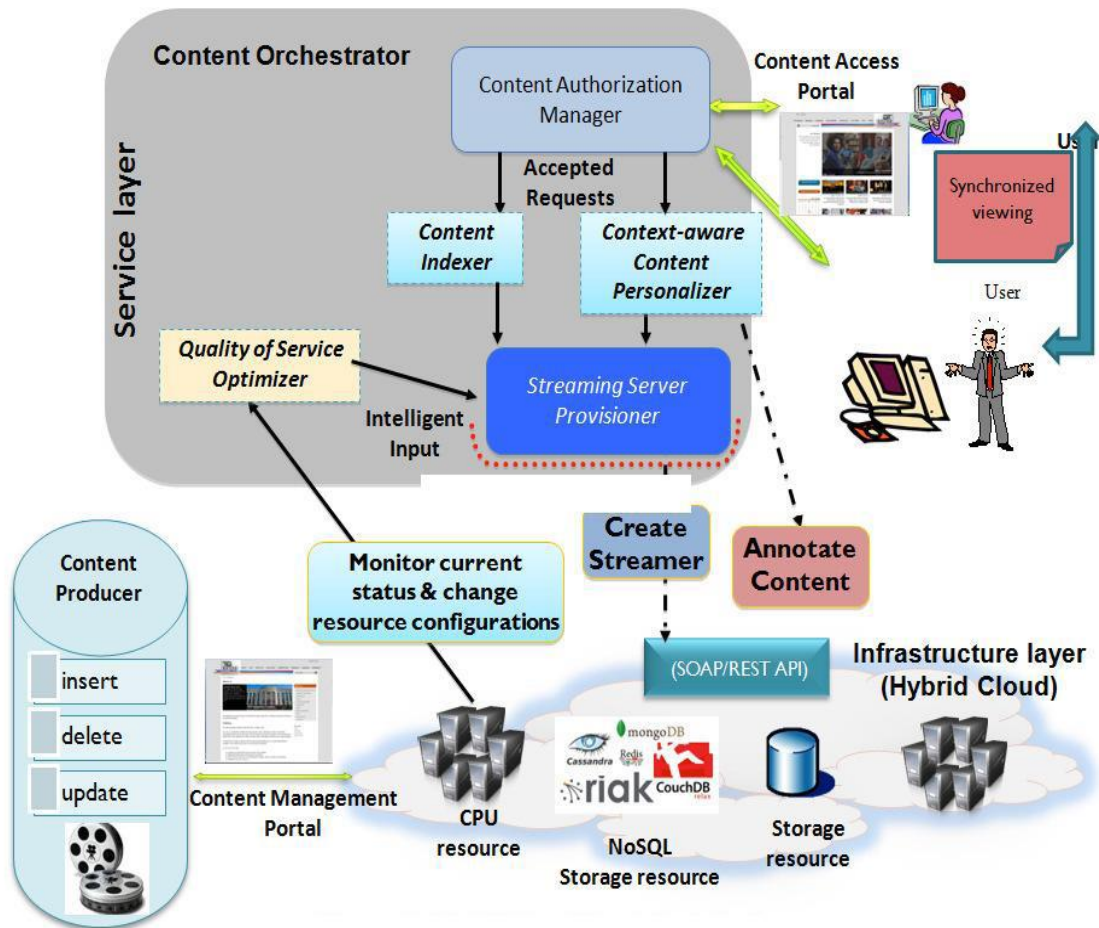


Figure: 12 The MediaWise Cloud architecture

MCCO is the heart of MediaWise cloud. It monitors hybrid clouds and provide mechanisms for QoS-aware cloud selection, scheduling and admission control. For example, as soon as the use request comes from the end user (via the content access portal) for content processing and delivery, the MCCO decides which virtual machine (VM) to provision out of several VMs running on several public clouds. This decision is based on the type of request and the QoS status of the VM on a particular public cloud. Hence, the MediaWise cloud offers QoS-based content placement, delivery as well as compute functionality that is critical in matching end-user SLAs.

SyncCast

SyncCast is a CDN, which facilitates global load balancing, utilizing tier-1 Internet backbones. SyncCast offers solutions from application development, Web hosting and Internet connectivity to deployment and systems integration. It provides solutions for delivering digital content and related data via the Internet and other media. SyncCast is designed to deliver content rapidly in a cost-effective manner. SyncCast load balances client traffic through the use of load-balancing equipments from major components such as F5, Cisco and Foundry. Thus, SyncCast allows the users to connect to the streaming servers in multiple data centers over the most efficient network path, and the users perceive better network performance. SyncCast offers P2P streaming technologies to the streaming clients. SyncCast P2P technology intelligently monitors the quality of audio/video stream to each user and it switches them to another stream source in occurrence of service quality degradation. SyncCast also provides collocation and storage services along with load balancing, firewall, and database management and managed backup services.

Netli

Netli is a privately owned company based in Mountain View, California. It is a provider of business quality Internet. Netli has computer clusters in 13 cities around the world. Netli's services are – NetLightning® for optimizing delivery of Web applications and content; Netli Offload™ to deliver infrastructure that meets enterprise requirements; NetliView™ to provide near real-time information on the performance, availability, and usage pattern of business applications; and NetliContinuity™ to gain strategic control and management of data center resources. The NetliOne platform is a global Application Delivery Network (ADN) that ensures short response time, improved visibility and control of applications over the Internet. Netli ADN services are delivered over the NetliOne platform. NetliOne platform consists of a series of globally distributed Virtual Data Centers (VDCs) and Application Access Points (AAPs), a global DNS redirection and IP address mapping system, a high performance protocol and content optimization software, an online monitoring and reporting system, and a 24X7 network operations center . The global set of Netli's VDC and AAP forms a direct bi-nodal network between origin servers and users at the edge of the Internet. Netli uses a proprietary protocol that accelerates application and content delivery bi-directionally. When a user accesses a Web application, the browser request is processed by Netli's dynamic mapping and redirection system which directs the request to a nearby Virtual Data Center (VDC). On receiving a page request, the VDC supplies the base page, simultaneously parses the base page, predicts requests for subsequent embedded objects and issues corresponding requests to the origin server. The VDC acts as the proxy for the origin server which acknowledges the browser session, and the request is forwarded to the Application Access Point (AAP) located next to the origin server. The AAP then issues the request to the origin server, obtains response and forwards it to the user through the VDC .

Accellion

Accellion, Inc. a privately held company headquartered in Palo Alto, California, is the provider of on-demand secure file transfer appliance and data management solutions. Accellion customers are industries such as advertising/media production, manufacturing, healthcare, consumer goods, higher education etc. Accellion's products are built on the SeOS (SmartEdge Operating System) technology, which is a distributed file storage and transmission infrastructure for enterprise applications. The SeOS technology enables Accellion to move, replicate, and manage large sized files. It unifies and manages multiple

storage types, across geographically dispersed locations, using a range of transport and delivery protocols. Accellion Courier Secure File Transfer Appliances (SFTA) is an on-demand file transfer solution for securely exchanging files. It sends large attachments outside of the e-mail infrastructure yet providing the convenience of e-mail to both sender and receiver. The standard workflow for transferring files using SFTA is:

- 1) A user sends large-sized files including gigabyte-sized files – through a Web interface;
- 2) Files are uploaded to the Accellion Courier Standard SFTA;
- 3) The receiver receives an e-mail with an embedded, secure HTTP link;
- 4) The recipient clicks on the link and downloads the files from Accellion Courier

Standard SFTA. In an enterprise context, files are replicated between Accellion Courier Enterprise SFTAs and recipient downloads necessary files from the closest Accellion Courier Enterprise SFTA. SFTA is deployed in parallel with the existing e-mail infrastructure to take the file transfer load off the e-mail system. Accellion ensures the authorized access of data/files by the users through deployment of a LDAP (Lightweight Directory Access Protocol) infrastructure via LDAPS (LDAP over SSL/TLS) queries. Accellion SFTA can also be customized to access the authentication and address book lookup services via HTTPS (HTTP over SSL/TLS) SOAP calls to Web Services server. Accellion also provides online desktop and server backup and recovery solutions through Accellion Backup and Recovery Solutions (BRS).

AppStream

AppStream, Inc. is a private company funded by Draper Fisher Jurvetson, JK&B Capital, Goldman Sachs, Evergreen Partners, Sun Microsystems and Computer Associates. It is a provider of technology for on-demand software distribution and software license management tools for the extended enterprises. Its product is AppStream Software 5.0, which is a software distribution and license management platform. AppStream provides solutions in four key areas: self-service software distribution, software license management, remote software access, and virtual image distribution. AppStream allows users to launch an application from a browser or from a traditional desktop shortcut. With AppStream, software can be managed as service within an enterprise. Thus, the users in an enterprise are capable of streaming and caching both desktop and enterprise applications; since all application functionalities are preserved by AppStream, including interaction with peripherals and traditionally installed applications. AppStream software divides applications into segments (streamlets) required to start the application on a client desktop, delivering the streamlets to the users as needed based upon their usage behavior. Users get the flavor of using a fully installed product; while from the business perspective, AppStream provides the high functionality of a locally installed application allowing centralized access and scalability for any enterprise. The AppStream server communicates as a traditional Web application, using HTTP, with the Software Streaming Transfer Protocol (SSTP) running over HTTP for the most efficient delivery of application segments.

EdgeStream

EdgeStream, Inc., based in Southern California is a provider of video streaming applications over the public Internet. It provides video on-demand and IPTV streaming software to enable transportation of high bit rate video over Internet. EdgeStream developed Continuous Route Optimization (CROS) and Congestion Tunnel Through (ICTT) technologies that address the

latency, packet loss, and congestion bottlenecks . Embedded applications in Consumer Electronics Devices, wireless handheld devices, IP set top boxes, and advanced digital TV's can use the EdgeStream software for video streaming. EdgeStream server software consists of Content Management and Online Reporting (CMOR) Server Software Module, EdgeStream Control Server Software Module, EdgeStream Database System Module, and EdgeStream Streaming Server Module. CMOR module manages accounts, content, and all other servers in the system. It also generates Web-based real time reports for viewing statistics and transactions from a SQL database. The control module provides necessary authority to obtain the content location information along with streaming delivery management and logging functions. The database module maintains logs for accounting and billing purpose and the streaming server module is designed for load balancing and for running on standard low cost server platforms. The EdgeStream client software provides measurement of Internet connection quality on a second by second basis . EdgeStream offers demonstration of its performance to the prospective customers through maintaining a streaming server network, and offers short term and long term video hosting services for a quick and cost effective roll out of video applications.

Globix

Globix Corporation is a provider of Internet infrastructure and network services. It offers a comprehensive suite of services from offering network bandwidth, to the management of Web applications, servers, databases, to security, media streaming, and collocation. Globix Internet infrastructure consists of both a trans-Atlantic/trans-continental IP backbone as well as an optical network throughout the Northeast and mid-Atlantic regions. Globix IP backbone connects the customers to the Internet via a high-capacity network, fully owned and operated by Globix. Globix provide four types of services: network Services, hosting Services, managed services, and media services. Under the managed services, Globix offers security, storage, messaging, disaster recovery, monitoring, application and database management services. Globix also provides media services to capture, store, host and distribute media content from live event production, encoding, presentation tools, and traffic analysis. Globix load balancing service distributes traffic among multiple servers, sending the request to the least loaded server in the dedicated server cluster. Unlike software-based load balancer, the service is built with an ASIC-based hardware architecture that provides increased traffic performance. Globix also offers Globix Traffic Analyzer which is a monitoring and reporting service. It is used to measure and provide day-to-day summary reports on the physical network and server hardware, Web and application services, and backend database performance.

Akamai

Akamai technologies evolved out of an MIT research effort aimed at solving the flash crowd problem. It is the market leader in providing content delivery services. It owns more than 20,000 servers over 1,000 networks in 71 countries . Akamai's approach is based on the observation that serving Web content from a single location can present serious problems for site scalability, reliability and performance. Hence, a system is devised to serve requests from a variable number of surrogate servers at the network edge. Akamai servers deliver static, dynamic content and streaming audio and video. Akamai's infrastructure handles flash crowds by allocating more servers to sites experiencing high load, while serving all clients from nearby servers. The system directs client requests to the nearest available server likely to have the requested content. Akamai provides automatic network control through the

mapping technique (i.e. the direction of request to content servers), which uses a dynamic, fault-tolerant DNS system. The mapping system resolves a hostname based on the service requested, user location and network status. It also uses DNS for network load-balancing. Akamai name servers resolve hostnames to IP addresses by mapping requests to a server. Akamai agents communicate with certain border routers as peers; the mapping system uses BGP information to determine network topology. The mapping system in Akamai combines the network topology information with live network statistics – such as traceroute data – to provide a detailed, dynamic view of network structure and quality measures for different mappings.

Akamai's DNS-based load balancing system continuously monitors the state of services and their servers and networks. To monitor the entire system's health end-to-end, Akamai uses agents that simulate end-user behavior by downloading Web objects and measuring their failure rates and download times. Akamai uses this information to monitor overall system performance and to automatically detect and suspend problematic data centers or servers. Each of the content servers frequently reports its load to a monitoring application, which aggregates and publishes load reports to the local DNS server. That DNS server then determines which IP addresses (two or more) to return when resolving DNS names. If a certain server's load exceeds a certain threshold, the DNS server simultaneously assigns some of the server's allocated content to additional servers. If the server's load exceeds another threshold, the server's IP address is no longer available to clients. The server can thus shed a fraction of its load when it experiences moderate to high load. The monitoring system in Akamai also transmits data center load to the top-level DNS resolver to direct traffic away from overloaded data centers. In addition to load balancing, Akamai's monitoring system provides centralized reporting on content service for each customer and content server. This information is useful for network operational and diagnostic purposes.

CDN Name	Service Type	Coverage	Products/Solutions (If any)
Accellion www.accellion.com	Provides large file delivery service	Covers industries such as advertising/media production, healthcare, manufacturing, consumer goods, higher education etc.	Accellion Courier Secure File Transfer Appliances (SFTA) for on-demand and secure exchange of large-size files, and Accellion Backup and Recovery Services (BRS) as online desktop and server backup and recovery solution
Akamai www.akamai.com	Provides CDN service, including streaming	Covers 85% of the market. 20,000 servers in nearly 1,000 networks in 71 countries. It handles 20% of total Internet traffic today.	Edge Platform for handling static as well as dynamic content, Edge Control for managing applications, and Network Operations Control Center (NOCC)
AppStream www.appstream.com	Provides on demand software distribution and software license management tools for extended enterprises	Small business to Fortune 1,000 corporations, educational institutions and government	AppStream Software V5.0 including AppStream server, client and end-user application portal
EdgeStream www.edgestream.com	Provides disrupted video streaming applications over the public Internet	Provides video streaming over consumer cable or ADSL modem connections around the globe, even over paths that have 20 router hops between server and end-user	EdgeStream video on-demand and IPTV Streaming software for video streaming
Globix www.globix.com	Provides Internet infrastructure and network services	Consists of both a trans-Atlantic/trans-continental IP backbone as well as an optical network throughout the Northeast and mid-Atlantic regions. It has more than 1200 customers	N/A
Limelight Networks www.limelightnetworks.com	Provides distributed on-demand and live delivery of video, music, games and download	Surrogate servers located in 72 locations around the world	Limelight ContentEdge for distributed content delivery via HTTP, Limelight MediaEdge Streaming for distributed video and music delivery via streaming, and Limelight Custom CDN for custom distributed delivery solutions
Mirror Image www.mirror-image.com	Provides content delivery, streaming media, Web computing and reporting services	Surrogate servers located in 22 countries around the world.	N/A
Netli www.netli.com	Provides business quality Internet services over the NetliOne platform	Computer clusters in 13 cities around the world.	N/A
SyncCast www.synccast.com	Provides solutions for delivering digital content and related data via Internet	It covers motion picture companies, media production, music industry, and online retailers.	N/A

Table 2: Summary of existing commercial content delivery networks

Codeen

Codeen is an academic CDN test-bed developed at Princeton university. It is primarily used to support services delivered the Planet Lab project, a global research networks that supports developments of new network services. Codeen has many proxy nodes distributed at various planet lab node locations. The proxy perform the role of POPs and request redirectors. A number of related projects that use the Codeen CDN include web-based content distribution service, name lookup, synchronisation tools, activity monitoring and visualisation tools. A Codeen user sets their cache to a nearby high bandwidth plant lab node location. Request to the codeen node at the location is directed to the most appropriate member of the planet lab system that has a cached copy of the file. This file is forwarded to the client. However, this system lacks support for dynamic content distribution.

COMODIN

COMODIN (Coperative Media On-Demand on the InterNet) is an academic CDN providing streaming media service on current Internet infrastructure. COMODIN enables a collaborative experience while streaming media content via the Internet. For examples, a group of users can coordinate the state of a media file (e.g. play, pause etc). COMODIN follows a two layer architecture comprising of a base plane and a distributed set of playback components. The system employs IP-multicast to stream data across multiple clients. This academic CDN focuses more on content control collaboratively rather than content creation or distribution.

1.1. CoDaaS: An Experimental Cloud-Centric Content Delivery Platform for User-Generated Contents

CoDaaS is a cloud-centric content delivery system focused on distributing user-generated content in the most economical fashion while respecting the Quality-of-Service (QoS) requirements. It enables on-demand virtual content delivery to a targeted set of users. The system is built of hybrid cloud environments. Fig 13 presents an architecture overview of the CoDaaS system.

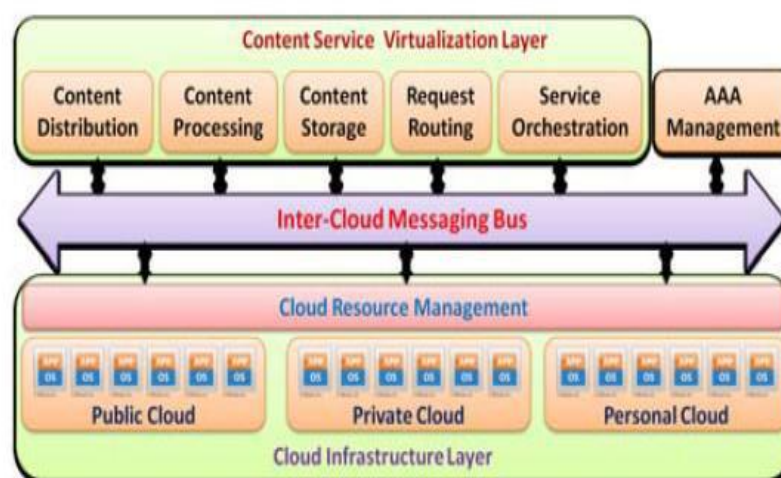


Figure: 13 CoDaaS system architecture

The CoDaaS architecture consists of three layers namely the cloud layer, the content service virtualizations layer and the security layer. The cloud layer comprises the hybrid cloud from which resources are used to develop a content distribution overlay. It also has a set of media rendering engines, managerial and service orchestration function. The content service visualization does the operation of content distribution, processing, storage and routing. This layer performs the function of a typical CDN. The security module is responsible to ensure authorized, authenticated and accountable access to resources across hybrid clouds. Finally, the inter-cloud messaging bus is employed to integrate all participating components into an integrated media application.

1.2. Globule – Globule is an open-source collaborative CDN developed at the Vrije Universities in Amsterdam. Globule is available for public use under open source license. Globule aims to allow Web content providers to organize together and operate their own world-wide hosting platform. It provides replication of content, monitoring of servers and redirecting of client requests to available replicas. In Globule, a site is defined as a collection of documents that belong to one specific user (the site's owner) and a server is a process running on a machine connected to a network, which executes an instance of the Globule software. Each server is capable of hosting one or more sites and to deliver content to the clients. Globule takes inter-node latency as the proximity measure. This metric is used to optimally place replicas to the clients, and to redirect the clients to an appropriate replica server. Globule is implemented as a third-party module for the Apache HTTP Server that allows any given server to replicate its documents to other Globule servers. To replicate content, content providers only need to compile an extra module into their Apache server and edit a simple configuration file. Globule automatically replicates content and redirects clients to a nearby replica. This can improve the site's performance, maintain the site available to its clients even if some servers are down, and to a certain extent help to resist to flash crowds and the Slashdot effect. Though Globule addresses issues like client localization, replica placement and Web application replication, a suitable mechanism is yet to develop in order to enforce the fair resource usage among participants. Moreover, the reliability of Globule system and the way it reacts to flash crowd events should be improved further.

1.3. Coral – Coral is a free, P2P content distribution network designed to mirror Web content. Coral is designed to use the bandwidth of volunteers to avoid slashdotting and to reduce the load on websites and other Web content providers in general. During beta testing, the Coral node network is hosted on PlanetLab (a large scale distributed research network of 755 nodes at 363 sites), instead of third party volunteer systems. The source code of CoralCDN is freely available under the terms of the GNU GPL. To use CoralCDN, a content publisher has to append “.nyud.net:8090” to the hostname in a URL. Clients are redirected to the nearby Coral Web caches transparently through DNS redirection. Coral Web caches cooperate to transfer data from nearby peers whenever possible, minimizing both the load on the origin Web server and the latency perceived by the user. Coral allows nodes to nearby cached objects without redundantly querying more distant nodes. It also prevents the creation of hotspots even under degenerate loads. Coral uses an indexing abstraction called Distributed Sloppy Hash Table (DSHT), which is a variant of DHTs. Coral's architecture is based on clusters of well-connected machines. Clusters are exposed in the interface to higher-level software, and in fact form a crucial part of the DNS redirection mechanism. Performance measurements of Coral demonstrate that it allows under-provisioned Web sites

to achieve dramatically higher capacity, and its clustering provides quantitatively better performance than locality-unaware systems . Thus Coral can handle large amounts of traffic with reasonable performance. But the main drawback of Coral is that it is very difficult to select replica placement. Coral also can not handle dynamically generated content. Most significantly, the use of DNS-based requestrouting in Coral limits its scalability. Moreover, the DNS redirection mechanism in Coral does not consider the heterogeneity of proxies while performing request-routing. Since volunteer nodes are used in Coral, a suitable mechanism should be in place to ensure the consistency and integrity of cached data. Coral also can not prevent a significant amount of traffic to be directed to the origin server at the start of the flash crowds.

CDN Name	Service Type	Coverage	Products/Solutions (if any)
CoDeeN www.codeen.cs.princeton.edu	Provides caching of content and redirection of HTTP requests	CoDeeN is an academic testbed content distribution network built on top of PlanetLab	N/A
COMODIN http://lisdip.deis.unical.it/research/index.html	Provides collaborative media playback service	COMODIN is an academic streaming CDN deployed on an international testbed	N/A
Coral www.coralcdn.org	Provides content replication in proportion to the content's popularity	Coral is a free peer-to-peer CDN. During beta testing, the Coral node network is hosted on PlanetLab [111], a large scale distributed research network of 400 servers, instead of third party volunteer systems. Of those 400 servers, about 275 are currently running Coral	N/A
Globule www.globule.org	Provides replication of content, monitoring of servers and redirecting client requests to available replicas	Globule is an open source collaborative CDN	N/A

Table 3: Summary of existing academic content delivery networks

Chapter 7

Exploring the Taxonomy of CDN

This section presents a detailed taxonomy of CDNs with respect to four different issues/factors. As shown in Figure 14, they are – CDN composition, content distribution and management, request-routing, and performance measurement.

The first issue covers several aspects of CDNs related to organization and formation. This classifies the CDNs with respect to their structural attributes. The second issue pertains to the content distribution mechanisms in the CDNs. It describes the content distribution and management approaches of CDNs in terms of surrogate placement, content selection and delivery, content outsourcing, and organization of caches/replicas. The third issue considered relates to the request-routing algorithms and request-routing methodologies in the existing CDNs. The final issue emphasizes on the performance measurement of CDNs and looks into the performance metrics and network statistics acquisition techniques used for CDNs. Each of the issues covered in the taxonomy is an independent field, for which extensive research are to be conducted.

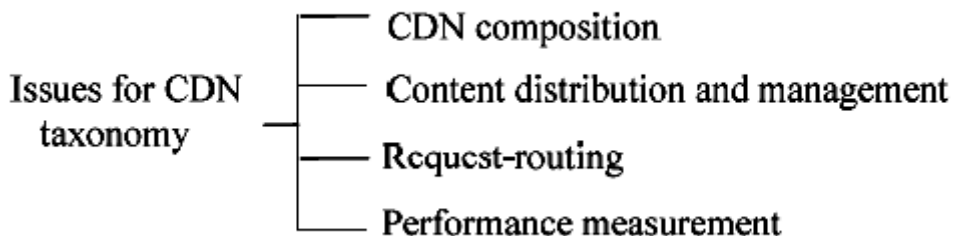


Figure 14: Issues for CDN taxonomy

1. CDN composition

A CDN typically incorporates dynamic information about network conditions and load on the cache servers, to redirect request and balance loads among surrogates. The analysis on the structural attributes of a CDN reveals the fact that CDN infrastructural components are closely related to each other. Moreover, the structure of a CDN varies depending on the content/services it provides to its users. Within the structure of a CDN, a set of surrogates is used to build the content-delivery infrastructure, some combinations of relationships and mechanisms are used for redirecting client requests to a surrogate and interaction protocols are used for communications among the CDN elements. Figure 15 shows the taxonomy based on the various structural characteristics of CDNs. These characteristics are central to the composition of a CDN and they address the organization, types of servers used, relationships and interactions among CDN components, as well as the different content and services provided by the CDNs.

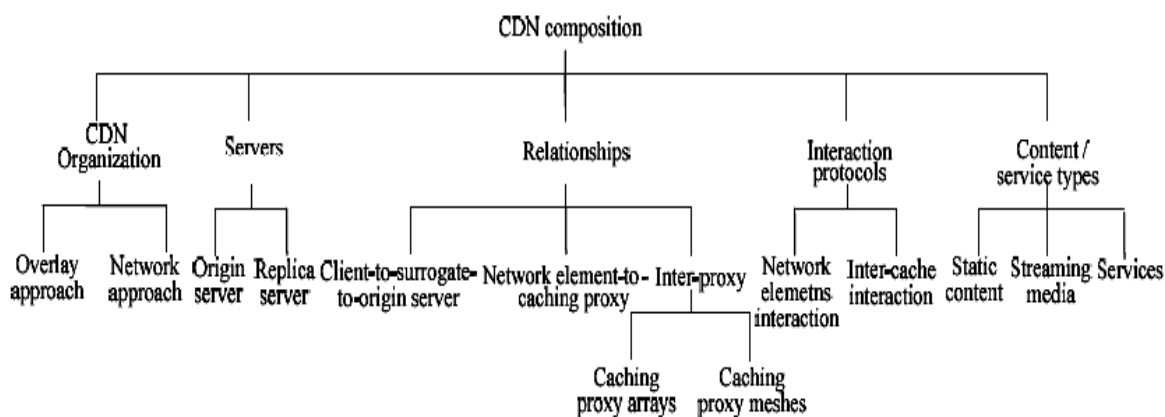


Figure 15: CDN composition taxonomy

CDN organization – There are two general approaches to building CDNs: overlay and network approach . In the overlay approach, application-specific servers and caches at several places in the network handle the distribution of specific content types (e.g. web content, streaming media, and real time video). Other than providing the basic network connectivity and guaranteed QoS for specific request/traffic, the core network components such as routers and switches play no active role in content delivery. Most of the commercial CDN providers such as Akamai, AppStream, and Limelight Networks follow the overlay approach for CDN organization. These CDN providers replicate content to thousands of cache server worldwide. When content requests are received from end-users, they are redirected to the nearest CDN server, thus improving Web site response time. As the CDN providers need not to control the underlying network infrastructure elements, the management is simplified in an overlay approach and it opens opportunities for new services. In the network approach, the network components including routers and switches are equipped with code for identifying specific application types and for forwarding the requests based on predefined policies. Examples of this approach include devices that redirect content requests to local caches or switch traffic coming to data centers to specific servers optimized to serve specific content types. Some CDN (e.g. Akamai, Mirror Image) use both the network and overlay approach for CDN organization. In such case, a network element (e.g. switch)

can act at the front end of a server farm and redirects the content request to a nearby application-specific surrogate server.

Servers – The servers used by a CDN are of two types – origin server and replica server. The server where the definitive version of a resource resides is called origin server, which is updated by the content provider. A server is called a replica server when it is holding a replica of a resource but may act as an authoritative reference for client responses. The origin server communicates with the distributed replica servers to update the content stored in it. A replica server in a CDN may serve as a media server, Web server or as a cache server. A media server serves any digital and encoded content. It consists of media server software. Based on client requests, a media server responds to the query with the specific video or audio clip. A Web server contains the links to the streaming media as well as other Web-based content that a CDN wants to handle. A cache server makes copies (i.e. caches) of content at the edge of the network in order to bypass the need of accessing origin server to satisfy every content request.

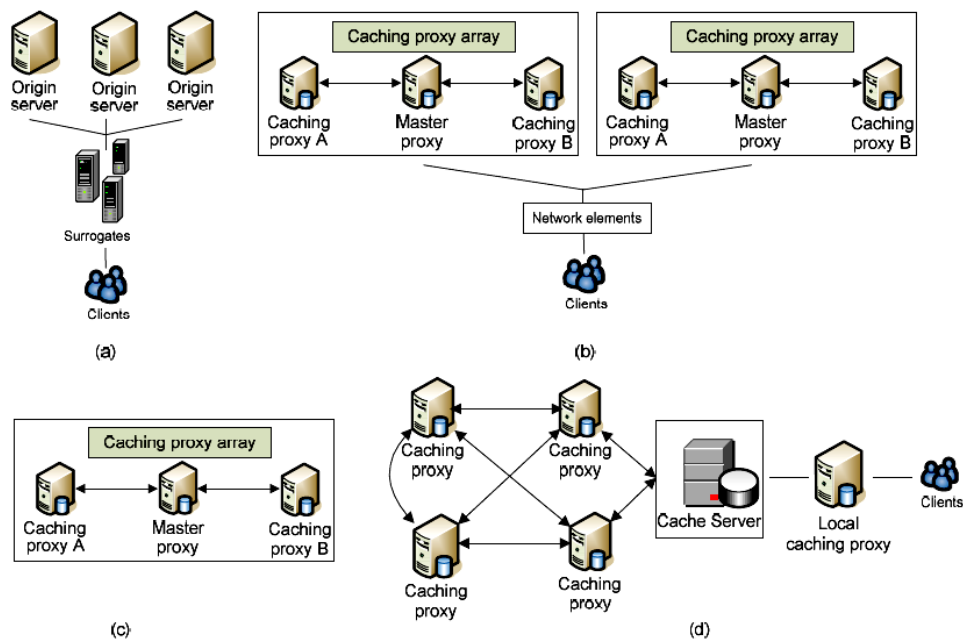


Figure 16: (a) Client-to-surrogate-to-origin server; (b) Network element-to-caching proxy; (c) Caching proxy arrays; (d) Caching proxy meshes

Relationships – The complex distributed architecture of a CDN exhibits different relationships between its constituent components. In this section, we try to identify the relationships that exist in the replication and caching environment of a CDN. The graphical representations of these relationships are shown in Figure 16. These relationships involve components such as clients, surrogates, origin server, proxy caches and other network elements. These components communicate to replicate and cache content within a CDN. Replication involves creating and maintaining duplicate copy of some content on different computer system. It typically involves ‘pushing’ content from the origin server to the replica

servers . On the other hand, caching involves storing cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests.

In a CDN environment, the basic relationship for content delivery is among the client, surrogates and origin servers. A client may communicate with surrogate server(s) for requests intended for one or more origin servers. Where a surrogate is not used, the client communicates directly with the origin server. The communication between a user and surrogate takes place in a transparent manner, as if the communication is with the intended origin server. The surrogate serves client requests from its local cache or acts as a gateway to the origin server. The relationship among client, surrogates and the origin server is shown in Figure 16(a).

CDNs can be formed using a network approach, where logic is deployed in the network elements (e.g. router, switch) to forward traffic to servers/proxies that are capable of serving client requests. The relationship in this case is among the client, network element and caching servers/proxies (or proxy arrays), which is shown in Figure 16(b). Other than these relationships, caching proxies within a CDN may communicate with each other. A proxy cache is an application-layer network service for caching Web objects. Proxy caches can be simultaneously accessed and shared by many users. A key distinction between the CDN proxy caches and ISP-operated caches is that the former serve content only for certain content provider, namely CDN customers, while the latter cache content from all Web sites.

Based on inter-proxy communication , caching proxies can be arranged in such a way that proxy arrays (Figure 16(c)) and proxy meshes (Figure 16(d)) are formed. A caching proxy array is a tightly-coupled arrangement of caching proxies. In a caching proxy array, an authoritative proxy acts as a master to communicate with other caching proxies. A user agent can have relationship with such an array of proxies. A caching proxy mesh is a loosely-coupled arrangement of caching proxies. Unlike the caching proxy arrays, proxy meshes are created when the caching proxies have one-to-one relationship with other proxies. Within a caching proxy mesh, communication can happen at the same level between peers, and with one or more parents . A cache server acts as a gateway to such a proxy mesh and forwards client requests coming from client's local proxy.

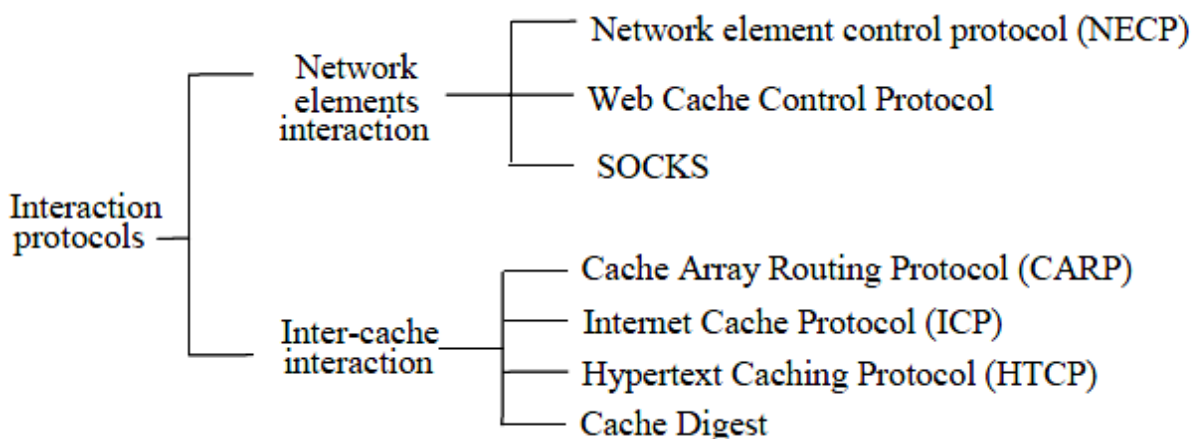


Figure 17: Various interaction protocols

Interaction protocols – Based on the communication relationships described earlier, we can identify the interaction protocols that are used for interaction among CDN elements. Such interactions can be broadly classified into two types: interaction among network elements and interaction between caches. Figure 17 shows various interaction protocols that are used in a CDN for interaction among CDN elements. Examples of protocols for network element interaction are Network Element Control Protocol (NECP) , Web Cache Coordination Protocol and SOCKS . On the other hand, Cache Array Routing Protocol (CARP) , Internet Cache Protocol (ICP) , Hypertext Caching protocol (HTCP) , and Cache Digest are the examples of inter-cache interaction protocols. These protocols are briefly described here:

NECP: The Network Element Control Protocol (NECP) is a lightweight protocol for signaling between servers and the network elements that forward traffic to them. The network elements consist of a range of devices, including content-aware switches and load-balancing routers. NECP allows network elements to perform load balancing across a farm of servers and redirection to interception proxies. However, it does not dictate any specific load balancing policy. Rather, this protocol provides methods for network elements to learn about server capabilities, availability and hints as to which flows can and cannot be served. Hence, network elements gather necessary information to make load balancing decisions. Thus, it avoids the use of proprietary and mutually incompatible protocols for this purpose. NECP is intended for use in a wide variety of server applications, including for origin servers, proxies, and interception proxies. It uses TCP as the transport protocol. When a server is initialized, it establishes a TCP connection to the network elements using a well known port number. Messages can then be sent bi-directionally between the server and network element. All NECP messages consist of a fixed-length header containing the total data length and variable length data. Most messages consist of a request followed by a reply or acknowledgement. Receiving a positive acknowledgement implies the recording of some state in a peer. This state can be assumed to remain in that peer until the state expires or the peer crashes. In other words, this protocol uses a ‘hard state’ model. Application level KEEPALIVE messages are used to detect a dead peer in such communications. When a node detects that its peer has been crashed, it assumes that all the states in that peer need to be reinstalled after the peer is revived.

WCCP: The Web Cache Coordination Protocol (WCCP) specifies interaction between one or more routers and one or more Web-caches. It runs between a router functioning as a redirecting network element and interception proxies. The purpose of such interaction is to establish and maintain the transparent redirection of selected types of traffic flow through a group of routers. The selected traffic is redirected to a group of Webcaches in order to increase resource utilization and to minimize response time. WCCP allows one or more proxies to register with a single router to receive redirected traffic. This traffic includes user requests to view pages and graphics on World Wide Web servers, whether internal or external to the network, and the replies to those requests. This protocol allows one of the proxies, the designated proxy, to dictate to the router how redirected traffic is distributed across the caching proxy array. WCCP provides the means to negotiate the specific method used to distribute load among Web caches. It also provides methods to transport traffic between router and cache.

SOCKS: The SOCKS protocol is designed to provide a framework for client-server applications in both the TCP and UDP domains to conveniently and securely use the services of a network firewall . The protocol is conceptually a "shim-layer" between the application layer and the transport layer, and as such does not provide network-layer gateway services,

such as forwarding of ICMP messages. When used in conjunction with a firewall, SOCKS provides an authenticated tunnel between the caching proxy and the firewall. In order to implement SOCKS protocol, TCP-based client applications are recompiled so that they can use the appropriate encapsulation routines in SOCKS library. When connecting to a cacheable content behind firewall, a TCP-based client has to open a TCP connection to the SOCKS port on the SOCKS server system. Upon successful establishment of the connection, a client negotiates for the suitable method for authentication, authenticates with the chosen method and sends a relay request. SOCKS server in turn establishes the requested connection or rejects it based on the evaluation result of the connection request.

CARP: The Cache Array Routing Protocol (CARP) is a distributed caching protocol based on a known list of loosely coupled proxy servers and a hash function for dividing URL space among those proxies. An HTTP client implementing CARP can route requests to any member of the Proxy Array. The proxy array membership table is defined as a plain ASCII text file retrieved from an Array Configuration URL. The hash function and the routing algorithm of CARP take a member proxy defined in the proxy array membership table, and make an on-the-fly determination about the proxy array member which should be the proper container for a cached version of a resource pointed to by a URL. Since requests are sorted through the proxies, duplication of cache content is eliminated and global cache hit rates are improved. Downstream agents can then access a cached resource by forwarding the proxied HTTP request for the resource to the appropriate proxy array member.

ICP: The Internet Cache Protocol (ICP) is a lightweight message format used for inter-cache communication. Caches exchange ICP queries and replies to gather information to use in selecting the most appropriate location in order to retrieve an object. Other than functioning as an object location protocol, ICP messages can also be used for cache selection. ICP is a widely deployed protocol. Although, Web caches use HTTP for the transfer of object data, most of the caching proxy implementation supports it in some form. It is used in a caching proxy mesh to locate specific Web objects in neighboring caches. One cache sends an ICP query to its neighbors and the neighbors respond with an ICP reply indicating a 'HIT' or a 'MISS'. Failure to receive a reply from the neighbors within a short period of time implies that the network path is either congested or broken. Usually, ICP is implemented on top of UDP in order to provide important features to Web caching applications. Since UDP is an uncorrected network transport protocol, an estimate of network congestion and availability may be calculated by ICP loss. This sort of loss measurement together with the round-trip-time provides a way to load balancing among caches.

HTCP: The Hypertext Caching Protocol (HTCP) is a protocol for discovering HTTP caches, cached data, managing sets of HTTP caches and monitoring cache activity. HTCP is compatible with HTTP 1.0, which permits headers to be included in a request and/or a response. This is in contrast with ICP, which was designed for HTTP 0.9. HTTP 0.9 allows specifying only a URI in the request and offers only a body in the response. Hence, only the URI without any headers is used in ICP for cached content description. Moreover, it also does not support the possibility of multiple compatible bodies for the same URI. On the other hand, HTCP permits full request and response headers to be used in cache management. HTCP also expands the domain of cache management to include monitoring a remote cache's additions and deletions, requesting immediate deletions, and sending hints about Web objects such as the third party locations of cacheable objects or the measured uncacheability or unavailability of Web objects. HTCP messages may be sent over UDP, or TCP. HTCP agents must not be isolated from network failure and delays. An HTCP agent should be

prepared to act in useful ways in the absence of response or in case of lost or damaged responses.

Cache Digest: Cache Digest is an exchange protocol and data format. Cache digests provide a solution to the problems of response time and congestion associated with other inter-cache communication protocols such as ICP and HTCP. They support peering between cache servers without a request response exchange taking place. Instead, other servers who peer with it fetch a summary of the content of the server (i.e. the Digest). When using cache digests it is possible to accurately determine whether a particular server caches a given URL. It is currently performed via HTTP. A peer answering a request for its digest will specify an expiry time for that digest by using the HTTP Expires header. The requesting cache thus knows when it should request a fresh copy of that peer's digest. In addition to HTTP, Cache Digests could be exchanged via FTP. Although the main use of Cache Digests is to share summaries of which URLs are cached by a given server, it can be extended to cover other data sources. Cache Digests can be a very powerful mechanism to eliminate redundancy and making better use of Internet server and bandwidth resources.

Content/service types – CDN providers host third-party content for fast delivery of any digital content, including – static content, streaming media (e.g. audio, real time video) and varying content services (e.g. directory service, e-commerce service, and file transfer service). The sources of content are large enterprises, web service providers, media companies, and news broadcasters. Variation in content and services delivered requires the CDN to adopt application-specific characteristics, architectures and technologies. Due to this reason, some of the CDNs are dedicated for delivering particular content and/or services. Here we analyze the characteristics of the content/service types to reveal their heterogeneous nature.

Static content: Static HTML pages, images, documents, software patches, audio and/or video files fall into this category. The frequency of change for the static content is low. All CDN providers support this type of content delivery. This type of content can be cached easily and their freshness can be maintained using traditional caching technologies.

Streaming media: Streaming media delivery is challenging for CDNs. Streaming media can be live or on demand. Live media delivery is used for live events such as sports, concerts, channel, and/or news broadcast. In this case, content is delivered 'instantly' from the encoder to the media server, and then onto the media client. In case of on-demand delivery, the content is encoded and then is stored as streaming media files in the media servers. The content is available upon requests from the media clients. On-demand media content can include audio and/or video on-demand, movie files and music clips. Streaming servers are adopted with specialized protocols for delivery of content across the IP network.

Services: A CDN can offer its network resources to be used as a service distribution channel and thus allows the value-added services providers to make their application as an Internet infrastructure service. When the edge servers host the software of value-added services for content delivery, they may behave like transcoding proxy servers, remote callout servers, or surrogate servers. These servers also demonstrate capability for processing and special hosting of the value-added Internet infrastructure services. Services provided by CDNs can be directory, Web storage, file transfer, and e-commerce services. Directory services are provided by the CDN for accessing the database servers. Users query for certain data is directed to the database servers and the results of frequent queries are cached at the edge servers of the CDN. Web storage service provided by the CDN is meant for storing content at

the edge servers and is essentially based on the same techniques used for static content delivery. File transfer services facilitate the worldwide distribution of software, virus definitions, movies on-demand, highly detailed medical images etc. All these contents are static by nature. Web services technologies are adopted by a CDN for their maintenance and delivery. E-commerce is highly popular for business transactions through the Web. Shopping carts for e-commerce services can be stored and maintained at the edge servers of the CDN and online transactions (e.g. third-party verification, credit card transactions) can be performed at the edge of CDNs. To facilitate this service, CDN edge servers should be enabled with dynamic content caching for e-commerce sites.

2. Content distribution and management

Content distribution and management is strategically vital in a CDN for efficient content delivery and for overall performance. Content distribution includes – the placement of surrogates to some strategic positions so that the edge servers are close to the clients; content selection and delivery based on the type and frequency of specific user requests and content outsourcing to decide which outsourcing methodology to follow. Content management is largely dependent on the techniques for cache organization (i.e. caching techniques, cache maintenance and cache update). The content distribution and management taxonomy is shown in Figure 18.

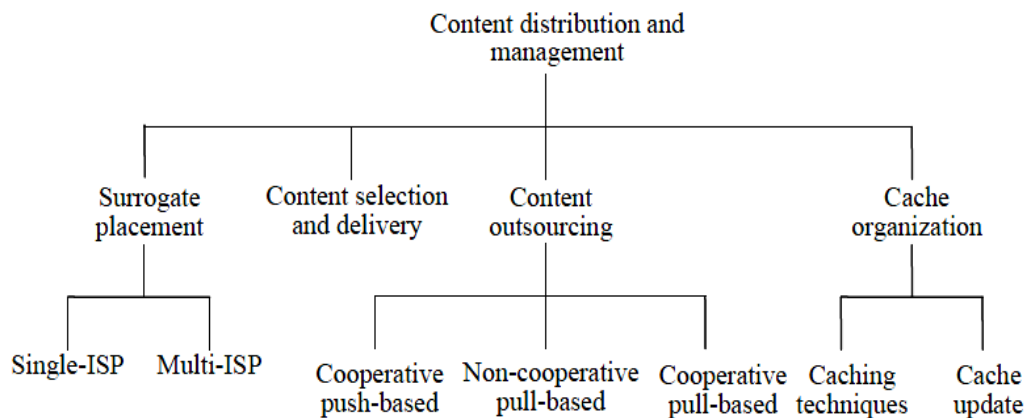


Figure18: Content distribution and management taxonomy

Surrogate placement – Since location of surrogate servers is closely related to the content delivery process, it puts extra emphasis on the issue of choosing the best location for each surrogate. The goal of optimal surrogate placement is to reduce user perceived latency for accessing content and to minimize the overall network bandwidth consumption for transferring replicated content from servers to clients. The optimization of both of these metrics results in reduced infrastructure and communication cost for the CDN provider. Therefore, optimal placement of surrogate servers enables a CDN to provide high quality services and low CDN prices. In this context, some theoretical approaches such as minimum k-center problem , k-hierarchically wellseparated trees (k-HST) have been proposed. These approaches model the server placement problem as the center placement problem which is

defined as follows: for the placement of a given number of centers, minimize the maximum distance between a node and the nearest center. k-HST algorithm solves the server placement problem according to graph theory. In this approach, the network is represented as a graph $G(V,E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of links. The algorithm consists of two phases. In the first phase, a node is arbitrarily selected from the complete graph (parent partition) and all the nodes which are within a random radius from this node form a new partition (child partition). The radius of the child partition is a factor of k smaller than the diameter of the parent partition. This process continues until each of the nodes is in a partition of its own. Thus the graph is recursively partitioned and a tree of partitions is obtained with the root node being the entire network and the leaf nodes being individual nodes in the network. In the second phase, a virtual node is assigned to each of the partitions at each level. Each virtual node in a parent partition becomes the parent of the virtual nodes in the child partitions and together the virtual nodes form a tree. Afterwards, a greedy strategy is applied to find the number of centers needed for the resulted k-HST tree when the maximum center-node distance is bounded by D . On the other hand, the minimum K-center problem is NP-complete

It can be described as follows: (1) Given a graph $G(V, E)$ with all its edges arranged in non-decreasing order of edge cost $c: c(e_1) \leq c(e_2) \leq \dots \leq c(e_m)$, construct a set of square graphs $G^2_1, G^2_2, \dots, G^2_m$. Each square graph of G , denoted by G^2 is the graph containing nodes V and edges (u, v) wherever there is a path between u and v in G . (2) Compute the maximal independent set M_i for each G^2_i . An independent set of G^2 is a set of nodes in G that are at least three hops apart in G and a maximal independent set M is defined as an independent set V' such that all nodes in $V - V'$ are at most one hop away from nodes in V' . (3) Find smallest i such that $M_i \leq K$, which is defined as j . (4) Finally, M_j is the set of K center.

Due to the computational complexity of these algorithms, some heuristics such as Greedy replica placement and Topology-informed placement strategy have been developed. These suboptimal algorithms take into account the existing information from CDN, such as workload patterns and the network topology. They provide sufficient solutions with lower computation cost. The greedy algorithm chooses M servers among N potential sites. In first iteration, the cost associated with each site is computed in the first iteration. It is assumed that access from all clients converges to the site under consideration. Hence, the lowest cost site is chosen. In the second iteration, the greedy algorithm searches for a second site (yielding the next lowest cost) in conjunction with the site already chosen. The iteration continues until M servers have been chosen. The greedy algorithm works well even with imperfect input data. But it requires the knowledge of the clients locations in the network and all pair wise inter-node distances. In topology-informed placement strategy, servers are placed on candidate hosts in descending order of out degrees (i.e. the number of other nodes connected to a node). Here the assumption is that nodes with more out degrees can reach more nodes with smaller latency. This approach uses Autonomous Systems (AS) topologies where each node represents a single AS and node link corresponds to BGP peering. In an improved topology-informed placement strategy router-level Internet topology is used instead of AS-level topology. In this approach, each LAN associated with a router is a potential site to place a server, rather than each AS being a site. Also some other server placement algorithms like Hot Spot and Tree-based replica placement are also used in this context. The hotspot algorithm places replicas near the clients generating greatest load. It sorts the N potential sites according to the amount of traffic generated surrounding them and places replicas at the top M sites that generate maximum traffic. The tree-based replica placement algorithm is based on the assumption that the underlying topologies are trees. This algorithm models the replica placement problem as a dynamic programming problem.

programming problem. In this approach, a tree T is divided into several small trees T_i and placement of t proxies is achieved by placing t'_i proxies in the best way in each small tree T_i , where $t = \sum t'_i$. Another example is Scan [15], which is a scalable replica management framework that generates replicas on demand and organizes them into an application-level multicast tree. This approach minimizes the number of replicas while meeting clients' latency constraints and servers' capacity constraints. Figure 9 shows different surrogate server placement strategies.

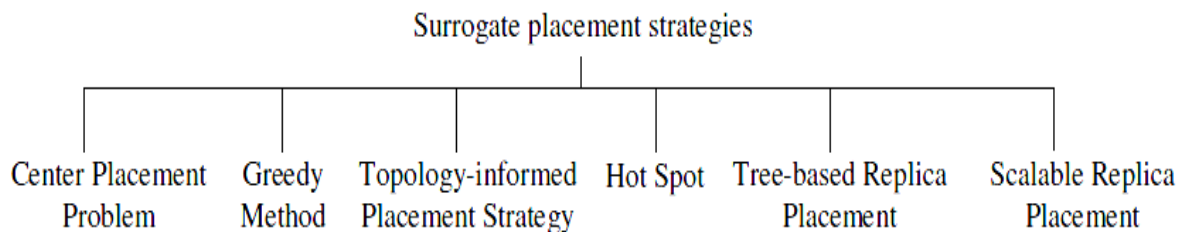


Figure 19: Surrogate placement strategies

For surrogate server placement, the CDN administrators also determine the optimal number of surrogate servers using single-ISP and multi-ISP approach. In the Single-ISP approach, a CDN provider typically deploys at least 40 surrogate servers around the network edge to support content delivery. The policy in a single-ISP approach is to put one or two surrogates in each major city within the ISP coverage. The ISP equips the surrogates with large caches. An ISP with global network can thus have extensive geographical coverage without relying on other ISPs. The drawback of this approach is that the surrogates may be placed at a distant place from the clients of the CDN provider. In Multi-ISP approach, the CDN provider places numerous surrogate servers at as many global ISP Points of Presence (POPs) as possible. It overcomes the problems with single-ISP approach and surrogates are placed close to the users and thus content is delivered reliably and timely from the requesting client's ISP. Some large CDN providers such as Akamai has more than 20000 servers. Other than the cost and complexity of setup, the main disadvantage of the multi-ISP approach is that each surrogate server receives fewer (or no) content requests which may result in idle resources and poor CDN performance. Estimation of performance of these two approaches shows that single-ISP approach works better for sites with low-to-medium traffic volumes, while the multi-ISP approach is better for high-traffic sites.

Content selection and delivery – The efficiency of content delivery lies in the right selection of content to be delivered to the end-users. An appropriate content selection approach can assist in reduction of client download time and server load. Figure 20 shows the taxonomy of content selection and delivery techniques. Content can be delivered to the customers in full or in partial.

Full-site content selection and delivery: It is a simplistic approach where the entire set of origin server's objects is outsourced to surrogate servers. In other words, in this approach, the surrogate servers perform 'entire replication' in order to deliver the total content site to the end-users. With this approach, a content provider configures its DNS in such a way that all client requests for its Web site are resolved by a CDN server, which then delivers all of the content. The main advantage of this approach is its simplicity. However, such a solution is not feasible considering the on-going increase in the size of Web objects. Although the price

of storage hardware is decreasing, sufficient storage space on the edge servers is never guaranteed to store all the content from content providers. Moreover, since the Web content is not static, the problem of updating such a huge collection of Web objects is unmanageable.

Partial site content selection and delivery: On the other hand, in partial-site content selection and delivery, surrogate servers perform ‘partial replication’ to deliver only embedded objects – such as web page images from the corresponding CDN. With partial-site content delivery, a content provider modifies its content so that links to specific objects have host names in a domain for which the CDN provider is authoritative. Thus, the base HTML page is retrieved from the origin server, while embedded objects are retrieved from CDN cache servers. A partial-site approach is better than the full-site approach in the sense that the former reduces loads on the origin server and on the site’s content generation infrastructure. Moreover, due to the infrequent change of embedded content, a partial-site approach exhibits better performance.

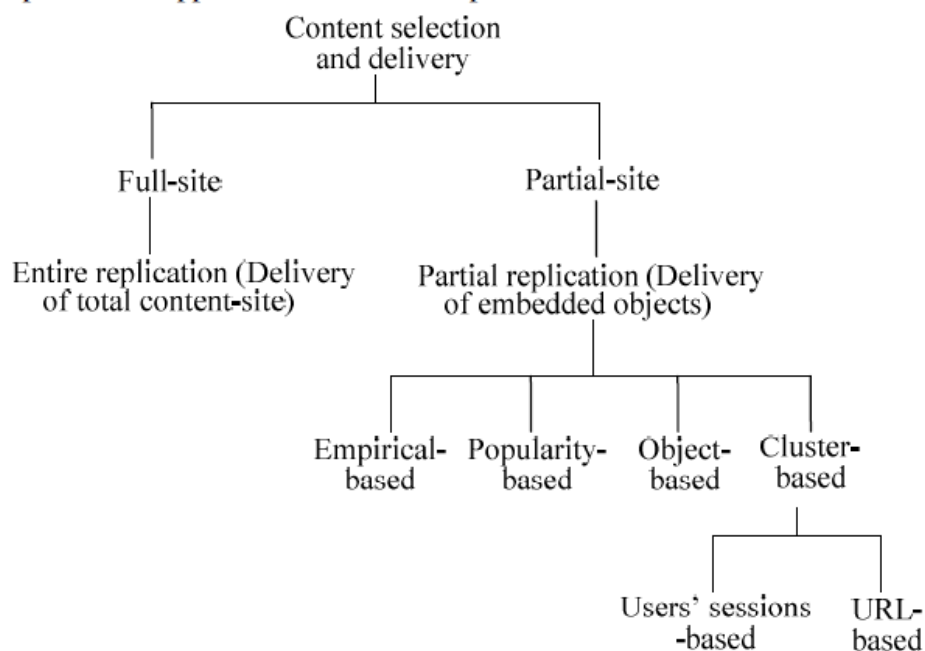


Figure 20: Taxonomy of content selection and delivery

Content selection is dependent on the suitable management strategy used for replicating Web content. Based on the approach to select embedded objects to perform replication, partial-site approach can be further divided into – empirical, popularity, object and cluster-based replication . In empirical-based approach, the Web site administrator empirically selects the content to be replicated to the edge servers. Heuristics are used in making such an empirical decision. The main drawback of this approach lies in the uncertainty in choosing the right heuristics. In popularity-based approach, the most popular objects are replicated to the surrogates. This approach is time consuming and reliable objects request statistics is not guaranteed due to the popularity of each object varies considerably . Moreover, such statistics are often not available for newly introduced content. In object-based approach, content is replicated to the surrogate servers in units of objects. This approach is greedy because each object is replicated to the surrogate server (under storage constraints) that gives the maximum

performance gain . Although such a greedy approach achieve the best performance, it suffers from high complexity to implement on real applications. In cluster-based approach, Web content is grouped based on either correlation or access frequency and is replicated in units of content clusters. The clustering procedure is performed either by fixing the number of clusters or by fixing the maximum cluster diameter, since neither the number nor the diameter of the clusters can ever be known. The content clustering can be either users' sessions-based or URL-based. In user's session-based approach, Web log files are used to cluster a set of users' navigation sessions, which show similar characteristics. This approach is beneficial because it helps to determine both the groups of users with similar browsing patterns and the groups of pages having related content. In URL-based approach, clustering of web content is done based on web site topology . The most popular objects are identified from a Web site and are replicated in units of clusters where the correlation distance between every pair of URLs is based on a certain correlation metric. Experimental results show that content replication based on such clustering approaches reduce client download time and the load on servers. But these schemes suffer from the complexity involved to deploy them.

Content outsourcing – Given a set of properly placed surrogate servers in a CDN infrastructure and a chosen content for delivery, choosing an efficient content outsourcing practice is crucial. Content outsourcing is performed using either of cooperative push-based, non-cooperative pull-based and cooperative pull-based approaches.

Cooperative push-based: This approach is based on the pre-fetching of content to the surrogates. Content is pushed to the surrogate servers from the origin, and surrogate servers cooperate to reduce replication and update cost. In this scheme, the CDN maintains a mapping between content and surrogate servers, and each request is directed to the closest surrogate server or otherwise the request is directed to the origin server. Under this approach, greedy-global heuristic algorithm is suitable for making replication decision among cooperating surrogate servers . Still it is considered as a theoretical approach since it has not been used by any CDN provider .

Non-cooperative pull-based: In this approach, client requests are directed (either using DNS redirection or URL rewriting) to their closest surrogate servers. If there is a cache miss, surrogate servers pull content from the origin server. Most popular CDN providers (e.g. Akamai, Mirror Image) use this approach. The drawback of this approach is that an optimal server is not always chosen to serve content request . Many CDNs use this approach since the cooperative push-based approach is still at the experimental stage .

Cooperative pull-based: The cooperative pull-based approach differs from the non-cooperative approach in the sense that surrogate servers cooperate with each other to get the requested content in case of cache miss. In the cooperative pull-based approach client requests are directed to the closest surrogate through DNS redirection. Using a distributed index, the surrogate servers find nearby copies of requested content and store it in the cache. The cooperative pull-based approach is reactive wherein a data object is cached only when the client requests it. An academic CDN Coral has implemented the cooperative pull-based approach using a variation of Distribution Hash Table (DHT). The optimal placement of outsourced content is another quite important content distribution issue. In the context of content outsourcing, it is crucial to determine in which surrogate servers the outsourced content should be replicated.

Cache organization – Content management is essential for CDN performance, which is mainly dependent on the cache organization approach followed by the CDN. Cache organization is in turn composed of the caching techniques used and the frequency of cache update to ensure the freshness, availability and reliability of content. Other than these two, the cache organization may also include the integration of caching policies on a CDN’s infrastructure. Such integration may be useful for a CDN for effective content management. Potential performance improvement is possible in terms of perceived latency, hit ratio and byte hit ratio if replication and caching is used together in a CDN. Moreover, the combination of caching with replication fortifies CDNs against flash crowd events.

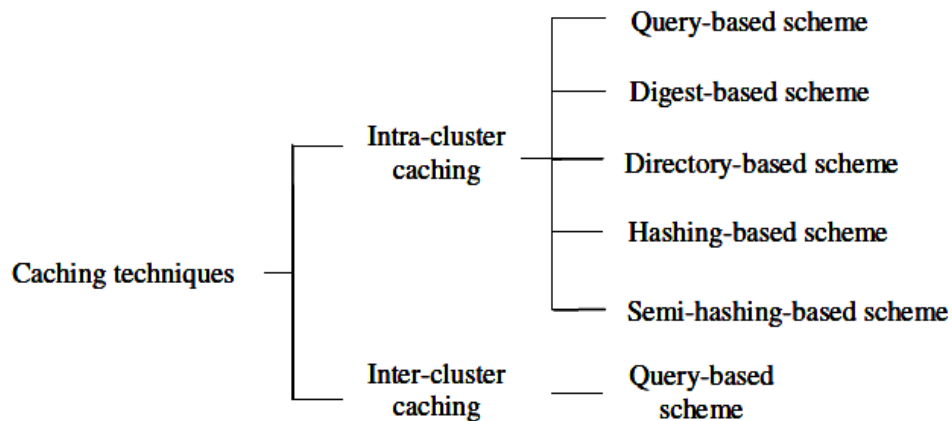


Figure 21: Caching techniques taxonomy

Caching techniques: Replicating content is common and widely accepted phenomenon in large-scale distributed environments like CDNs, where content is stored in more than one location for performance and reliability reasons. Different replication strategies are suitable for different applications. Replication in commercial CDNs is performed through caching content across the globe for high profile customers that need to deliver large volumes of data in a timely manner. Content caching in CDNs can be intra-cluster and inter-cluster basis. A taxonomy of caching techniques is shown in Figure 21.

Intra-cluster caching: For intra-cluster caching of content either of a query-based , digest-based , directory-based or hashing-based scheme can be used. In a query-based scheme, on a cache miss a CDN server broadcasts a query to other cooperating CDN servers. The problems with this scheme are the significant query traffic and the delay because a CDN server have to wait for the last ‘miss’ reply from all the cooperating surrogates before concluding that none of its peers has the requested content. Because of these drawbacks, the query-based scheme suffers from implementation overhead. The digest-based approach overcomes the problem of flooding queries in query-based scheme. In digest-based scheme, each of the CDN servers maintains a digest of content held by the other cooperating surrogates. The cooperating surrogates are informed about any sort of update of the content by the updating CDN server. On checking the content digest, a CDN server can take the decision to route a content request to a particular surrogate. The main drawback is that it suffers from update traffic overhead, because of the frequent exchange of the update traffic to make sure that the cooperating surrogates have correct information about each other. The directory-based scheme is a centralized version of the digest-based scheme. In directory-

based scheme, a centralized server keeps content information of all the cooperating surrogates inside a cluster. Each CDN server only notifies the directory server when local updates occur and queries the directory server whenever there is a local cache miss. This scheme experiences potential bottleneck and single point of failure since the directory server receives update and query traffic from all cooperating surrogates. In a hashing-based scheme the cooperating CDN servers maintain the same hashing function. A designated CDN server holds a content based on content's URL, IP addresses of the CDN servers, and the hashing function. All requests for that particular content is directed to that designated server. Hashing-based scheme is more efficient than other schemes since it has smallest implementation overhead and highest content sharing efficiency. However, it does not scale well with local requests and multimedia content delivery since the local client requests are directed to and served by other designated CDN servers. To overcome this problem, a semi-hashing-based scheme can be followed. Under the semi hashing based scheme, a local CDN server allocates a certain portion of its disk space to cache the most popular content for its local users and the remaining portion to cooperate with other CDN servers via a hashing function. Like pure hashing, semi-hashing has small implementation overhead and high content sharing efficiency. In addition, it has been found to significantly increase the local hit rate of the CDN.

Inter-cluster caching: Inter-cluster content routing is necessary when intra-cluster content routing fails. A hashing-based scheme is not appropriate for inter-cluster cooperative caching, because representative CDN servers of different clusters are normally distributed geographically. The digest-based or directory-based scheme is also not suitable for inter-cluster caching since the representative CDN servers have to maintain a huge content digest and/or directory including the content information of CDN servers in other clusters. Hence, a query-based scheme can be used for inter-cluster caching as stated in . In this approach, when a cluster fails to serve a content request, it queries other neighboring cluster(s). If the content is obtainable from this neighbor, it replies with a 'hit' message or if not, it forwards the request to other neighboring clusters. All the CDN servers inside a cluster use hashing based scheme for serving content request and the representative CDN server of a cluster only queries the designated server of that cluster to serve a content request. Hence, this scheme uses the hashing-based scheme for intra-cluster content routing and the query-based scheme for inter-cluster content routing. This approach improves performance since it limits flooding of query traffic and overcomes the problem of delays when retrieving content from remote servers through the use of a Timeout value and TTL (Time-to-Live) number with each query message.

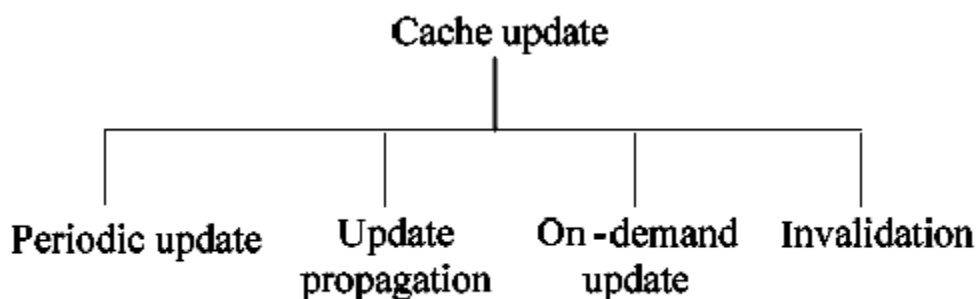


Figure 22: Cache update taxonomy

Cache update: Cached objects in the surrogate servers of a CDN have associated expiration times after which they are considered stale. Ensuring the freshness of content is necessary to serve the clients with up to date information. If there are delays involved in propagating the content, a CDN provider should be aware that the content may be inconsistent and/or expired. To manage the consistency and freshness of content at replicas, CDNs deploy different cache update techniques. The taxonomy of cache update mechanisms is shown in Figure 22.

The most common cache update method is the periodic update. To ensure content consistency and freshness, the content provider configures its origin Web servers to provide instructions to caches about what content is cacheable, how long different content is to be considered fresh, when to check back with the origin server for updated content, and so forth . With this approach, caches are updated in a regular fashion. But this approach suffers from significant levels of unnecessary traffic generated from update traffic at each interval. The update propagation is triggered with a change in content. It performs active content pushing to the CDN cache servers. In this mechanism, an updated version of a document is delivered to all caches whenever a change is made to the document at the origin server. For frequently changing content, this approach generates excess update traffic. On-demand update is a cache update mechanism where the latest copy of a document is propagated to the surrogate cache server based on prior request for that content. This approach follows a assume nothing structure and content is not updated unless it is requested. The disadvantage of this approach is the back and forth traffic between the cache and origin server in order to ensure that the delivered content is the latest. Another cache update approach is invalidation, in which an invalidation message is sent to all surrogate caches when a document is changed at the origin server. The surrogate caches are blocked from accessing the documents when it is being changed. Each cache needs to fetch an updated version of the document individually later. The drawback of this approach is that it does not make full use of the distribution network for content delivery and belated fetching of content by the caches may lead to inefficiency of managing consistency among cached contents.

Generally CDNs give the content provider control over freshness of content and ensure that all CDN sites are consistent. However, content providers themselves can build their own policies or use some heuristics to deploy organization specific caching policies. In the first case, content providers specify their caching policies in a format unique to the CDN provider, which propagates the rule sets to its caches. These rules specify instructions to the caches on how to maintain the freshness of content through ensuring consistency. In the later case, a content provider can apply some heuristics rather than developing complex caching policies. With this approach, some of the caching servers adaptively learn over time about the frequency of change of content at the origin server and tune their behavior accordingly.

3. Request-routing

A request-routing system is responsible for routing client requests to an appropriate surrogate server for the delivery of content. It consists of a collection of network elements to support request-routing for a single CDN. It directs client requests to the replica server ‘closest’ to the client. However, the closest server may not be the best surrogate server for servicing the client request . Hence, a request-routing system uses a set of metrics such as network

proximity, client perceived latency, distance, and replica server load in an attempt to direct users to the closest surrogate that can best serve the request. The content selection and delivery techniques (i.e. full-site and partial-site) used by a CDN have a direct impact on the design of its request-routing system. If the full-site approach is used by a CDN, the request-routing system assists to direct the client requests to the surrogate servers as they hold all the outsourced content. On the other hand, if the partial-site approach is used, the request-routing system is designed in such a way that on receiving the client request, the origin server delivers the basic content while surrogate servers deliver the embedded objects. The request-routing system in a CDN has two parts: deployment of a request-routing algorithm and use of a request-routing mechanism . A request-routing algorithm is invoked on receiving a client request. It specifies how to select an edge server in response to the given client request. On the other hand, a request-routing mechanism is a way to inform the client about the selection. Such a mechanism at first invokes a request-routing algorithm and then informs the client about the selection result it obtains.

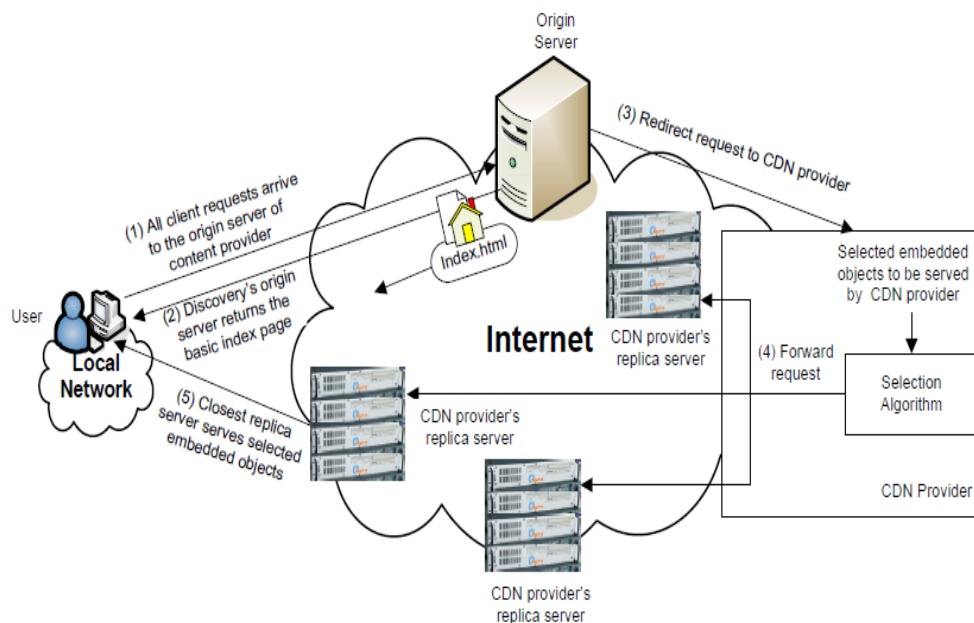


Figure 23 : Request-routing in a CDN environment

Figure 23 provides a high-level view of the request-routing in a CDN environment. The interaction flows are: (1) the client requests content from the content provider by specifying its URL in the Web browser. Client's request is directed to its origin server; (2) when origin server receives a request, it makes a decision to provide only the basic content (e.g. index page of the Web site) that can be served from its origin server; (3) to serve the high bandwidth demanding and frequently asked content (e.g. embedded objects – fresh content, navigation bar, banner ads etc.), content provider's origin server redirects client's request to the CDN provider; (4) using the proprietary selection algorithm, the CDN provider selects the replica server which is 'closest' to the client, in order to serve the requested embedded objects; (5) selected replica server gets the embedded objects from the origin server, serves the client requests and caches it for subsequent request servicing.

Request-routing algorithms

The algorithms invoked by the request-routing mechanisms can be adaptive or non-adaptive (Figure 24). Adaptive algorithms consider the current system condition to select a cache server for content delivery. Current condition of the system is obtained by estimating some metrics like load on the replica servers or the congestion of selected network links. Non-adaptive request-routing algorithms use some heuristics for selecting a cache server rather than considering the current system condition. A non-adaptive algorithm is easy to implement, while the former is more complex. Complexity of adaptive algorithms arises from their ability to change behavior to cope with an enduring situation. A non-adaptive algorithm works efficiently when the assumptions made by the heuristics are met.

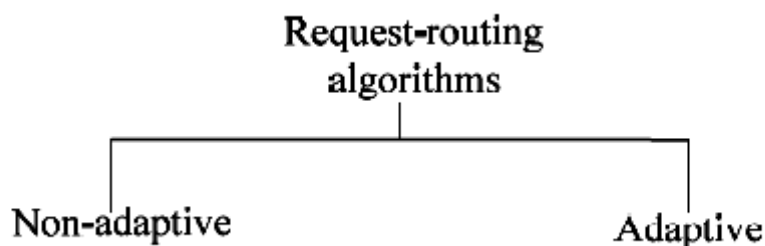


Figure 24: Taxonomy of request-routing algorithms

Non-adaptive request-routing algorithms: An example of the most common and simple non-adaptive request-routing algorithm is round-robin, which distributes all requests to the CDN cache servers and attempts to balance load among them. It is assumed that all the cache servers have similar processing capability and that any of them can serve any client request. Such simple algorithms are efficient for clusters, where all the replica servers are located at the same place. But the round-robin request-routing algorithm does not perform well for wide area distributed systems where the cache servers are located at distant places. In this case it does not consider the distance of the replica servers. Hence, client requests may be directed to more distant replica servers, which cause poor performance perceived by the users. Moreover, the aim of load balancing is not fully achieved since processing different request can involve significantly different computational costs.

In another non-adaptive request-routing algorithm, all replica servers are ranked according to the predicted load on them. Such prediction is done based on the number of requests each of the servers has served so far. This algorithm takes client-server distance into account and client requests are directed to the replica servers in such a way that load is balanced among them. The assumption here is that the replica server load and the client server distance are the most influencing factors for the efficiency of request processing. Though it has been observed in that deploying this algorithm can perform well for request-routing, the client perceived performance may still be poor.

Several other interesting non-adaptive request-routing algorithms are implemented in the Cisco Distributed Director. One of these algorithms considers the percentage of client requests that each replica server receives. A server receiving more requests is assumed to be more powerful. Hence, client requests are directed to the more powerful servers to achieve better resource utilization. Another algorithm defines preference of one server over another in

order to delegate the former to serve client requests. The Distributed Director also supports random request distribution to replica servers. Furthermore, some other non adaptive algorithms can be found which considers the client's geographic location to redirect requests to the nearby replica. But this algorithm suffers from the fact that client requests may be assigned to overloaded replica servers, which may degrade client perceived performance.

Adaptive request-routing algorithms: Globule uses an adaptive request-routing algorithm that selects the replica server closest to the clients in terms of network proximity . The metric estimation in Globule is based on path length which is updated periodically. The metric estimation service used in globule is passive, which does not introduce any additional traffic to the network. However, results in show that the distance metric estimation procedure is not very accurate.

Cisco Distributed Director has implemented an adaptive request-routing algorithm. The request routing algorithm deployed in this system takes into account a weighted combination of three metrics, namely inter-AS distance, intra-AS distance, and end-to-end latency. Though this algorithm is flexible since it makes use of three metrics, the deployment of an agent in each replica server for metric measurement makes it complex and costly. Moreover, the active latency measurement techniques used by this algorithm introduce additional traffic to the Internet. Furthermore, the isolation of Distributed Director component from the replica server makes it unable to probe the servers to obtain their load information.

Akamai uses a complex adaptive request-routing algorithm. It takes into consideration a number of metrics such as replica server load, the reliability of loads between the client and each of the replica servers, and the bandwidth that is currently available to a replica server. This algorithm is proprietary to Akamai and the technology details have not been revealed.

Request-routing mechanisms

Request-routing mechanisms inform the client about the selection of replica server, generated by the request routing algorithms. Request-routing mechanisms can be classified according to several criteria. In this section we classify them according to the variety of request processing. As shown in Figure 25, they can be classified as: Global Server Load Balancing (GSLB) , DNS-based request-routing , HTTP redirection, URL rewriting , any casting , and CDN peering .

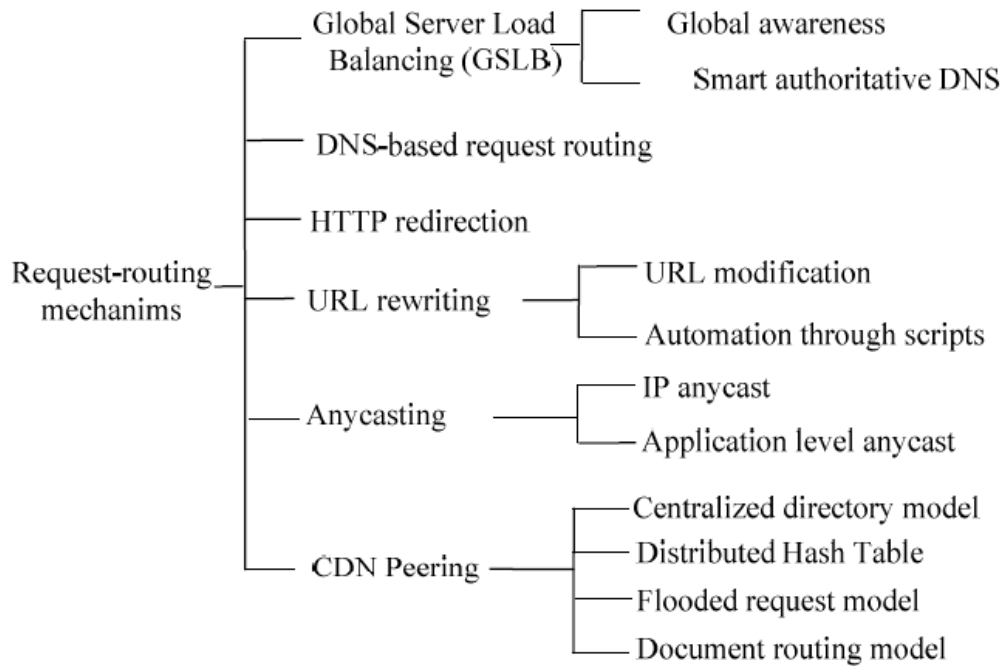


Figure 25: Taxonomy of request-routing mechanisms

These routing schemes are stated in the following:

Global Server Load Balancing (GSLB): In this approach, service nodes (which serve content to end-users) consisting of a GSLB-enabled Web switch and a number of real Web servers are distributed in several locations around the world. Two new capabilities of the service nodes allow them to support global server load balancing. The first is global awareness and the second is smart authoritative DNS . In local server load balancing each service node is aware of the health and performance information of the Web servers directly attached to it. In GSLB, one service node is aware of the information in other service nodes and includes their virtual IP address in its list of servers. Hence, the Web switches making up each service node are globally aware and each knows the addresses of all the other service nodes. They also exchange performance information among the Web switches in GSLB configuration. To make use of such global awareness, the GSLB switches act as a smart authoritative DNS for certain domains. The advantage of GSLB is that since the service nodes are aware of each other, each GSLB switch can select the best surrogate server for any request. Thus this approach facilitates choosing servers not only from the pool of locally connected real servers, but also the remote service nodes. Another significant advantage of GSLB is that the network administrator can add GSLB capability to the network without adding any additional networking devices. A disadvantage of GSLB is the manual configuration of the service nodes to enable them with GSLB capability.

DNS-based request-routing: In this approach, the content distribution services rely on the modified DNS servers to perform the mapping between a surrogate server’s symbolic name and its numerical IP address. It is used for full-site content selection and delivery. In DNS-based request-routing, a domain name has multiple IP addresses associated to it. When an end-user’s content request comes, the DNS server of the service provider returns the IP addresses of servers holding the replica of the requested object. The client’s DNS resolver

chooses a server among these. To decide, the resolver may issue probes to the servers and choose based on response times to these probes. It may also collect historical information from the clients based on previous access to these servers. Both full and partial-site CDN providers use DNS redirection. The performance and effectiveness of DNS-based request-routing has been examined in a number of recent studies . The advantage of this approach is the transparency as the services are referred to by means of their DNS names, and not their IP addresses. DNS-based approach is extremely popular because of its simplicity and independence from any actual replicated service. Since it is incorporated to the name resolution service it can be used by any Internet application . In addition, the ubiquity of DNS as a directory service provides advantages during request routing. The main disadvantage of DNS-based request-routing is that, it increases network latency because of the increase in DNS lookup times. CDN administrators typically resolve this problem by splitting CDN DNS into two levels (low-level DNS and high-level DNS) for load distribution . Another disadvantage is that it does not take into account the IP address of the clients. The knowledge of Internet location of the client DNS server limits the ability of the request-routing system to determine a client's proximity to the surrogate. Another limitation of this approach is that it is not scalable and it has limited control on client identification since a DNS query does not carry the addresses of the querying client. Most significantly, DNS cannot be relied upon to control all incoming requests due to caching of DNS data at both the ISP and client level. Indeed, it can have control over as little as 5% of requests in many instances . Furthermore, since clients do not access the actual domain names that serve their requests, it leads to the absence of any alternate server to fulfill client requests in case of failure of the target surrogate server.

HTTP redirection: The approach propagates information about replica server sets in HTTP headers. HTTP protocols allow a Web server to respond to a client request with a special message that tells the client to resubmit its request to another server. HTTP redirection can be used for both full-site and partial-site content selection and delivery. This mechanism can be used to build a special Web server, which accepts client requests, chooses replica servers for them and redirects clients to those servers. It requires changes to both Web servers and clients to process extra headers. Clients must also be modified to implement replica server selection. The main advantage of this approach is flexibility and simplicity. Another advantage is that replication can be managed at fine granularity, since individual web pages are considered as a granule . The most significant disadvantage of HTTP redirection is the lack of transparency. Moreover, the overhead perceived through this approach is significant since it introduces extra message round-trip into request processing as well as over HTTP.

URL rewriting or Navigation hyperlink: Though most CDN systems use a DNS based routing scheme, some systems use the URL rewriting. It is mainly used for partial-site content selection and delivery where embedded objects are sent as a response to client requests. In this approach, the origin server redirects the clients to different surrogate servers by rewriting the dynamically generated pages' URL links. For example, with a Web page containing an HTML file and some embedded objects, the Web server would modify references to embedded objects so that the client could fetch them from the best surrogate server. To automate this process, CDNs provide special scripts that transparently parse Web page content and replace embedded URLs . URL rewriting can be pro-active or reactive. In the pro-active URL rewriting, the URLs for embedded objects of the main HTML page are formulated before the content is loaded in the origin server. In reactive approach involves rewriting the embedded URLs of a HTML page when the client request reaches the origin server. The main advantage of URL rewriting is that the clients are not bound to a single

surrogate server, because the rewritten URLs contain DNS names that point to a group of surrogate servers. Also finer level of granularity can be achieved through this approach since embedded objects can be considered as granule. The disadvantages through this approach are the delay for URL-parsing and the possible bottleneck introduced by an in-path element. Another disadvantage is that content with modified reference to nearby surrogate server rather than to the origin server is non-cacheable.

Anycasting: The anycasting approach can be divided into two: IP anycasting and Application-level

Anycasting . IP anycasting can be suitable for request-routing and service location. It targets network-wide replication of the servers over potentially heterogeneous platforms. A disadvantage of IP anycasting is some part of IP address space is to be allocated for anycast address. Clients interact with the anycast resolvers by generating an anycast query. The resolver processes the query and replies with an anycast response. A Metric database, associated with each anycast resolver contains performance data about replica servers. The performance is estimated based on the load and the request processing capability of the servers. The overhead of the performance measurement is kept at a manageable level. The performance data can be used in the selection of a server from a group, based on user-specified performance criteria. An advantage of application level anycasting is that better flexibility can be achieved through this approach. One disadvantage of this approach is, deploying the anycasting mechanism for request-routing requires changes to the servers as well as to the clients. Hence, it may lead to increased cost considering possibly large number of servers and clients.

CDN peering: Peer-to-peer content networks are formed by symmetrical connections between host computers. Peered CDNs deliver content on each other's behalf. Thus, a CDN could expand its reach to a larger client population by using partnered CDN servers and their nearby forward proxies. A content provider usually has contracts with only one CDN and each CDN contacts other peer CDNs on the content provider's behalf . Peering CDNs are more fault-tolerant as the necessary information retrieval network can be developed on the peering members themselves instead of relying on a dedicated infrastructure like traditional CDNs. To locate content in CDN peering, a centralized directory model, Distributed Hash Table (DHT), flooded request model or document routing model can be used . In a centralized directory model, peers contact a centralized directory where all the peers publish content that they want to share with others. When the directory receives a request it responds with the information of the peer that holds the requested content. When more than one peer matches the request, the best peer is selected based on metrics such as network proximity, highest bandwidth, least congestion and highest capacity. On receiving the response from the directory, the requesting peer contacts the peer that it has been referred to for content retrieval. The drawback of this approach is that, the centralized directory is subject to a single point of failure. Moreover, the scalability of a system based on a centralized directory is limited to the capacity of the directory. Archi , WAIS are the examples of centralized directory systems for retrieving FTP files located on various systems. In systems using DHTs, peers are indexed through hashing keys within a distributed system. Then a peer holding the desired content can be found through applying complex queries . The advantage of this approach is the ability to perform load balancing by offloading excess loads to the less-loaded peers . In the flooded request model, a request from a peer is broadcast to the peers directly connected to it. These peers in turn forwards the message to other peers directly connected to them. This process continues until the request is answered or some broadcast

limit is reached. The drawback of this approach is that it generates unnecessary network traffic and hence, it requires enormous bandwidth. Thus, it suffers from scalability problem and it limits the size of the network. Gnutella is the example of a system using the flooded request model. In document routing model an authoritative peer is asked for referral to get the requested content. Each peer in the model is helpful, though they partially complete the referral information. Each referral moves to the requester closer to a peer that can satisfy the query. The main advantage of this approach is that it can complete a comprehensive search within a bounded number of steps. Moreover, it shows good performance and it is scalable enough to grow significantly large.

4. Performance measurement

Performance measurement of a CDN is done to measure its ability to serve the customers with the desired content and/or service. Customers utilizing the service need feedback on the performance of the CDN as a whole. Performance measurement offers the ability to predict, monitor and ensure the end-to-end performance of a CDN. The measurement is achieved with a combination of hardware and software-based probes distributed around the CDN, as well as using the logs from various servers within the CDN. Typically five key metrics are used by the content providers to evaluate the performance of a CDN. Those are:

- **Cache hit ratio:** It is defined as the ratio of the number of cached documents versus total documents requested. A high hit rate reflects that a CDN is using an effective cache policy to manage its caches.
- **Reserved bandwidth:** It is the measure of the bandwidth used by the origin server. It is measured in bytes and is retrieved from the origin server.
- **Latency:** It refers to the user perceived response time. Reduced latency signifies the decreases in bandwidth reserved by the origin server.
- **Surrogate server utilization:** It refers to the fraction of time during which the surrogate servers remain busy. This metric is used by the administrators to calculate CPU load, number of requests served and storage I/O usage.
- **Reliability:** Packet-loss measurements are used to determine the reliability of a CDN. High reliability indicates that a CDN incurs less packet loss and is always available to the clients.

Performance measurement can be accomplished based on internal performance measures as well as from the customer perspective. Thus, it involves the usage of internal measurement technologies and external services (e.g. MediaMetrics and Nielsen ratings). However, a CDN provider's own performance testing can be misleading, since it may perform well for a particular Web site and/or content, but perform poorly for others. To ensure reliable and efficient performance measurement, it can also be performed by independent third-party such as Keynote Systems or Giga Information Group. The performance measurement taxonomy is shown in Figure 26.

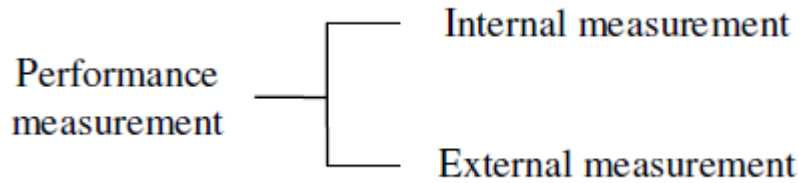


Figure 26: Performance measurement taxonomy

Internal measurement – Measurement of the content delivery services throughout the network can be performed by collecting and analyzing the logs from the caches and streaming media servers. CDN servers could be equipped with the ability to collect statistics in order to get an end-to-end measurement of its performance. In addition, deployment of probes (hardware and/or software) throughout the network and correlation of the information collected by probes with the cache and server logs can be used to measure the end-to-end performance of the CDN. Internal measurement can also be performed by using some third-party tools that can produce graphical representation of performance data.

External measurement – In addition to internal performance measurement, external measurement of performance by an independent third-party informs the CDN customers about the verified and guaranteed performance. This process is efficient since the independent performance-measuring companies support benchmarking networks of strategically located measurement computers connected through major Internet backbones in several cities. These computers measure how a particular Web site performs from the end-user's perspective, considering meaningful metrics on Web application performance in critical areas. Giga Information Group is one such evaluator that measure performance by analyzing a CDN in terms of several parameters such as scalability/availability, manageability, content type support, ease of use, cost, coverage, reporting, and viability.

Section 2

Services offered by Amazon Cloud

Chapter 8

Amazon's bouquet of Cloud Computing services

Amazon has introduced a computing platform that has changed the face of computing in the last decade. First, it installed a powerful computing infrastructure to sustain its core business, eCommerce, selling a variety of goods ranging from books and CDs to gourmet foods and home appliances. Then Amazon discovered that this infrastructure can be further extended to provide affordable and easy to use resources for enterprise computing, as well as, computing for the masses.

Amazon has a long history of using a decentralized IT infrastructure. This arrangement enabled our development teams to access compute and storage resources on demand, and it has increased overall productivity and agility. By 2005, Amazon had spent over a decade and millions of dollars building and managing the large-scale, reliable, and efficient IT infrastructure that powered one of the world's largest online retail platforms. Amazon launched Amazon Web Services (AWS) so that other organizations could benefit from Amazon's experience and investment in running a large-scale distributed, transactional IT infrastructure. AWS has been operating since 2006, and today serves hundreds of thousands of customers worldwide. Today Amazon.com runs a global web platform serving millions of customers and managing billions of dollars' worth of commerce every year.

Using AWS, you can requisition compute power, storage, and other services in minutes and have the flexibility to choose the development platform or programming model that makes the most sense for the problems they're trying to solve. You pay only for what you use, with no up-front expenses or long-term commitments, making AWS a cost-effective way to deliver applications. Here are some of examples of how organizations, from research firms to large enterprises, use AWS today:

A large enterprise quickly and economically deploys new internal applications, such as HR solutions, payroll applications, inventory management solutions, and online training to its distributed workforce. An e-commerce website accommodates sudden demand for a “hot” product caused by viral buzz from Facebook and Twitter without having to upgrade its infrastructure. A pharmaceutical research firm executes large-scale simulations using computing power provided by AWS. Media companies serve unlimited video, music, and other media to their worldwide customer base.

1. The Differences that Distinguish AWS

AWS is readily distinguished from other vendors in the traditional IT computing landscape because it is:

Flexible: AWS enables organizations to use the programming models, operating systems, databases, and architectures with which they are already familiar. In addition, this flexibility helps organizations mix and match architectures in order to serve their diverse business needs.

Cost-effective: With AWS, organizations pay only for what they use, without up-front or long-term commitments.

Scalable and elastic: Organizations can quickly add and subtract AWS resources to their applications in order to meet customer demand and manage costs.

Secure: In order to provide end-to-end security and end-to-end privacy, AWS builds services in accordance with security best practices, provides the appropriate security features in those services, and documents how to use those features.

Experienced: When using AWS, organizations can leverage Amazon’s more than fifteen years of experience delivering large-scale, global infrastructure in a reliable, secure fashion.

Flexible

The first key difference between AWS and other IT models is flexibility. Using traditional models to deliver IT solutions often requires large investments in new architectures, programming languages, and operating systems. Although these investments are valuable, the time that it takes to adapt to new technologies can also slow down your business and prevent you from quickly responding to changing markets and opportunities. When the opportunity to innovate arises, you want to be able to move quickly and not always have to support legacy infrastructure and applications or deal with protracted procurement processes.

In contrast, the flexibility of AWS allows you to keep the programming models, languages, and operating systems that you are already using or choose others that are better suited for their project. You don’t have to learn new skills. Flexibility means that migrating legacy applications to the cloud is easy and cost-effective. Instead of re-writing applications, you can easily move them to the AWS cloud and tap into advanced computing capabilities. Building applications on AWS is very much like building applications using existing hardware resources. Since AWS provides a flexible, virtual IT infrastructure, you can use the services together as a platform or separately for specific needs. AWS run almost anything—from full

web applications to batch processing to offsite data back-ups. In addition, you can move existing SOA-based solutions to the cloud by migrating discrete components of legacy applications. Typically, these components benefit from high availability and scalability, or they are self-contained applications with few internal dependencies. Larger organizations typically run in a hybrid mode where pieces of the application run in their data center and other portions run in the cloud. Once these organizations gain experience with the cloud, they begin transitioning more of their projects to the cloud, and they begin to appreciate many of the benefits outlined in this document. Ultimately, many organizations see the unique advantages of the cloud and AWS and make it a permanent part of their IT mix.

Finally, AWS provides you flexibility when provisioning new services. Instead of the weeks and months it takes to plan budget, procure, set up, deploy, operate, and hire for a new project, you can simply sign up for AWS and immediately begin deployment on the cloud the equivalent of 1, 10, 100, or 1,000 servers. Whether you want to prototype an application or host a production solution, AWS makes it simple for you to get started and be productive. Many customers find the flexibility of AWS to be a great asset in improving time to market and overall organizational productivity.

Cost-Effective

Cost is one of the most complex elements of delivering contemporary IT solutions. It seems that for every advance that will save money, there is often a commensurate investment needed to realize that savings. For example, developing and deploying an e-commerce application can be a low-cost effort, but a successful deployment can increase the need for hardware and bandwidth. Furthermore, owning and operating your own infrastructure can incur considerable costs, including power, cooling, real estate, and staff. In contrast, the cloud provides an on-demand IT infrastructure that lets you consume only the amount of resources that you actually need. You are not limited to a set amount of storage, bandwidth, or computing resources. It is often difficult to predict requirements for these resources. As a result, you might provision too few resources, which has an impact on customer satisfaction, or you might provide too many resources and miss an opportunity to maximize return on investment (ROI) through full utilization. The cloud provides the flexibility to strike the right balance. AWS requires no up-front investment, long-term commitment, or minimum spend. You can get started through a completely self-service experience online, scale up and down as needed, and terminate your relationship with AWS at any time. You can access new resources almost instantly. The ability to respond quickly to changes, no matter how large or small, means that you can take on new opportunities and meet business challenges that could drive revenue and reduce costs.

Scalable and Elastic

In the traditional IT organization, scalability and elasticity were often equated with investment and infrastructure. In the cloud, scalability and elasticity provide opportunity for savings and improved ROI. AWS uses the term elastic to describe the ability to scale computing resources up and down easily, with minimal friction. Elasticity helps you avoid provisioning resources up front for projects with variable consumption rates or short lifetimes. Instead of acquiring hardware, setting it up, and maintaining it in order to allocate resources to your applications, you use AWS to allocate resources using simple API calls. Imagine what would happen to a traditional IT shop if traffic to an application doubled or tripled in a short period. For example, during benefits open enrolment periods, many corporate users generate

significant traffic to internal applications. You need to be confident that your existing infrastructure can handle a spike in traffic, and that the spike will not interfere with normal business operations. Elastic Load Balancing and Auto Scaling can automatically scale your AWS cloud-based resources up to meet unexpected demand, and then scale those resources down as demand decreases.

The AWS cloud is also a useful resource for implementing short-term jobs, mission-critical jobs, and jobs repeated at regular intervals. For example, when a pharmaceutical company needs to run drug simulations (a short-term job), it can use AWS to spin up resources in the cloud, and then shut them down when it no longer needs additional resources. When an enterprise has to quickly deal with the effects of natural disaster on its data center (a mission-critical job), it can use AWS to tap into new storage and computing resources to accommodate demand. Furthermore, AWS can preserve computing resources and reduce costs for regularly repeated tasks, such as month-end payroll or invoice processing.

Secure

AWS delivers a scalable cloud-computing platform that provides customers with end-to-end security and end-to-end privacy. AWS builds security into its services in accordance with security best practices, and documents how to use the security features. It is important that you leverage AWS security features and best practices to design an appropriately secure application environment. Ensuring the confidentiality, integrity, and availability of your data is of the utmost importance to AWS, as is maintaining your trust and confidence. AWS takes the following approaches to secure the cloud infrastructure:

Certifications and accreditations:

AWS has in the past successfully completed multiple SAS70 Type II audits, and now publishes a Service Organization Controls 1 (SOC 1) report, published under both the SSAE 16 and the ISAE 3402 professional standards. In addition to the SOC 1 report, AWS publishes a Service Organization Controls 2 (SOC 2), Type II report. Similar to the SOC 1 in the evaluation of controls, the SOC 2 report is an attestation report that expands the evaluation of controls to the criteria set forth by the American Institute of Certified Public Accountants (AICPA) Trust Services Principles. Additionally, AWS publishes a Service Organization Controls 3 (SOC 3) report. The SOC 3 report is a publically-available summary of the AWS SOC 2 report and provides the AICPA SysTrust Security Seal. The report includes the external auditor's opinion of the operation of controls (based on the AICPA's Security Trust Principles included in the SOC 2 report), the assertion from AWS management regarding the effectiveness of controls, and an overview of AWS Infrastructure and Services. In addition, AWS has achieved ISO 27001 certification, and has been successfully validated as a Level 1 service provider under the Payment Card Industry (PCI) Data Security Standard (DSS). In the realm of public sector certifications, AWS has achieved Agency Authority to Operate (ATOs) under the Federal Risk and Authorization Management Program (FedRAMP) at the Moderate impact level for AWS GovCloud (US) and all US regions. The AWS ATOs are the result of a comprehensive, independent assessment of the FedRAMP control requirements. The authorization package can be leveraged by all federal, state, and local governments. AWS enables US government agencies to achieve and sustain compliance with the Federal Information Security Management Act (FISMA). The AWS infrastructure has been evaluated by independent assessors for a variety of government systems as part of their

system owners' approval process. Numerous Federal Civilian and Department of Defense (DoD) organizations have successfully achieved security authorizations for systems hosted on AWS in accordance with the Risk Management Framework (RMF) process defined in NIST 800-37 and DoD Information Assurance Certification and Accreditation Process (DIACAP). AWS's secure infrastructure has helped federal agencies expand cloud computing use cases and deploy sensitive government data and applications in the cloud while complying with the rigorous security requirements of federal standards. We will continue to obtain the appropriate security certifications and conduct audits to demonstrate the security of our infrastructure and services. The AWS GovCloud (US) region supports US International Traffic in Arms Regulations (ITAR) compliance. As a part of managing a comprehensive ITAR compliance program, companies subject to ITAR export regulations must control unintended exports by restricting access to protected data to US Persons and restricting physical location of that data to the US. AWS GovCloud (US) provides an environment physically located in the US and where access by AWS Personnel is limited to US Persons, thereby allowing qualified companies to transmit, process, and store protected articles and data subject to ITAR restrictions. The AWS GovCloud (US) environment has been audited by an independent third-party to validate the proper controls are in place to support customer export compliance programs for this requirement. AWS will continue to obtain the appropriate security certifications and accreditations to demonstrate the security of our infrastructure and services.

Physical security:

Amazon has many years of experience designing, constructing, and operating large-scale data centers. The AWS infrastructure is located in Amazon-controlled data centers throughout the world. Knowledge of the location of the data centers is limited to those within Amazon who have a legitimate business reasons for this information. The data centers are physically secured in a variety of ways to prevent unauthorized access.

Secure services:

Each service in the AWS cloud is architected to be secure. The services contain a number of capabilities that restrict unauthorized access or usage without sacrificing the flexibility that customers demand.

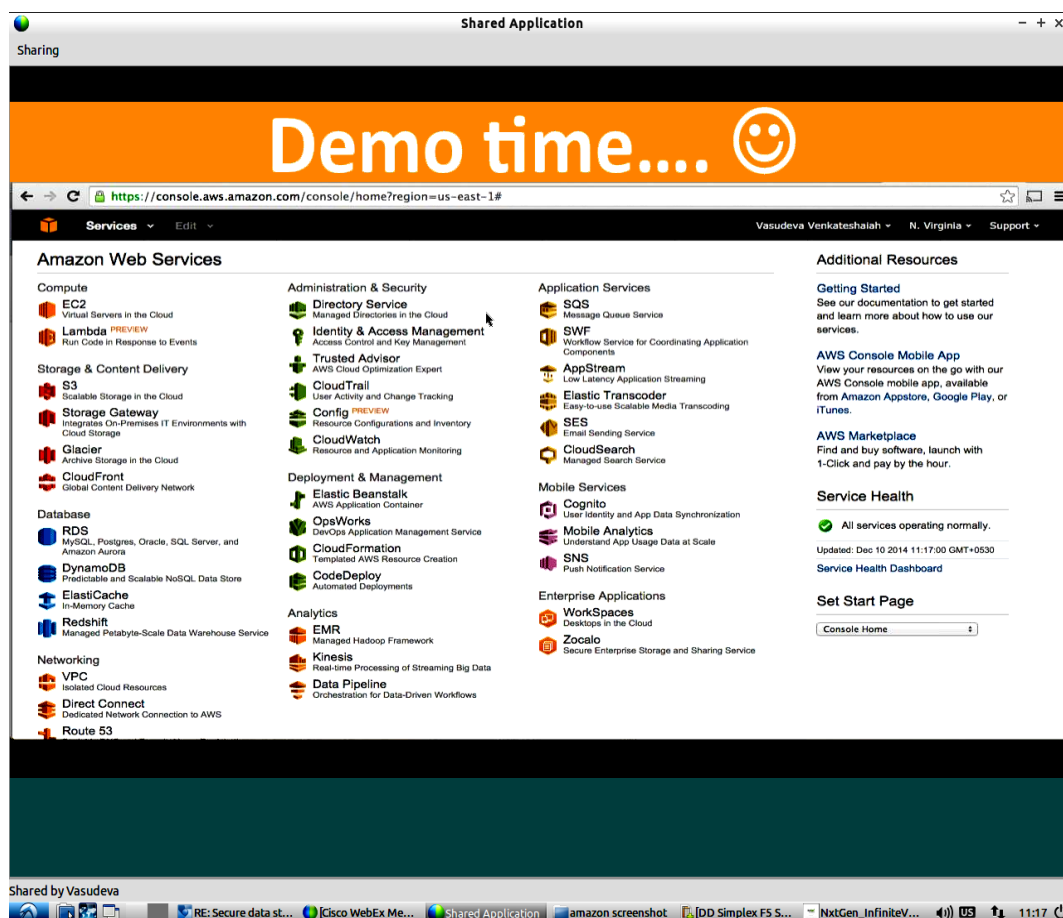
Data privacy:

You can encrypt personal and business data in the AWS cloud, and publish backup and redundancy procedures for services so that your customers can protect their data and keep their applications running.

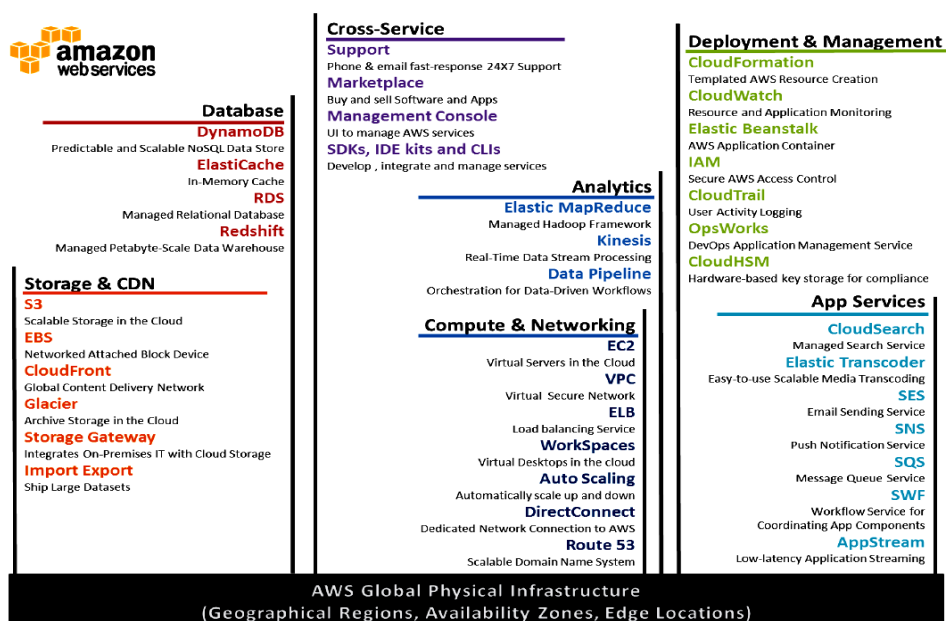
Chapter 9

An Overview of Amazon's Web Services in Cloud

AWS consists of many cloud services that you can use in combinations tailored to your business or organizational needs. This section introduces the AWS services in the following categories: compute, networking, storage and content delivery, databases, analytics, application services, deployment and management, mobile and applications.



1. Amazon Compute Service



Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers and system administrators. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances (called Amazon EC2 instances) to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers and system administrators the tools to build failure resilient applications and isolate themselves from common failure scenarios.

Benefits of EC2

❖ Elastic Web-Scale Computing

Amazon EC2 enables you to increase or decrease capacity within minutes, not hours or days. You can commission one, hundreds or even thousands of server instances simultaneously. Of course, because this is all controlled with web service APIs, your application can automatically scale itself up and down depending on its needs.

❖ Completely Controlled

You have complete control of your Amazon EC2 instances. You have root access to each one, and you can interact with them as you would any machine. You can stop your Amazon EC2 instance while retaining the data on your boot partition, and then subsequently restart the

same instance using web service APIs. Instances can be rebooted remotely using web service APIs. In addition, you can use the AWS Management Console, a simple, web-based user interface, to access and manage your Amazon EC2 instances.

❖ **Flexible Cloud Hosting Services**

You can choose among multiple Amazon EC2 instance types, operating systems, and software packages. Amazon EC2 allows you to select a configuration of memory, CPU, instance storage, and the boot partition size that is optimal for your choice of operating system and application. For example, your choice of operating systems includes numerous Linux distributions and Microsoft Windows Server.

❖ **Designed for use with other Amazon Web Services**

Amazon EC2 works in conjunction with Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB, and Amazon Simple Queue Service (Amazon SQS) to provide a complete solution for computing, query processing, and storage across a wide range of applications.

❖ **Reliable**

Amazon EC2 offers a highly reliable environment where replacement instances can be rapidly and predictably commissioned. The service runs within Amazon's proven network infrastructure and data centers. The Amazon EC2 Service Level Agreement commitment is 99.95% availability for each Amazon EC2 region.

❖ **Secure**

Amazon EC2 works in conjunction with Amazon Virtual Private Cloud (Amazon VPC) to provide security and robust networking functionality for your computer resources. Your compute instances are located in an Amazon Virtual Private Cloud with an IP address range that you specify. You decide which instances are exposed to the Internet and which remain private. Security groups and network ACLs allow you to control inbound and outbound network access to and from your instances. You can connect your existing IT infrastructure to resources in your Amazon VPC using industry-standard encrypted IPsec VPN connections. You can provision your Amazon EC2 resources as Dedicated Instances. Dedicated Instances are Amazon EC2 instances that run on hardware dedicated to a single customer for additional isolation.

❖ **Inexpensive**

Amazon EC2 passes on to you the financial benefits of Amazon's scale. You pay a very low rate for the compute capacity you actually consume with no long-term commitments. This frees you from the costs and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into much smaller variable costs. On-Demand Instances also remove the need to buy "safety net" capacity to handle periodic traffic spikes.

❖ **Reserved Instances**

Reserved Instances give you the option to make a low, one-time payment for each instance you want to reserve and in turn receive a significant discount on the hourly charge for that instance. There are three Reserved Instance types (Light, Medium, and Heavy Utilization Reserved Instances) that enable you to balance the amount you pay upfront with your effective hourly price. The Reserved Instance Marketplace is also available, which provides you with the opportunity to sell Reserved Instances if your needs change. For example, you

might want to move instances to a new AWS region, change to a new instance type, or sell capacity for projects that end before your Reserved Instance term expires.

❖ **Spot Instances**

Spot Instances allow you to bid on unused Amazon EC2 capacity and run those instances for as long as your bid exceeds the current Spot Price. The Spot Price changes periodically based on supply and demand, and customers whose bids meet or exceed it gain access to the available Spot Instances. If you can be flexible about when your applications need to run, Spot Instances can significantly lower your Amazon EC2 costs.

❖ **Auto Scaling**

Auto Scaling allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define. With the Auto Scaling service, you can ensure that the number of Amazon EC2 instances you're using increases seamlessly during demand spikes to maintain performance, and decreases automatically during demand lulls to minimize costs. Auto Scaling is particularly well suited for applications that experience hourly, daily, or weekly variability in usage. Features and Benefits

Maintain your Amazon EC2 instance availability

Whether you are running one Amazon EC2 instance or thousands, you can use Auto Scaling to detect impaired Amazon EC2 instances and unhealthy applications, and replace the instances without your intervention. This ensures that your application is getting the compute capacity that you expect.

Automatically Scale Your Amazon EC2 Fleet

Auto Scaling enables you to follow the demand curve for your applications closely, reducing the need to manually provision Amazon EC2 capacity in advance. For example, you can set a condition to add new Amazon EC2 instances in increments to the Auto Scaling group when the average utilization of your Amazon EC2 fleet is high; and similarly, you can set a condition to remove instances in the same increments when CPU utilization is low. If you have predictable load changes, you can set a schedule through Auto Scaling to plan your scaling activities. You can use Amazon Cloud Watch to send alarms to trigger scaling activities and Elastic Load Balancing to help distribute traffic to your instances within Auto Scaling groups. Auto Scaling enables you to run your Amazon EC2 fleet at optimal utilization.

❖ **Elastic Load Balancing**

Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic. Elastic Load Balancing detects unhealthy instances and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored. Customers can enable Elastic Load Balancing within a single Availability Zone or across multiple zones for even more consistent application performance.

2. Use Cases of Amazon Services

Achieving Even Better Fault Tolerance for Your Applications

You can build fault tolerant applications by placing your Amazon EC2 instances in multiple Availability Zones. To achieve even more fault tolerance with less manual intervention, you can use Elastic Load Balancing. You get improved fault tolerance by placing your compute instances behind an Elastic Load Balancer, as it can automatically balance traffic across multiple instances and multiple Availability Zones and ensure that only healthy Amazon EC2 instances receive traffic. You can setup an Elastic Load Balancer to load balance incoming application traffic across Amazon EC2 instances in a single Availability Zone or multiple Availability Zones. Elastic Load Balancing can detect the health of Amazon EC2 instances. When it detects unhealthy Amazon EC2 instances, it no longer routes traffic to those unhealthy Amazon EC2 instances. Instead, it spreads the load across the remaining healthy Amazon EC2 instances. If all of your Amazon EC2 instances in a particular Availability Zone are unhealthy, but you have set up Amazon EC2 instances in multiple Availability Zones, Elastic Load Balancing will route traffic to your healthy Amazon EC2 instances in those other zones. It will resume load balancing to the original Amazon EC2 instances when they have been restored to a healthy state.

DNS Failover for Elastic Load Balancing

You can use Amazon Route 53 health checking and DNS failover features to enhance the availability of the applications running behind Elastic Load Balancers. Route 53 will fail away from a load balancer if there are no healthy EC2 instances registered with the load balancer or if the load balancer itself is unhealthy. Using Route 53 DNS failover, you can run applications in multiple AWS regions and designate alternate load balancers for failover across regions. In the event that your application is unresponsive, Route 53 will remove the unavailable load balancer endpoint from service and direct traffic to an alternate load balancer in another region.

Auto Scaling with Elastic Load Balancing

Let's say that you want to make sure that the number of healthy Amazon EC2 instances behind an Elastic Load Balancer is never fewer than two. You can use Auto Scaling to set these conditions, and when Auto Scaling detects that a condition has been met, it automatically adds the requisite amount of Amazon EC2 instances to your Auto Scaling Group. Or, if you want to make sure that you add Amazon EC2 instances when latency of any one of your Amazon EC2 instances exceeds 4 seconds over any 15 minute period, you can set that condition, and Auto Scaling will take the appropriate action on your Amazon EC2 instances — even when running behind an Elastic Load Balancer. Auto Scaling works equally well for scaling Amazon EC2 instances whether you're using Elastic Load Balancing or not.

Using Elastic Load Balancing in your Amazon VPC

Elastic Load Balancing makes it easy to create an internet-facing entry point into your VPC or to balance load between tiers of your application within your VPC. You can assign security groups to your ELB to control which ports are open to a list of allowed sources. Because Elastic Load Balancing is attached to your VPC, all of your existing Network Access Control Lists (ACL's) and Routing Tables continue to provide additional network controls. When you create a load balancer in your VPC, you can specify whether the load balancer is internet-facing (the default) or internal. If you select internal, you do not need to have an

internet gateway to reach the load balancer, and the private IP addresses of the load balancer will be used in the load balancer's DNS record.

3. AWS Lambda

AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you, making it easy to build applications that respond quickly to new information. AWS Lambda starts running your code within milliseconds of an event such as an image upload, in-app activity, website click, or output from a connected device. You can also use AWS Lambda to create new back-end services where compute resources are automatically triggered based on custom requests. With AWS Lambda you pay only for the requests served and the compute time required to run your code.

4. Amazon EC2 Container Service

Amazon EC2 Container Service is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run distributed applications on a managed cluster of Amazon EC2 instances. Amazon EC2 Container Service lets you launch and stop container-enabled applications with simple API calls, allows you to query the state of your cluster from a centralized service, and gives you access to many familiar Amazon EC2 features like security groups, Amazon EBS volumes and IAM roles. You can use Amazon EC2 Container Service to schedule the placement of containers across your cluster based on your resource needs, isolation policies, and availability requirements. Amazon EC2 Container Service eliminates the need for you to operate your own cluster management and configuration management systems or worry about scaling your management infrastructure.

5. VM Import/Export

VM Import/Export enables you to easily import virtual machine images from your existing environment to Amazon EC2 instances and export them back to your on-premises environment. This offering allows you to leverage your existing investments in the virtual machines that you have built to meet your IT security, configuration management, and compliance requirements by bringing those virtual machines into Amazon EC2 as ready-to-use instances. You can also export imported instances back to your on-premises virtualization infrastructure, allowing you to deploy workloads across your IT infrastructure.

To import your images, use the Amazon EC2 API tools, or if you use the VMware vSphere virtualization platform, the Amazon EC2 VM Import Connector to target a virtual machine (VM) image in your existing environment. You then specify which Availability Zone and instance type you want to run in Amazon EC2, and VM Import/Export will automatically transfer the image file and create your instance. Once you have imported your VMs, you can take advantage of Amazon's elasticity, scalability and monitoring via offerings like Auto Scaling, Elastic Load Balancing and Cloud Watch to support your imported images. Your instance will be up and running in Amazon EC2 in as little time as it takes to upload your image.

You can export previously imported EC2 instances using the Amazon EC2 API tools. You

simply specify the target instance, virtual machine file format and a destination Amazon S3 bucket, and VM Import/Export will automatically export the instance to the Amazon S3 bucket. You can then download and launch the exported VM within your on premises virtualization infrastructure.

Using the Import/Export Tools

VM Import/Export offers several ways to import your virtual machine into Amazon EC2. The first method is to import your VM image using the AWS CLI tools. To get started, simply Download and install the AWS Command Line Interface. Verify that your VM satisfies the prerequisites for VM Import, prepare it for import, and export it from its current environment as an OVA file (or VMDK, VHD, or RAW). Upload the VM image to S3 using the AWS CLI. Multi-part uploads will provide improved performance. As an alternative, you can also send the VM image to AWS using the AWS Import service. Once the VM image is uploaded, import your VM using the `ec2 import-image` command. As part of this command, you can specify the licensing model and other parameters for your imported image. Use the `ec2 describe-import-image-tasks` command to monitor the import progress. Once your import task is completed, you can use the `ec2 run-instances` command to create an Amazon EC2 instance from the AMI generated during the import process. You can export EC2 instances you previously imported using the Amazon EC2 CLI tools:

Download and install the EC2 Command Line Interface.

- ✓ Export the instance using the `ec2-create-instance-export-task` command. The export command captures the parameters necessary (instance ID, S3 bucket to hold the exported image, name of the exported image, VMDK, OVA or VHD format) to properly export the instance to your chosen format. The exported file is saved in an S3 bucket that you previously created.
- ✓ Use `ec2-describe-export-tasks` to monitor the export progress.
- ✓ Use `ec2-cancel-export-task` to cancel an export task prior to completion.
- ✓ Alternatively, if you use the VMware vSphere virtualization platform, you can use the AWS Management Portal for vCenter, which provides you a simple graphical user interface to import your virtual machines.

6. Licensing Model

In general, when you import your Microsoft Windows VM images into Amazon EC2, AWS will provide the appropriate Microsoft Windows Server license key for your imported instance. Hourly EC2 instance charges cover the Microsoft Windows Server software and underlying hardware resources. Your on-premise Microsoft Windows Server license key will not be used by EC2 and you are free to reuse it for other Microsoft Windows VM images within your on-premise environment. You are responsible for complying with the terms of your agreement(s) with Microsoft.

If you export an Amazon EC2 instance, access to the Microsoft Windows Server license key for that instance is no longer available through AWS. You will need to reactivate and specify a new license key for the exported VM image after it is launched in your on-premise

virtualization platform.

When you import Red Hat Enterprise Linux (RHEL) VM images, you can use license portability for your RHEL instances. With license portability, you are responsible for maintaining the RHEL licenses for imported instances, which you can do using Red Hat Cloud Access. More information about Cloud Access subscriptions for Red Hat Enterprise Linux is available from Red Hat. Please contact Red Hat to verify your eligibility.

7. Common Uses for VM Import/Export

Migrate Your Existing Applications and Workloads to Amazon EC2

Migrate your existing VM-based applications and workloads to Amazon EC2. Using VM Import, you can preserve the software and settings that you have configured in your existing VMs, while benefiting from running your applications and workloads in Amazon EC2. Once your applications and workloads have been imported, you can run multiple instances from the same image, and you can create Snapshots to backup your data. You can use AMI and Snapshot copy to replicate your applications and workloads around the world. You can change the instance types that your applications and workloads use as their resource requirements change. You can use Cloud Watch to monitor your applications and workloads after you have imported them. And you can take advantage of Auto Scaling, Elastic Load Balancing, and all of the other Amazon Web Services to support your applications and workloads after you have migrated them to Amazon EC2.

Copy Your VM Image Catalogue to Amazon EC2

Copy your existing VM image catalogue to Amazon EC2. If you use a catalogue of approved VM images, a common practice in Enterprise computing environments, VM Import enables you to copy your image catalogue to Amazon EC2, which will create Amazon EC2 AMIs from your VMs, which will serve as your image catalogue within Amazon EC2. Your existing software, including products that you have installed like anti-virus software, intrusion detection systems, and more, can all be imported along with your VM images.

Create a Disaster Recovery Repository for your VM images

Import your on-premises VM images to Amazon EC2 for backup and disaster recovery contingencies. VM Import will store the imported images as Elastic Block Store-backed AMIs so they're ready to launch in Amazon EC2 when you need them. In the event of a contingency, you can quickly launch your instances to preserve business continuity while simultaneously exporting them to rebuild your on-premises infrastructure. You only pay for Elastic Block Store charges until you decide to launch the instances. Once launched, you pay normal Amazon EC2 service charges for your running instances. If you choose to export your instances, you will pay normal S3 storage charges.

Chapter 10

Understanding Amazon's Network Service

1. Amazon VPC

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. You can easily customize the network configuration for your Amazon VPC. For example, you can create a public-facing subnet for your web servers that have access to the Internet, and place your backend systems such as databases or application servers in a private-facing subnet with no Internet access. You can leverage multiple layers of security (including security groups and network access control lists) to help control access to Amazon EC2 instances in each subnet. Additionally, you can create a hardware virtual private network (VPN) connection between your corporate data center and your Amazon VPC and leverage the AWS cloud as an extension of your corporate data center.

Features and Benefits

Multiple Connectivity Options, A variety of connectivity options exist for your Amazon Virtual Private Cloud. You can connect your VPC to the Internet, to your data center, or other VPC's based on the AWS resources that you want to expose publicly and those that you want to keep private. Connect directly to the Internet (public subnets)– You can launch instances into a publicly accessible subnet where they can send and receive traffic from the Internet.

Connect to the Internet using Network Address Translation (private subnets)– Private subnets can be used for instances that you do not want to be directly addressable from the Internet. Instances in a private subnet can access the Internet without exposing their private IP address by routing their traffic through a Network Address Translation (NAT) instance in a public subnet. Connect securely to your corporate datacenter– All traffic to and from instances in your VPC can be routed to your corporate datacenter over an industry standard, encrypted IPsec hardware VPN connection. Connect privately to other VPCs- Peer VPCs together to share resources across multiple virtual networks owned by your or other AWS accounts.

Connect to Amazon S3 without using an internet gateway or NAT, and control what buckets, requests, users, or groups are allowed through a VPC Endpoint for S3. Combine connectivity methods to match the needs of your application– You can connect your VPC to both the Internet and your corporate data center and configure Amazon VPC route tables to direct all traffic to its proper destination.

Secure Amazon VPC provides advanced security features such as security groups and network access control lists to enable inbound and outbound filtering at the instance level and subnet level. In addition, you can store data in Amazon S3 and restrict access so that it's only accessible from instances in your VPC. Optionally, you can also choose to launch Dedicated Instances which run on hardware dedicated to a single customer for additional isolation..You can create a VPC quickly and easily using the AWS Management Console. You can select one of the common network setups that best match your needs and press "Start VPC Wizard." Subnets, IP ranges, route tables, and security groups are automatically created for you, so you can concentrate on creating the applications to run in your VPC.

All the Scalability and Reliability of AWS

Amazon VPC provides all the same benefits as the rest of the AWS platform. You can instantly scale your resources up or down, select Amazon EC2 instances types and sizes that are right for your applications, and pay only for the resources you use - all within Amazon's proven infrastructure.

Use Cases of Amazon VPC

Host a simple, public-facing website .You can host a basic web application, such as a blog or simple website in a VPC, and gain the additional layers of privacy and security afforded by Amazon VPC. You can help secure the website by creating security group rules which allow the webserver to respond to inbound HTTP and SSL requests from the Internet while simultaneously prohibiting the webserver from initiating outbound connections to the Internet. You can create a VPC that supports this use case by selecting "VPC with a Single Public Subnet Only" from the Amazon VPC console wizard.

Host multi-tier web applications You can use Amazon VPC to host multi-tier web applications and strictly enforce access and security restrictions between your web servers, application servers, and databases. You can launch web servers in a publicly accessible subnet and application servers and databases in non-publically accessible subnets. The application servers and databases can't be directly accessed from the Internet, but they can still access the Internet via a NAT instance to download patches, for example. You can control access between the servers and subnets using inbound and outbound packet filtering provided by

network access control lists and security groups. To create a VPC that supports this use case, you can select "VPC with Public and Private Subnets" in the Amazon VPC console wizard.

Host scalable web applications in the AWS cloud that are connected to your datacenter You can create a VPC where instances in one subnet, such as web servers, communicate with the Internet while instances in another subnet, such as application servers, communicate with databases on your corporate network. An IPsec VPN connection between your VPC and your corporate network helps secure all communication between the application servers in the cloud and databases in your datacenter. Web servers and application servers in your VPC can leverage Amazon EC2 elasticity and Auto Scaling features to grow and shrink as needed. You can create a VPC to support this use case by selecting "VPC with Public and Private Subnets and Hardware VPN Access" in the Amazon VPC console wizard.

Extend your corporate network into the cloud

You can move corporate applications to the cloud, launch additional web servers, or add more compute capacity to your network by connecting your VPC to your corporate network. Because your VPC can be hosted behind your corporate firewall, you can seamlessly move your IT resources into the cloud without changing how your users access these applications. You can select "VPC with a Private Subnet Only and Hardware VPN Access" from the Amazon VPC console wizard to create a VPC that supports this use case.

Disaster Recovery

You can periodically backup your mission critical data from your datacenter to a small number of Amazon EC2 instances with Amazon Elastic Block Store (EBS) volumes, or import your virtual machine images to Amazon EC2. In the event of a disaster in your own datacenter, you can quickly launch replacement compute capacity in AWS to ensure business continuity. When the disaster is over, you can send your mission critical data back to your datacenter and terminate the Amazon EC2 instances that you no longer need. By using Amazon VPC for disaster recovery, you can have all the benefits of a disaster recovery site at a fraction of the normal cost.

2. AWS Direct Connect

AWS Direct Connect makes it easy to establish a dedicated network connection from your premises to AWS. Using AWS Direct Connect, you can establish private connectivity between AWS and your data center, office, or co-location environment, which in many cases can reduce your network costs, increase bandwidth throughput, and provide a more consistent network experience than Internet-based connections.

AWS Direct Connect lets you establish a dedicated network connection between your network and one of the AWS Direct Connect locations. Using industry standard 802.1Q virtual LANS (VLANs), this dedicated connection can be partitioned into multiple virtual interfaces. This allows you to use the same connection to access public resources, such as objects stored in Amazon S3, using public IP address space, and private resources, such as Amazon EC2 instances running within an Amazon VPC, using private IP address space, while maintaining network separation between the public and private environments. Logical connections can be reconfigured at any time to meet your changing needs.

Service Highlights

Reduces Your Bandwidth Costs

If you have bandwidth-heavy workloads that you wish to run in AWS, AWS Direct Connect reduces your network costs into and out of AWS in two ways. First, by transferring data to and from AWS directly, you can reduce your bandwidth commitment to your Internet service provider. Second, all data transferred over your dedicated connection is charged at the reduced AWS Direct Connect data transfer rate rather than Internet data transfer rates.

Consistent Network Performance

Network latency over the Internet can vary given that the Internet is constantly changing how data gets from point A to B. With AWS Direct Connect, you choose the data that utilizes the dedicated connection and how that data is routed which can provide a more consistent network experience over Internet-based connections.

Compatible with all AWS Services

AWS Direct Connect is a network service, and works with all AWS services that are accessible over the Internet, such as Amazon Simple Storage Service (Amazon S3), Elastic Compute Cloud (Amazon EC2), and Amazon Virtual Private Cloud (Amazon VPC).

Private Connectivity to your Amazon VPC

You can use AWS Direct Connect to establish a private virtual interface from your on-premise network directly to your Amazon VPC, providing you with a private, high bandwidth network connection between your network and your VPC. With multiple virtual interfaces, you can even establish private connectivity to multiple VPCs while maintaining network isolation.

Elastic

AWS Direct Connect makes it easy to scale your connection to meet your needs. AWS Direct Connect provides 1 Gbps and 10 Gbps connections, and you can easily provision multiple connections if you need more capacity. You can also use AWS Direct Connect instead of establishing a VPN connection over the Internet to your Amazon VPC, avoiding the need to utilize VPN hardware that frequently can't support data transfer rates above 4 Gbps. You can sign up for AWS Direct Connect service quickly and easily using the AWS Management Console. The console provides a single view to efficiently manage all your connections and virtual interfaces. You can also download customized router templates for your networking equipment after configuring one or more virtual interfaces.

Amazon Route 53

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. It is designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet applications by translating human readable names, such as `www.example.com`, into the numeric IP addresses, such as `192.0.2.1`, that computers use to connect to each other.

Amazon Route 53 effectively connects user requests to infrastructure running in AWS—such as Amazon EC2 instances, Elastic Load Balancing load balancers, or Amazon S3 buckets—

and can also be used to route users to infrastructure outside of AWS. You can use Amazon Route 53 to configure DNS health checks to route traffic to healthy endpoints or to independently monitor the health of your application and its endpoints. Amazon Route 53 makes it possible for you to manage traffic globally through a variety of routing types, including Latency Based Routing, Geo DNS, and Weighted Round Robin—all of which can be combined with DNS Failover in order to enable a variety of low-latency, fault-tolerant architectures. Amazon Route 53 also offers Domain Name Registration—you can purchase and manage domain names such as example.com and Amazon Route 53 will automatically configure DNS settings for your domains.

Chapter 11:

Advantages offered by Amazon Storage Services

1. Amazon S3

Amazon Simple Storage Service (Amazon S3) provides developers and IT teams with safe, secure, highly-scalable object storage. Amazon S3 provides a simple web-services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. Amazon S3 can be used alone or together with Amazon EC2/EBS, Amazon Glacier, and third-party storage repositories and gateways to provide cost-effective object storage for a wide variety of use cases including cloud applications, content distribution, backup and archiving, disaster recovery, and big data analytics. Amazon S3 stores data as objects within resources called buckets. You can store as many objects as you want within a bucket, and write, read, and delete objects in your bucket. Objects can be up to 5 terabytes in size. You can control access to the bucket (for example, which can create, delete, and retrieve objects in the bucket), view access logs for the bucket and its objects, and choose the AWS region where Amazon S3 will store the bucket and its contents.

Use Cases Amazon S3

Backup

Amazon S3 offers a highly durable, scalable, and secure solution for backing up and archiving your critical data. You can use Amazon S3's versioning capability to provide even further protection for your stored data. You can also define lifecycle rules to archive sets of Amazon S3 objects to Amazon Glacier, an extremely low-cost storage service.

Content Storage and Distribution

Amazon S3 provides highly durable and available storage for a variety of content. It allows you to offload your entire storage infrastructure into the cloud, where you can take advantage of Amazon S3's scalability and pay-as-you-go pricing to handle your

growing storage needs. You can distribute your content directly from Amazon S3 or use Amazon S3 as an origin store for delivery of content to your Amazon CloudFront edge locations.

Big Data Analytics

Whether you're storing pharmaceutical or financial data, or multimedia files such as photos and videos, Amazon S3 is the ideal big data object store. AWS offers a comprehensive portfolio of services to help you manage big data by reducing costs, scaling to meet demand, and increasing the speed of innovation.

Static Website Hosting

You can host your entire static website on Amazon S3 for a low-cost, highly available hosting solution that scales automatically to meet traffic demands. With Amazon S3, you can reliably serve your traffic and handle unexpected peaks without worrying about scaling your infrastructure.

Cloud-native Application Data

Amazon S3 provides high performance, highly available storage that makes it easy to scale and maintain cost-effective mobile and Internet-based apps that run fast. With Amazon S3, you can add any amount of content and access it from anywhere, so you can deploy applications faster and reach more customers.

Disaster Recovery

Amazon S3's highly durable, secure, global infrastructure offers a robust disaster recovery solution designed to provide superior data protection. Whether you're looking for disaster recovery in the cloud or from your corporate data center to Amazon S3, AWS has the right solution for you.

Key Features

Security and Access Management

Amazon S3 provides several mechanisms to control and monitor who can access your data as well as how, when, and where they can access it.

Lifecycle Management

Amazon S3 provides a number of capabilities to manage the lifecycle of your data, including automated archival using the lower-cost Amazon Glacier.

Versioning

Amazon S3 allows you to enable versioning so you can preserve, retrieve, and restore every version of every object stored in an Amazon S3 bucket.

Encryption

You can securely upload/download your data to Amazon S3 via SSL-encrypted endpoints. Amazon S3 also provides multiple options for encryption of data at rest, and allows you to manage your own keys or have Amazon S3 manage them for you.

Cost Monitoring and Controls

Amazon S3 has several features for managing and controlling your costs, including bucket tagging to manage cost allocation and integration with Amazon Cloud Watch to receive billing alerts.

Choice of AWS Region

Amazon S3 is available globally in multiple AWS regions. You can choose the region where a bucket is stored to optimize for latency, minimize costs, or address regulatory requirements.

Programmatic Access Using the AWS SDKs

Amazon S3 is supported by the AWS SDKs for Java, PHP, .NET, Python, Node.js, Ruby, and the AWS Mobile SDK. The SDK libraries wrap the underlying REST API, simplifying your programming tasks.

Transfer Data to and from Amazon S3 with Ease AWS

Supports several methods for uploading and retrieving data in Amazon S3 including the public Internet, AWS Direct Connect, and the AWS Import/Export service. And AWS Storage Gateway automatically backs up on-premises data to Amazon S3.

Flexible Storage Options

Amazon S3 is designed for 99.999999999% durability and 99.99% availability of objects over a given year. There is also a low-cost Reduced Redundancy Storage option for less critical data and Amazon Glacier for archiving cold data at the lowest possible cost.

2. AWS Storage Gateway

AWS Storage Gateway is a service connecting an on-premises software appliance with cloud-based storage to provide seamless and secure integration between an organization's on-premises IT environment and AWS's storage infrastructure. The service allows you to securely store data in the AWS cloud for scalable and cost-effective storage. The AWS Storage Gateway supports industry-standard storage protocols that work with your existing applications. It provides low-latency performance by maintaining frequently accessed data on-premises while securely storing all of your data encrypted in Amazon S3 or Amazon Glacier.

The AWS Storage Gateway supports three configurations:

Gateway-Cached Volumes: You can store your primary data in Amazon S3, and retain your frequently accessed data locally. Gateway-Cached volumes provide substantial cost savings on primary storage, minimize the need to scale your storage on-premises, and retain low-latency access to your frequently accessed data.

Gateway-Stored Volumes: In the event you need low-latency access to your entire data set, you can configure your on-premises data gateway to store your primary data locally, and asynchronously back up point-in-time snapshots of this data to Amazon S3. Gateway-Stored volumes provide durable and inexpensive off-site backups that you can recover locally or

from Amazon EC2 if, for example, you need replacement capacity for disaster recovery.

Gateway-Virtual Tape Library (Gateway-VTL): With Gateway-VTL you can have a limitless collection of virtual tapes. Each virtual tape can be stored in a Virtual Tape Library backed by Amazon S3 or a Virtual Tape Shelf backed by Amazon Glacier. The Virtual Tape Library exposes an industry standard iSCSI interface which provides your backup application with on-line access to the virtual tapes. When you no longer require immediate or frequent access to data contained on a virtual tape, you can use your backup application to move it from its Virtual Tape Library to your Virtual Tape Shelf in order to further reduce your storage costs.

Benefits AWS Storage Gateway

Secure

The AWS Storage Gateway securely transfers your data to AWS over SSL and stores data encrypted at rest in Amazon S3 and Amazon Glacier using Advanced Encryption Standard (AES) 256, a secure symmetric-key encryption standard using 256-bit encryption keys.

Durably backed by Amazon S3 and Amazon Glacier

The AWS Storage Gateway durably stores your on-premises application data by uploading it to Amazon S3 and Amazon Glacier. Amazon S3 and Amazon Glacier redundantly store data in multiple facilities and on multiple devices within each facility. Amazon S3 and Amazon Glacier also perform regular, systematic data integrity checks and are built to be automatically self-healing.

Compatible

There is no need to re-architect your on-premises applications. Gateway-Cached volumes and Gateway-Stored volumes expose a standard iSCSI block disk device interface and Gateway-VTL presents a standard iSCSI virtual tape library interface.

Cost-Effective

By making it easy for your on-premises applications to store data on Amazon S3 or Amazon Glacier, AWS Storage Gateway reduces the cost, maintenance, and scaling challenges associated with managing primary, backup and archive storage environments. You pay only for what you use with no long-term commitments.

Designed for use with other Amazon Web Services

Gateway-Stored volumes and Gateway-Cached volumes are designed to seamlessly integrate with Amazon S3, Amazon EBS, and Amazon EC2 by enabling you to store point-in-time snapshots of your on-premises application data in Amazon S3 as Amazon EBS snapshots for future recovery on-premises or in Amazon EC2. This integration allows you to easily mirror data from your on-premises applications to applications running on Amazon EC2 in disaster recovery (DR) and on-demand compute capacity cases. Gateway-VTL integrates with Amazon Glacier and allows you to cost effectively and durably store your archive and long-term backup data.

Optimized for Network Efficiency

The AWS Storage Gateway efficiently uses your internet bandwidth to speed up the upload of your on-premises application data to AWS. The AWS Storage Gateway only

uploads data that has changed, minimizing the amount of data sent over the internet. You can also use AWS Direct Connect to further increase throughput and reduce your network costs by establishing a dedicated network connection between your on-premises gateway and AWS.

Common Use Cases of AWS Storage Gateway

Backup

The AWS Storage Gateway enables your existing on-premise to cloud backup applications to store primary backups on Amazon S3's scalable, reliable, secure, and cost-effective storage service. You can create Gateway-Cached storage volumes and mount them as iSCSI devices to your on-premises backup application servers. All data is securely transferred to AWS over SSL and stored encrypted in Amazon S3 using AES 256-bit encryption. Using Gateway-Cached volumes provides an attractive alternative to the traditional choice of maintaining and scaling costly storage hardware on-premises.

For scenarios where you want to keep your primary data or backups on-premises, you can use Gateway-Stored volumes to keep this data locally, and backup this data off-site to Amazon S3. Gateway-Stored volumes provide an attractive alternative to dealing with the longer recovery times and operational burden of managing off-site tape storage for backups.

Use cases for backup and recovery

Databases

AWS storage solutions deliver highly scalable, durable, and reliable cloud storage for backup, and is designed to support mission-critical databases, including Oracle and SAP. With an easy to use web interface, Amazon S3 is designed to deliver flexibility, agility, geo-redundancy, and robust data protection.

Remote/branch office

Amazon S3 provides scalable and cost-effective storage on-demand for your organization's data. Using AWS Storage Gateway you can easily back up data from one location to another for quick recovery.

Media content

AWS storage solutions are designed to deliver capacity and flexibility to make backup and recovery easy for you. Amazon S3 provides scalable and cost-effective storage, on-demand, and eliminates all the heavy lifting of deploying infrastructure.

Disaster Recovery and Resilience

The AWS Storage Gateway, together with EC2, can mirror your entire production environment for disaster recovery (DR). Planning for business continuity in the event of a power outage, fire, flood, or other disaster can be challenging. It requires investments in redundant infrastructure and staff across multiple datacenters and costly storage replication solutions. AWS Storage Gateway and Amazon EC2 together provide a simple cloud-hosted DR solution. Using Amazon EC2, you can configure virtual machine images of your DR application servers and only pay for these servers when you need them. In the event your on-premises infrastructure goes down, you

simply launch the Amazon EC2 compute instances you need and attach them to copies of your on-premises data. The AWS Storage Gateway addresses the challenges of replicating data for DR by enabling you to create Gateway-Cached volumes that store your data in Amazon S3. By storing your data using the AWS Storage Gateway, you will be prepared for DR if you lose your on-premises application or storage.

Benefits of Using AWS for Disaster Recovery

Fast Performance

Fast disk-based storage and retrieval of files.

No Tape

Eliminate costs associated with transporting, storing, and retrieving tape media and associated tape backup software.

Compliance

Fast retrieval of files allows you to avoid fines for missing compliance deadlines.

Elasticity

Add any amount of data, quickly. Easily expire and delete without handling media.

Secure

Secure and durable cloud disaster recovery platform with industry-recognized certifications and audits.

Partners

AWS solution provider and system integration partners to help with your deployment.

Corporate File Sharing

Managing on-premises storage for departmental file shares and home directories typically results in high capital and maintenance costs, under-utilized hardware, and restrictive user quotas. The AWS Storage Gateway addresses these on-premises scaling and maintenance issues by enabling you to seamlessly store your corporate file shares on Amazon S3, while keeping a copy of your frequently accessed files on-premises. This minimizes the need to scale your on-premises file storage infrastructure, while still providing low-latency access to your frequently accessed data. Using the AWS Storage Gateway, you can create Gateway-Cached storage volumes up to 32 TB in size and mount them as iSCSI devices from your on-premises file servers. You can then expose these volumes as Common Internet File System (CIFS) shares or Network File System (NFS) mount points to your client machines. The AWS Storage Gateway durably stores files written to these shares or mount points in Amazon S3, while maintaining a cache of recently written and recently read files locally on your on-premises storage hardware for low-latency access. Since you only pay for the storage you actually use, you can scale your storage on-demand and avoid the costs of under-utilized hardware.

Data Mirroring to Cloud-Based Compute Resources

If you want to leverage Amazon EC2's on-demand compute capacity for additional capacity during peak periods, for new projects, or as a more cost-effective way to run your normal workloads, you can use the AWS Storage Gateway to mirror your volume

data to Amazon EC2 instances. If you're running development and User Acceptance Testing (UAT) environments in Amazon EC2 to take advantage of AWS's on-demand compute capacity, you can use the AWS Storage Gateway to ensure these environments have ongoing access to the latest data from your production systems on-premises.

Use Cases for Gateway-Virtual Tape Library

Magnetic Tape Replacement for Archiving and Long-Term Backup - Using Gateway-VTL, you can store data requiring long term retention and infrequent access without changing your existing backup applications and tape-based processes. Although magnetic tape-based storage can be cost-effective when operated at scale, it can be a drain on resources as one (or more) tape libraries need to be maintained (often in geographically distinct locations) requiring specialized personnel, and taking up valuable space in data centers. In addition, the tapes themselves must be carefully stored and managed, which can include periodically copying data from old tapes onto new ones to ensure that your data can still be read as tape technology standards evolve.

Tape's low cost potential also requires accurate capacity planning, a process that is usually error-prone, especially when storage growth is unpredictable, as it often is. Over provisioning capacity can result in under utilization and higher costs, while under provisioning can trigger expensive hardware upgrades far earlier than planned. Even when capacity planning is accurate, periodic hardware upgrades are still common as older tape libraries are less efficient and therefore costlier to operate. Archiving valuable data using a tape-based solution also requires costly, multi-site, redundant data centers and offsite vaulting to guarantee durability. This approach also requires manual handling of tape media which increases the risk of data loss.

By using Gateway-VTL, you can eliminate these challenges associated with owning and operating on-premises physical tape infrastructure by storing your archive and long-term backup data on a limitless collection of virtual tapes. Your virtual tapes can be stored in a Virtual Tape Library backed by Amazon S3 or a Virtual Tape Shelf backed by Amazon Glacier. The Virtual Tape Library provides your backup application with on-line access to the virtual tapes. When you no longer require immediate or frequent access to data contained on a virtual tape, you can use your backup application to move it from its Virtual Tape Library to your Virtual Tape Shelf in order to further reduce your storage costs.

Gateway-VTL allows you to eliminate the need for large upfront capital expense and expensive multi-year support commitments. With Gateway-VTL you pay only for the capacity you use and scale as your needs grow. With the Gateway-VTL solution you also don't need to worry about transporting storage media to offsite facilities and manual handling of tape media. The Gateway-VTL solution reduces your costs and simplifies your data management process while improving the durability of your archive and long-term backup solution.

3. Amazon Glacier

Amazon Glacier is an extremely low-cost cloud archive storage service that provides secure and durable storage for data archiving and online backup. In order to keep costs low, Amazon Glacier is optimized for data that is infrequently accessed and for which retrieval times of several hours are suitable. With Amazon Glacier, customers can reliably store large

or small amounts of data for as little as \$0.01 per gigabyte per month, a significant savings compared to on-premises solutions. Companies typically over-pay for data archiving. First, they're forced to make an expensive upfront payment for their archiving solution (which does not include the ongoing cost for operational expenses such as power, facilities, staffing, and maintenance). Second, since companies have to guess what their capacity requirements will be, they understandably over-provision to make sure they have enough capacity for data redundancy and unexpected growth. This set of circumstances results in under-utilized capacity and wasted money. With Amazon Glacier, you pay only for what you use. Amazon Glacier changes the game for data archiving and cloud backup as you pay nothing upfront, pay a very low price for storage, and can scale your usage up or down as needed, while AWS handles all of the operational heavy lifting required to do data retention well. It only takes a few clicks in the AWS Management Console to set up Amazon Glacier and then you can upload any amount of data you choose.

Benefits of Amazon Glacier

Low Cost

Starting at \$0.01 per gigabyte per month, Amazon Glacier allows you to archive large amounts of data at a very low cost. You pay for what you need, with no minimum commitments or up-front fees.

Secure

Amazon Glacier supports data transfer over SSL and automatically encrypts your data at rest. You can also control access to your data using AWS Identity and Access Management (IAM). For customers who must comply with regulatory standards such as PCI and HIPAA, Amazon Glacier's data protection features can be used as part of an overall strategy to achieve compliance. The various data security and reliability features offered by Amazon Glacier are described in detail below.

Encryption by default

Amazon Glacier automatically encrypts data at rest using Advanced Encryption Standard (AES) 256-bit symmetric keys and supports secure transfer of your data over Secure Sockets Layer (SSL).

Immutable archives

Data stored in Amazon Glacier is immutable, meaning that after an archive is created it cannot be updated. This ensures that data such as compliance and regulatory records cannot be altered after they have been archived.

Flexible access control with IAM policies

Amazon Glacier supports Identity and Access Management (IAM) policies, which enables organizations with multiple employees to create and manage multiple users under a single AWS account. With IAM policies, you create fine-grained policies to control to your Amazon Glacier vaults. You can write IAM policies to selectively grant or revoke certain permissions and actions on each Amazon Glacier vault.

Mandatory request signing

Amazon Glacier requires all requests to be signed for authentication protection. To sign a request, you calculate a digital signature using a cryptographic hash function that returns

a hash value that you include in the request as your signature. After receiving your request, Amazon Glacier recalculates the signature using the same hash function and input that you used to sign the request before processing the request.

Durable

Amazon Glacier provides a highly durable storage infrastructure designed for online backup and archival. Your data is redundantly stored across multiple facilities and multiple devices in each facility. Amazon Glacier provides a highly durable storage infrastructure designed for long-term data archival storage. It is designed to provide average annual durability of 99.999999999% for an archive. The service redundantly stores data in multiple facilities and on multiple devices within each facility. To increase durability, Amazon Glacier synchronously stores your data across multiple facilities before confirming a successful upload.

To prevent corruption of data packets over the wire, Amazon Glacier uploads the checksum of the data during data upload. It compares the received checksum with the checksum of the received data to detect bit flips over the wire. Similarly, it validates data authenticity with checksums during data retrieval. Unlike traditional systems, that can require laborious data verification and manual repair, Amazon Glacier performs regular, systematic data integrity checks and is built to be automatically self-healing.

Simple

Amazon Glacier allows you to offload the administrative burden of operating storage infrastructure to AWS. Data uploaded to Amazon Glacier remains stored for as long as needed with no additional effort from you.

Flexible

Amazon Glacier scales to meet your storage needs. There is no limit to how much data you can store, and you can choose to store your data in the AWS region that supports your regulatory and business criteria.

Integrated

Through Amazon S3 lifecycle policies, you can optimize your storage costs by moving infrequently accessed objects from Amazon S3 to Amazon Glacier (or vice-versa).

4. Amazon Elastic Block Store

Amazon Elastic Block Store (Amazon EBS) provides persistent block level storage volumes for use with Amazon EC2 instances in the AWS cloud. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability. Amazon EBS volumes offer the consistent and low-latency performance needed to run your workloads. With Amazon EBS, you can scale your usage up or down within minutes—all while paying a low price for only what you provision.

Benefits Amazon Glacier

Reliable, secure storage

Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure. Amazon EBS encryption provides seamless support for data at rest security and data in motion security between EC2 instances and EBS volumes. Amazon's flexible access control policies allow you to specify who can access which EBS volumes. Access control plus encryption offers a strong defense-in-depth security strategy for your data.

Consistent and low-latency performance

Amazon EBS General Purpose (SSD) volumes and Amazon EBS Provisioned IOPS (SSD) volumes deliver low-latency through SSD technology and consistent I/O performance scaled to the needs of your application. Stripe multiple volumes together to achieve even higher I/O performance.

Backup, restore, innovate

Backup your data by taking point-in-time snapshots of your Amazon EBS volumes. Boost the agility of your business by using Amazon EBS snapshots to create new EC2 instances.

Quickly scale up, easily scale down

Increase or decrease block storage and performance within minutes, enjoying the freedom to adjust as your needs evolve. Commission thousands of volumes simultaneously.

Geographic flexibility

Amazon EBS provides the ability to copy snapshots across AWS regions, enabling geographical expansion, data center migration, and disaster recovery.

EBS Features

Amazon EBS Snapshots- Amazon EBS provides the ability to save point-in-time snapshots of your volumes to Amazon S3. Amazon EBS Snapshots are stored incrementally: only the blocks that have changed after your last snapshot are saved, and you are billed only for the changed blocks. If you have a device with 100 GB of data but only 5 GB has changed after your last snapshot, a subsequent snapshot consumes only 5 additional GB and you are billed only for the additional 5 GB of snapshot storage, even though both the earlier and later snapshots appear complete.

When you delete a snapshot, you remove only the data not needed by any other snapshot. All active snapshots contain all the information needed to restore the volume to the instant at which that snapshot was taken. The time to restore changed data to the working volume is the same for all snapshots.

Snapshots can be used to instantiate multiple new volumes, expand the size of a volume, or move volumes across Availability Zones. When a new volume is created, you may choose to create it based on an existing Amazon EBS snapshot. In that scenario, the new volume begins as an exact replica of the snapshot.

The following are key features of Amazon EBS Snapshots:

Immediate access to Amazon EBS volume data - After a volume is created from a snapshot, there is no need to wait for all of the data to transfer from Amazon S3 to your Amazon EBS volume before your attached instance can start accessing the volume. Amazon EBS Snapshots implement lazy loading, so that you can begin using them right away.

Resizing Amazon EBS volumes - When you create a new Amazon EBS volume based on a snapshot, you may specify a larger size for the new volume. Make certain that your file system or application supports resizing a device.

Sharing Amazon EBS Snapshots - Amazon EBS Snapshots' shareability makes it easy for you to share data with your co-workers or others in the AWS community. Authorized users can create their own Amazon EBS volumes based on your Amazon EBS shared snapshots; your original snapshot remains intact. If you choose, you can also make your data available publicly to all AWS users.

Copying Amazon EBS Snapshots across AWS regions - Amazon EBS's ability to copy snapshots across AWS regions makes it easier to leverage multiple AWS regions for geographical expansion, data center migration and disaster recovery. You can copy any snapshot accessible to you: snapshots you created; snapshots shared with you; and snapshots from the AWS Marketplace, VM Import/Export, and AWS Storage Gateway.

Amazon EBS-Optimized Instances

For an additional low, hourly fee, customers can launch certain Amazon EC2 instance types as EBS-optimized instances. EBS-optimized instances enable EC2 instances to fully use the IOPS provisioned on an EBS volume. EBS-optimized instances deliver dedicated throughput between Amazon EC2 and Amazon EBS, with options between 500 and 4,000 Megabits per second (Mbps) depending on the instance type used. The dedicated throughput minimizes contention between Amazon EBS I/O and other traffic from your EC2 instance, providing the best performance for your EBS volumes. EBS-optimized instances are designed for use with all Amazon EBS volume types.

Amazon EBS Availability and Durability

Amazon EBS volumes are designed to be highly available and reliable. At no additional charge to you, Amazon EBS volume data is replicated across multiple servers in an Availability Zone to prevent the loss of data from the failure of any single component. Amazon EBS volumes are designed for an annual failure rate (AFR) of between 0.1% - 0.2%, where failure refers to a complete or partial loss of the volume, depending on the size and performance of the volume. This makes EBS volumes 20 times more reliable than typical commodity disk drives, which fail with an AFR of around 4%. EBS also supports a snapshot feature, which is a good way to take point-in-time backups of your data.

Amazon EBS Encryption and AWS Identity and Access Management

Amazon EBS encryption offers seamless encryption of EBS data volumes and snapshots, eliminating the need to build and manage a secure key management infrastructure. EBS encryption enables data at rest security by encrypting your data volumes and snapshots using

Amazon-managed keys or keys you create and manage using the AWS Key Management Service (KMS). In addition, the encryption occurs on the servers that host EC2 instances, providing encryption of data as it moves between EC2 instances and EBS data volumes. Access to Amazon EBS volumes is integrated with AWS Identity and Access Management (IAM). IAM enables access control to your Amazon EBS volumes. For more information, see AWS Identity and Access Management.

Functionality

AWS IAM allows you to:

- ✓ **Manage IAM users and their access** – You can create users in IAM, assign them individual security credentials (in other words, access keys, passwords, and multi-factor authentication devices), or request temporary security credentials to provide users access to AWS services and resources. You can manage permissions in order to control which operations a user can perform.
- ✓ **Manage IAM roles and their permissions** – You can create roles in IAM and manage permissions to control which operations can be performed by the entity, or AWS service, that assumes the role. You can also define which entity is allowed to assume the role.
- ✓ **Manage federated users and their permissions** – You can enable identity federation to allow existing identities (e.g. users) in your enterprise to access the AWS Management Console, to call AWS APIs, and to access resources, without the need to create an IAM user for each identity.

Chapter 12

Exploring Cloudfront: Amazon's Content Delivery Network (CDN) Services

1. Amazon's Content Delivery Network (CDN)

A CDN is large distributed system of servers are deployed in data centers around the world, often over multiple backbones. Their key purpose is to serve the end user with the best experience possible by lowering latency and increasing performance and more noticeably to the end user, improving page load times. This is done by caching static content in locations closer to the user. Amazon Cloudfront CDN provides a seamless integration with all of its other Web Services.

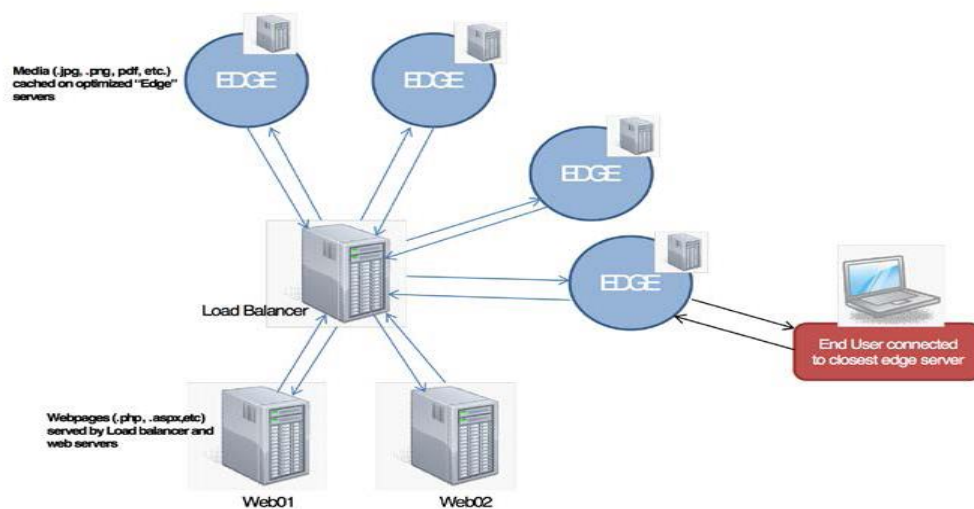


Figure 27: CDN architecture

Cloudfront CDN Benefits

Fast – CDN uses “edges” around the world to cache static content close to end users and to provide lower latency and higher sustained data transfer rates in a scalable manner.

Simple to use – Amazons AWS management console allows for easy setup. Designed to work with other Amazon Web Services – Since we are already using Amazon’s EC2 servers behind an Elastic Load Balancer, we can use Cloudfront to deliver the client’s entire site.

Cost-Effective – Pay for the content you deliver.

Flexible – The service automatically adjusts when server load increases and decreases.

Reliable – Built with Amazon’s highly reliable infrastructure and with edges around the world, end users are routed to the closest available edge.

Global – As mentioned before, edge locations can be setup around the world.

Safe - Better protection from DoS attacks on origin server.

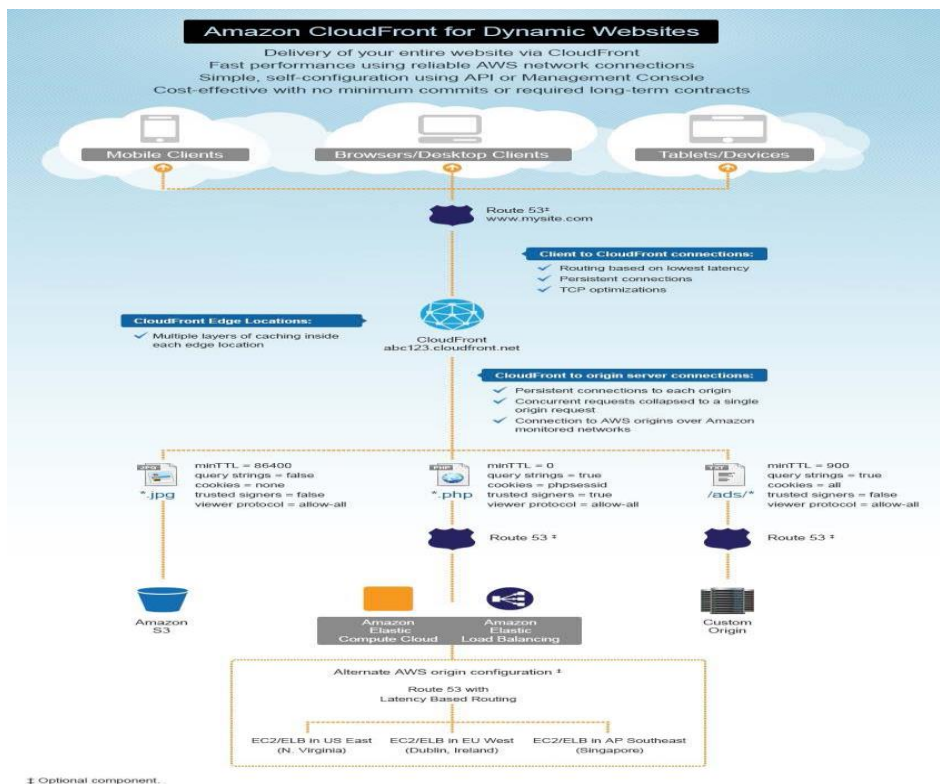


Figure 28: Cloudfront

Common Use Cases of Cloudfront CDN

There are many great use cases for Amazon Cloudfront, including:

Delivering your entire website or web application - A typical website generally contains a mix of static content and dynamic content. Static content includes images or style sheets; dynamic or application generated content includes elements of your site that are personalized to each viewer. A website may also have forms that a user submits to log in, search or post a comment.

You can use a single Cloudfront distribution as a content distribution network to deliver your entire website, including both static and dynamic or interactive content to the end users to content uploaded by the end user to the origin. This means that you can continue to use a single domain name (e.g., www.mysite.com) for your entire website without the need to separate your static and dynamic content. Meanwhile, you can still continue to use separate origin servers for different types of content on your website. Amazon Cloudfront provides you with granular control for configuring multiple origin servers and caching properties for different URLs on your website. These performance optimizations and functionality can help speed up the download of your entire website which can help lower site abandonment.

Amazon Cloudfront improves performance of entire website - Amazon Cloudfront can help improve performance of your entire website in the following ways:

Amazon Cloudfront can cache static content at each edge location. This means that your popular static content (e.g., your site's logo, navigational images, cascading style sheets, JavaScript code, etc.) will be available at a nearby edge location for the browsers to download with low latency and improved performance for viewers. Caching popular static content with Amazon Cloudfront also helps you offload requests for such files from your origin sever – Cloudfront serves the cached copy when available and only makes a request to your origin server if the edge location receiving the browser's request does not have a copy of the file.

Amazon Cloudfront proxies' requests for dynamic or interactive content (e.g., web forms, comments, login boxes, etc.) back to your origin running in an AWS Region or any other origin. Each of your end users is routed to the edge location closest to them, in terms of internet latency. Then, their requests are carried back to your origin server running in AWS on connections that Amazon monitors and optimizes for performance. Amazon Cloudfront also reuses existing connections between the Cloudfront edge and the origin server reducing connection setup latency for each origin request. Other connection optimizations are also applied to avoid internet bottlenecks and fully utilize available bandwidth between the edge location and the viewer. This means that Amazon Cloudfront can speed-up the delivery of your dynamic content and provide your viewers with a consistent and reliable, yet personalized experience when navigating your web application on Cloudfront can speed-up the delivery of your dynamic content and provide your viewers with a consistent and reliable, yet personalized experience when navigating your web application.

Amazon Cloudfront allows content to be uploaded to your origin server. All requests to upload content are proxied by Amazon Cloudfront edge locations back to your origin. Amazon Cloudfront also applies the same performance benefits to upload requests as those applied to the requests for downloading dynamic content.

You can also use Amazon Cloudfront edge locations to upload large files (up to 20GB per file) to your origin using the PUT HTTP method. Amazon Cloudfront can also be used to deliver your API using HTTP methods such as GET, HEAD, POST, PUT, DELETE, PATCH and OPTIONS.

Distributing software or other large files

Amazon Cloudfront is a good choice for software developers who wish to distribute applications, updates or other downloadable software to end users. Amazon Cloudfront's high data transfer rates speed up downloading your applications, improving the customer experience and lowering your costs. Amazon Cloudfront also offers lower prices than Amazon S3 at higher usage tiers.

Delivering media files

If your application involves rich media – audio or video – that is frequently accessed, you will benefit from Amazon Cloudfront's lower data transfer prices and improved data transfer speeds. Amazon Cloudfront offers multiple options for delivering your media files – both pre-recorded media and live media.

1.1.1. Streaming of pre-recorded media:

You can deliver your on-demand media using Adobe's Real Time Messaging Protocol (RTMP) streaming via Amazon Cloudfront. You store the original copy of your media files in Amazon S3 and use Amazon Cloudfront for low-latency delivery of your media content. Amazon Cloudfront integrates with Amazon S3 so you can configure media streaming by making a simple API call or with a few clicks in the AWS Management Console. You also benefit for the high throughput delivery of your media when using Amazon Cloudfront, so you can deliver content in full HD quality to your viewers.

1.1.2. Progressive download of on-demand media:

You can store the original versions of your media content in Amazon S3 and configure an Amazon Cloudfront download distribution for progressive download of your video and audio files. Popular media files are cached at the edge to help you scale and give your viewers the best possible performance.

1.1.3. Delivering live events:

If you need to deliver a live event – audio or video – to a global audience, Amazon Cloudfront can improve performance and help offload requests to your origin infrastructure by caching your live media for a short period of time and collapsing simultaneous requests for the same media fragment to a smaller number of requests sent to the origin. In addition, Amazon Cloudfront's live HTTP solutions give you the ability to deliver your live event to viewers using different device platforms, including both Flash based and Apple iOS devices.

Chapter 13

An Introduction to Amazon's Database Services

1. Amazon RDS

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database management tasks, freeing you up to focus on your applications and business. Amazon RDS gives you access to the capabilities of a familiar MySQL, Oracle, SQL Server, PostgreSQL or Amazon Aurora relational database management system. This means that the code, applications, and tools you already use today with your existing databases can be used with Amazon RDS. Amazon RDS automatically patches the database software and backs up your database, storing the backups for a user-defined retention period and enabling point-in-time recovery. You benefit from the flexibility of being able to scale the compute resources or storage capacity associated with your Database Instance (DB Instance) via a single API call.

Amazon RDS DB Instances can be provisioned with either standard storage or Provisioned IOPS storage. Amazon RDS Provisioned IOPS is a storage option designed to deliver fast, predictable, and consistent I/O performance, and is optimized for I/O-intensive, transactional (OLTP) database workloads. In addition, Amazon RDS makes it easy to use replication to enhance availability and reliability for production workloads. Using the Multi-AZ deployment option you can run mission critical workloads with high availability and built-in

automated fail-over from your primary database to a synchronously replicated secondary database in case of a failure. Amazon RDS for MySQL also enables you to scale out beyond the capacity of a single database deployment for read-heavy database workloads. As with all Amazon Web Services, there are no up-front investments required, and you pay only for the resources you use.

Service Highlights

Simple to Deploy Database Web Service

Amazon RDS makes it easy to go from project conception to deployment. Use the AWS Management Console or simple API calls to access the capabilities of a production-ready relational database in minutes without worrying about infrastructure provisioning or installing and maintaining database software.

Managed

Amazon RDS handles time-consuming database management tasks, such as backups, patch management, and replication, allowing you to pursue higher value application development or database refinements.

Compatible

With Amazon RDS, you get native access to a relational database management system. This facilitates compatibility with your existing tools and applications. In addition, Amazon RDS gives you optional control over which supported engine version powers your DB Instance via DB Engine Version Management.

Fast, Predictable Performance

Database Instances using Amazon RDS's MySQL, Oracle, SQL Server, and Oracle engines can be provisioned with General Purpose (SSD) Storage, Provisioned IOPS (SSD) Storage, or Magnetic Storage. Amazon RDS General Purpose (SSD) Storage delivers a consistent baseline of 3 IOPS per provisioned GB and provides the ability to burst up to 3,000 IOPS. Amazon RDS Provisioned IOPS (SSD) Storage is a high-performance storage option designed to deliver fast, predictable, and consistent performance for I/O intensive transactional database workloads. You can provision from 1,000 IOPS to 30,000 IOPS per DB Instance. (Maximum realized IOPS will vary by engine type.). Magnetic Storage (formerly known as Amazon RDS Standard storage) is useful for small database workloads where data is accessed less frequently.

Database Instances using the Amazon Aurora engine employ a fault-tolerant, self-healing SSD-backed virtualized storage layer purpose-built for database workloads.

Scalable Database in the Cloud

You can scale the compute and storage resources available to your database to meet your application needs using the Amazon RDS API or the AWS Management Console. If you are using Amazon RDS Provisioned IOPS storage with Amazon RDS for MySQL, Oracle, or PostgreSQL, you can provision and scale the storage up to 3TB and IOPS to up to 30,000. Note that maximum realized IOPS will vary by engine type. In addition, for the MySQL, PostgreSQL, and Amazon Aurora database engines, you can also associate one or more Read

Replicas with your database instance deployment, enabling you to scale beyond the capacity of a single database instance for read-heavy workloads.

The Amazon Aurora database engine allows you to scale your storage up to 64TB. You can associate up to 15 Amazon Aurora Replicas with your database instance deployment. Amazon Aurora Replicas share the same underlying storage as the source instance, lowering costs and avoiding the need to copy data to the replica nodes.

Reliable

Amazon RDS has multiple features that enhance reliability for critical production databases, including automated backups, DB snapshots, automatic host replacement, and Multi-AZ deployments. Amazon RDS runs on the same highly reliable infrastructure used by other Amazon Web Services. For the Amazon Aurora engine, Amazon RDS uses RDS Multi-AZ technology to automate failover to one of up to 15 Aurora Replicas you have created in any of three Availability Zones.

Designed for use with other Amazon Web Services

Amazon RDS is tightly integrated with other Amazon Web Services. For example, an application running in Amazon EC2 will experience low-latency database access to an Amazon RDS DB Instance in the same region.

Secure

Amazon RDS provides a number of mechanisms to secure your DB Instances. Amazon RDS allows you to encrypt your databases using keys you manage through AWS Key Management Service (KMS). On a database instance running with Amazon RDS encryption, data stored at rest in the underlying storage is encrypted, as are its automated backups, read replicas, and snapshots. Amazon RDS supports Transparent Data Encryption in SQL Server and Oracle. Transparent Data Encryption in Oracle is integrated with AWS CloudHSM, which allows you to securely generate, store, and manage your cryptographic keys in single-tenant Hardware Security Module (HSM) appliances within the AWS cloud. Amazon RDS includes web service interfaces to configure firewall settings that control network access to your database. Amazon RDS allows you to run your DB Instances in Amazon Virtual Private Cloud (Amazon VPC). Amazon VPC enables you to isolate your DB Instances by specifying the IP range you wish to use, and connect to your existing IT infrastructure through industry-standard encrypted IPsec VPN.

Inexpensive

You pay very low rates and only for the resources you actually consume. In addition, you benefit from the option of On-Demand pricing with no up-front or long-term commitments, or even lower hourly rates via our reserved pricing option.

On-Demand DB Instances let you pay for compute capacity by the hour with no long-term commitments. This frees you from the costs and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into much smaller variable costs.

Reserved DB Instances give you the option to reserve capacity within a datacenter and in turn receive a significant discount on the hourly charge for instances that are covered by the reservation. You can choose between are three RI payment options -- No Upfront, Partial Upfront, All Upfront -- which enable you to balance the amount you pay upfront with your effective hourly price and receive a significant discount over On-Demand prices.

2. Amazon Aurora

Amazon Aurora is a MySQL-compatible, relational database engine that combines the speed and availability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases. Amazon Aurora provides up to five times better performance than MySQL at a price point one tenth that of a commercial database while delivering similar performance and availability. Amazon Aurora joins MySQL, Oracle, Microsoft SQL Server, and PostgreSQL as the fifth database engine available to customers through Amazon RDS. Amazon RDS handles routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair.

3. Amazon DynamoDB

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed database and supports both document and key-value data models. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, the Internet of things (IoT), and many other applications.

Benefit

Fast, Consistent Performance

Amazon DynamoDB is designed to deliver consistent, fast performance at any scale for all applications. Average service-side latencies are typically single-digit milliseconds. As our data volumes grow and application performance demands increase, Amazon DynamoDBmatic partitioning and solid state drive (SSD) technologies to meet your throughput requirements and deliver low latencies at any scale.

Highly Scalable

When creating a table, simply specify how much request capacity you require. If your hroughput requirements change, simply update your table's request capacity using the AWS Management Console or the Amazon DynamoDB API. Amazon DynamoDB manages all the scaling behind the scenes, and you are still able to achieve your prior throughput levels while scaling is underway.

Flexible

Amazon DynamoDB supports both document and key-value data structures, giving you the flexibility to design the best architecture that is optimal for your application.

Fine-grained Access Control

Amazon DynamoDB integrates with AWS Identity and Access Management (IAM) for fine-grained access control for users within your organization. You can assign unique security credentials to each user and control each user's access to services and resources.

Fully Managed

Amazon DynamoDB is a fully managed cloud NoSQL database service—you simply create a database table, set your throughput, and let the service handle the rest. You no longer need to worry about database management tasks such as hardware or software provisioning, setup and configuration, software patching, operating a reliable, distributed database cluster, or partitioning data over multiple instances as you scale.

4. Amazon Redshift

Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse solution that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools. You can start small with no commitments or upfront costs and scale to a petabyte or more. Amazon Redshift delivers fast query performance by using columnar storage technology to improve I/O efficiency and parallelizing queries across multiple nodes. Amazon Redshift uses standard PostgreSQL JDBC and ODBC drivers, allowing you to use a wide range of familiar SQL clients. Data load speed scales linearly with cluster size, with integrations to Amazon S3, Amazon DynamoDB, Amazon Elastic MapReduce, Amazon Kinesis, or any SSH-enabled host. We've automated most of the common administrative tasks associated with provisioning, configuring and monitoring a data warehouse. Backups to Amazon S3 are continuous, incremental, and automatic. Restores are fast; you can start querying in minutes while your data is spooled down in the background. Enabling disaster recovery across regions takes just a few clicks. Security is built-in. You can encrypt data at rest and in transit using hardware accelerated AES-256 and SSL, isolate your clusters using Amazon VPC, and even manage your keys using hardware security modules (HSMs). All API calls, connection attempts, queries, and changes to the cluster are logged and auditable.

Features and Benefits Of Amazon Redshift

Amazon Redshift delivers fast query performance by using columnar storage technology to improve I/O efficiency and parallelizing queries across multiple nodes. Amazon Redshift has custom JDBC and ODBC drivers that you can download from the Connect Client tab of our Console, allowing you to use a wide range of familiar SQL clients. You can also use standard PostgreSQL JDBC and ODBC drivers. Data load speed scales linearly with cluster size, with integrations to Amazon S3, Amazon DynamoDB, Amazon Elastic MapReduce, Amazon Kinesis or any SSH-enabled host.

Amazon Redshift's data warehouse architecture allows you to automate most of the common administrative tasks associated with provisioning, configuring and monitoring a cloud data warehouse. Backups to Amazon S3 are continuous, incremental and automatic. Restores are fast; you can start querying in minutes while your data is spooled down in the background. Enabling disaster recovery across regions takes just a few clicks. Security is built-in. You can encrypt data at rest and in transit using hardware-accelerated AES-256 and SSL, isolate your clusters using Amazon VPC and even manage your keys using AWS Key Management

Service (KMS) and hardware security modules (HSMs). All API calls, connection attempts, queries and changes to the cluster are logged and auditable. You can use AWS CloudTrail to audit Redshift API calls.

Optimized for Data Warehousing

Amazon Redshift uses a variety of innovations to obtain very high query performance on datasets ranging in size from a hundred gigabytes to a petabyte or more. It uses columnar storage, data compression, and zone maps to reduce the amount of I/O needed to perform queries. Amazon Redshift has a massively parallel processing (MPP) data warehouse architecture, parallelizing and distributing SQL operations to take advantage of all available resources. The underlying hardware is designed for high performance data processing, using local attached storage to maximize throughput between the CPUs and drives, and a 10GigE mesh network to maximize throughput between nodes.

Scalable

With a few clicks of the AWS Management Console or a simple API call, you can easily change the number or type of nodes in your cloud data warehouse as your performance or capacity needs change. Dense Storage (DS) nodes allow you to create very large data warehouses using hard disk drives (HDDs) for a very low price point. Dense Compute (DC) nodes allow you to create very high performance data warehouses using fast CPUs, large amounts of RAM and solid-state disks (SSDs). Amazon Redshift enables you to start with as little as a single 160GB DC1.Large node and scale up all the way to a petabyte or more of compressed user data using 16TB DS2.8XLarge nodes. While resizing, Amazon Redshift places your existing cluster into read-only mode, provisions a new cluster of your chosen size, and then copies data from your old cluster to your new one in parallel. You can continue running queries against your old cluster while the new one is being provisioned. Once your data has been copied to your new cluster, Amazon Redshift will automatically redirect queries to your new cluster and remove the old cluster.

Cheap

No Up-Front Costs

You pay only for the resources you provision. You can choose On-Demand pricing with no up-front costs or long-term commitments, or obtain significantly discounted rates with Reserved Instance pricing. On-Demand pricing starts at just \$0.25/hour per 160GB DC1.Large node or \$0.85/hour per 2TB DS2.XLarge node. With Partial Upfront Reserved Instances, you can lower your effective price to \$0.10/hour per DC1.Large node (\$5,500/TB/year) or \$0.228/hour per DS2.XLarge node (\$999/TB/year).

Simple

Get Started in Minutes With a few clicks in the AWS Management Console or simple API calls, you can create a cluster, specifying its size, underlying node type, and security profile. Amazon Redshift will provision your nodes, configure the connections between them, and secure the cluster. Your data warehouse should be up and running in minutes.

Fully Managed

Amazon Redshift handles all the work needed to manage, monitor, and scale your data warehouse, from monitoring cluster health and taking backups to applying patches and

upgrades. You can easily resize your cluster as your performance and capacity needs change. By handling all these time-consuming, labor-intensive tasks, Amazon Redshift frees you up to focus on your data and business.

Fault Tolerant

Amazon Redshift has multiple features that enhance the reliability of your data warehouse cluster. All data written to a node in your cluster is automatically replicated to other nodes within the cluster and all data is continuously backed up to Amazon S3. Amazon Redshift continuously monitors the health of the cluster and automatically re-replicates data from failed drives and replaces nodes as necessary.

Automated Backups

Amazon Redshift's automated snapshot feature continuously backs up new data on the cluster to Amazon S3. Snapshots are continuous, incremental and automatic. Amazon Redshift stores your snapshots for a user-defined period, which can be from one to thirty-five days. You can take your own snapshots at any time, which leverage all existing system snapshots and are retained until you explicitly delete them. Redshift can also asynchronously replicate your snapshots to S3 in another region for disaster recovery. Once you delete a cluster, your system snapshots are removed but your user snapshots are available until you explicitly delete them.

Fast Restores

You can use any system or user snapshot to restore your cluster using the AWS Management Console or the Amazon Redshift APIs. Your cluster is available as soon as the system metadata has been restored and you can start running queries while user data is spooled down in the background.

Secure

Encryption

With just a couple of parameter settings, you can set up Amazon Redshift to use SSL to secure data in transit and hardware-accelerated AES-256 encryption for data at rest. If you choose to enable encryption of data at rest, all data written to disk will be encrypted as well as any backups. By default, Amazon Redshift takes care of key management but you can choose to manage your keys using your own hardware security modules (HSMs), AWS CloudHSM, or AWS Key Management Service.

Network Isolation

Amazon Redshift enables you to configure firewall rules to control network access to your data warehouse cluster. You can run Amazon Redshift inside Amazon Virtual Private Cloud (Amazon VPC) to isolate your data warehouse cluster in your own virtual network and connect it to your existing IT infrastructure using industry-standard encrypted IPsec VPN.

Audit and Compliance. Amazon Redshift integrates with AWS CloudTrail to enable you to audit all Redshift API calls. Amazon Redshift also logs all SQL operations, including connection attempts, queries and changes to your database. You can access these logs using SQL queries against system tables or choose to have them downloaded to a secure

location on Amazon S3. Amazon Redshift is compliant with SOC1, SOC2, SOC3 and PCI DSS Level 1 requirements.

Compatible

SQL

Amazon Redshift is a SQL data warehouse solution and uses industry standard ODBC and JDBC connections. You can download our custom JDBC and ODBC drivers from the Connect Client tab of our Console. Many popular software vendors have certified Amazon Redshift with their offerings to enable you to continue to use the tools you do today.

Integrated

Amazon Redshift is integrated with other AWS services and has built in commands to load data in parallel to each node from Amazon S3, Amazon DynamoDB or your EC2 and on-premise servers using SSH. AWS Data Pipeline, Amazon Kinesis, and AWS Lambda integrate with Amazon Redshift as a data target.

5. Amazon ElastiCache

Amazon ElastiCache is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases. ElastiCache supports two open-source in-memory caching engines:

Memcached: A widely adopted memory object caching system. ElastiCache is protocol compliant with Memcached, so popular tools that you use today with existing Memcached environments will work seamlessly with the service.

Redis: A popular open-source in-memory key-value store that supports data structures such as sorted sets and lists. ElastiCache supports Redis master / slave replication which can be used to achieve redundancy across Availability Zones. Amazon ElastiCache automatically detects and replaces failed nodes, reducing the overhead associated with self-managed infrastructures. It provides a resilient system that mitigates the risk of overloaded databases, which slow website and application load times. Through integration with Amazon CloudWatch, Amazon ElastiCache provides enhanced visibility into key performance metrics associated with your Memcached or Redis nodes.

Benefits

Simple to Deploy

Amazon ElastiCache makes it very easy to deploy a Memcached or Redis compliant cache environment. Use the AWS Management Console or simple API calls to access the capabilities of a production-ready cloud cache cluster in minutes without worrying about infrastructure provisioning, or installing and maintaining cache software.

Managed

Amazon ElastiCache automates time-consuming management tasks --such as patch management, failure detection and recovery-- allowing you to pursue higher value application development.

Compatible

With Amazon ElastiCache, you get native access to the Memcached or Redis in-memory caching environments. This facilitates compatibility with your existing tools and applications.

Elastic

With a simple API call or a few clicks of the AWS Management Console, you can add or delete Cache Nodes to your cloud cache cluster to meet your application load. Auto Discovery for Memcached enables automatic discovery of Cache Nodes by ElastiCache Clients when the nodes are added to or removed from an Amazon ElastiCache Cluster.

Reliable

Amazon ElastiCache has multiple features that enhance reliability for critical production deployments, including automatic failure detection and recovery. Amazon ElastiCache runs on the same highly reliable infrastructure used by other Amazon Web Services.

Integrated

Amazon ElastiCache is designed for seamless use with other Amazon Web Services, including Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB, Amazon Elastic Compute Cloud (Amazon EC2), Amazon CloudWatch, and Amazon Simple Notification Service (Amazon SNS).

Secure

Amazon ElastiCache provides a number of mechanisms to secure your Cache Cluster. Amazon ElastiCache provides web service interfaces that allow you to configure firewall settings that control network access to your Cache Cluster.

Amazon ElastiCache allows you to run your Cache Cluster in Amazon Virtual Private Cloud (Amazon VPC). Amazon VPC enables you to isolate your Cache Cluster by specifying the IP ranges you wish to use for your Cache Nodes, and connect to your existing applications inside Amazon VPC.

Cost Effective

Amazon ElastiCache saves you the administrative cost of setting up and managing a multi-node Cache Cluster. You can scale up and scale down the number of Cache Nodes in your Cache Cluster to deliver optimum performance as your application usage pattern changes, paying only for the resources you actually consume. The on-demand pricing allows you to pay for memory/compute capacity by the hour with no long-term commitments. This makes the use of Amazon ElastiCache very cost effective and frees you from the costs and complexities of planning, purchasing, and maintaining hardware.

Multi-AZ

Amazon ElastiCache provides replication features for the Redis engine and Multi-AZ functionality. You can take advantage of multiple AWS Availability Zones to gain availability, and scale beyond the capacity constraints of a single cache node. In case of primary node loss, ElastiCache will automatically detect the failure and failover to a read replica to provide higher availability without the need for manual intervention.

Backup and Restore

Amazon ElastiCache for Redis helps you protect your data by creating snapshots of your clusters. Via a few clicks on the console or simple API calls, you can set up automatic snapshots, as well as initiate a backup whenever you choose. The snapshots can then be used for seeding new ElastiCache for Redis clusters.

Key Use Cases

Amazon ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads (such as social networking, gaming, media sharing and Q&A portals) or compute-intensive workloads (such as a recommendation engine). Caching improves application performance by storing critical pieces of data in memory for low-latency access. Cached information may include the results of I/O-intensive database queries or the results of computationally-intensive calculations.

Chapter 14

An Overview of Amazon's Analytics Service

1. Amazon EMR

Amazon Elastic MapReduce (Amazon EMR) is a web service that makes it easy to quickly and cost-effectively process vast amounts of data. Amazon EMR uses Hadoop, an open source framework, to distribute your data and processing across a resizable cluster of Amazon EC2 instances. Amazon EMR is used in a variety of applications, including log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, and bioinformatics. Customers launch millions of Amazon EMR clusters every year.

How to Use Amazon EMR

To use Amazon EMR, you simply:

Develop your data processing application. You can use Java, Hive (a SQL-like language), Pig (a data processing language), Cascading, Ruby, Perl, Python, R, PHP, C++, or Node.js. Amazon EMR provides code samples and tutorials to get you up and running quickly.

Upload your application and data to Amazon S3. If you have a large amount of data to upload, you may want to consider using AWS Import/Export (to upload data using physical storage devices) or AWS Direct Connect (to establish a dedicated network connection from your data center to AWS). If you prefer, you can also write your data directly to a running cluster.

Configure and launch your cluster. Using the AWS Management Console, EMR's Command Line Interface, SDKs, or APIs, specify the number of EC2 instances to provision

in your cluster, the types of instances to use (standard, high memory, high CPU, high I/O, etc.), the applications to install (Hive, Pig, HBase, etc.), and the location of your application and data. You can use Bootstrap Actions to install additional software or change default settings.

(Optional) Monitor the cluster. You can monitor the cluster's health and progress using the Management Console, Command Line Interface, SDKs, or APIs. EMR integrates with Amazon CloudWatch for monitoring/alarming and supports popular monitoring tools like Ganglia. You can add/remove capacity to the cluster at any time to handle more or less data. For troubleshooting, you can use the console's simple debugging GUI.

Retrieve the output. Retrieve the output from Amazon S3 or HDFS on the cluster. Visualize the data with tools like Tableau and MicroStrategy. Amazon EMR will automatically terminate the cluster when processing is complete. Alternatively you can leave the cluster running and give it more work to do.

2. Amazon Kinesis

Amazon Kinesis is a fully managed service for real-time processing of streaming data at massive scale. Amazon Kinesis can collect and process hundreds of terabytes of data per hour from hundreds of thousands of sources, allowing you to easily write applications that process information in real-time, from sources such as website click streams, marketing and financial information, manufacturing instrumentation and social media, and operational logs and metering data. With Amazon Kinesis applications, you can build real-time dashboards, capture exceptions and generate alerts, drive recommendations, and make other real-time business or operational decisions. You can also easily send data to a variety of other services such as Amazon S3, Amazon DynamoDB, or Amazon Redshift. In a few clicks and a couple of lines of code, you can start building applications which respond to changes in your data stream in seconds, at any scale, while only paying for the resources you use.

3. AWS Data Pipeline

AWS Data Pipeline is a web service that helps you reliably process and move data between different AWS compute and storage services as well as on-premises data sources at specified intervals. With AWS Data Pipeline, you can regularly access your data where it's stored, transform and process it at scale, and efficiently transfer the results to AWS services such as Amazon S3, Amazon RDS, Amazon DynamoDB, and Amazon EMR.

AWS Data Pipeline helps you easily create complex data processing workloads that are fault tolerant, repeatable, and highly available. You don't have to worry about ensuring resource availability, managing inter-task dependencies, retrying transient failures or timeouts in individual tasks, or creating a failure notification system. AWS Data Pipeline also allows you to move and process data that was previously locked up in on-premises data silos.

Chapter 15

A Brief Overview of Amazon's Application Services

1. Amazon SQS

Amazon Simple Queue Service (Amazon SQS) is a fast, reliable, scalable, fully managed message queuing service. Amazon SQS makes it simple and cost-effective to decouple the components of a cloud application. You can use SQS to transmit any volume of data, at any level of throughput, without losing messages or requiring other services to be always available. With Amazon SQS, you can offload the administrative burden of operating and scaling a highly available messaging cluster, while paying a low price for only what you use.

Amazon SWF

Amazon Simple Workflow Service (Amazon SWF) Amazon SWF helps developers build, run, and scale background jobs that have parallel or sequential steps. You can think of SWF as a fully-managed state tracker and task coordinator in the cloud. If your application's steps take more than 500 milliseconds to complete, you need to track the state of processing. If you need to recover, or retry if a task fails, Amazon SWF can help you. Amazon SWF promotes a separation between the control flow of your background job's stepwise logic and the actual units of work that contain your unique business logic. This allows you to separately manage, maintain, and scale the state machinery of your application from the core business logic that differentiates it. As your business requirements change, you can easily change application logic without having to worry about the underlying state machinery, task dispatch, and flow control.

Amazon SWF runs within Amazon's high-availability data centers, so the state tracking and task processing engine is available whenever applications need them. Amazon SWF

redundantly stores the tasks, reliably dispatches them to application components, tracks their progress, and keeps their latest state. Amazon SWF replaces the complexity of custom-coded workflow solutions and process automation software with a fully managed cloud workflow web service. This eliminates the need for developers to manage the infrastructure plumbing of process automation and allows them to focus their energy on the unique functionality of their application. Amazon SWF seamlessly scales with your application's usage. No manual administration of the workflow service is required when you add more cloud workflows to your application or increase the complexity of your workflows. Amazon SWF lets you write your application components and coordination logic in any programming language and run them in the cloud or on-premises.

Amazon AppStream

Amazon AppStream is a flexible, low-latency service that lets you stream resource-intensive applications from the cloud. Amazon AppStream deploys and renders your application on AWS infrastructure and streams the output to mass-market devices, such as personal computers, tablets, and mobile phones. Because your application is running in the cloud, it can scale to handle vast computational and storage needs, regardless of the devices your customers are using. You can choose to stream either all or parts of your application from the cloud. Amazon AppStream enables use cases for applications that wouldn't be possible running natively on mass-market devices. Using Amazon AppStream, your applications are no longer constrained by the hardware in your customers' hands. Amazon AppStream includes a SDK that currently supports streaming applications from Microsoft Windows Server 2008 R2 to devices running FireOS, Android, Chrome, iOS, Mac OS X, and Microsoft Windows.

Amazon SES

Amazon Simple Email Service (Amazon SES) is a cost-effective outbound-only email sending service built on the reliable and scalable infrastructure that Amazon.com has developed to serve its own customer base. With Amazon SES, you can send transactional email, marketing messages, or any other type of high-quality content, and you only pay for what you use. Along with high deliverability, Amazon SES provides easy, real-time access to your sending statistics and built-in notifications for bounces, complaints, and deliveries to help you fine-tune your cloud-based email-sending strategy.

Amazon Elastic Transcoder

Amazon Elastic Transcoder is media transcoding in the cloud. It is designed to be a highly scalable, easy-to-use, and cost-effective way for developers and businesses to convert (or transcode) media files from their source format into versions that will play back on devices like smartphones, tablets, and PCs. Amazon Elastic Transcoder manages all aspects of the transcoding process for you transparently and automatically. There's no need to administer software, scale hardware, tune performance, or otherwise manage transcoding infrastructure. You simply create a transcoding job by specifying the location of your source video and how you want it transcoded. Amazon Elastic Transcoder also provides transcoding presets for popular output formats, which means that you don't need to guess about which settings work

best on particular devices. All these features are available via service APIs and the AWS Management Console.

Amazon Cloud Search

Amazon Cloud Search is a managed service in the AWS cloud that makes it easy to set up, manage, and scale a search solution for your website or application. Amazon CloudSearch supports 34 languages and popular search features such as highlighting, auto complete, and geospatial search.

With Amazon Cloud Search, you can quickly add custom search capabilities to your website or application without having to become a search expert or worry about hardware provisioning, setup, and maintenance. With a few clicks in the AWS Management Console, you can create a search domain and upload the data you want to make searchable, and Amazon Cloud Search automatically provisions the required resources and deploys a highly tuned search index. As your volume of data and traffic fluctuates, Amazon Cloud Search seamlessly scales to meet your needs. You can easily change your search parameters, fine-tune search relevance, and apply new settings at any time without having to re-upload your data.

2. Mobile Services

Amazon SNS

Amazon Simple Notification Service (Amazon SNS) is a fast, flexible, fully managed push messaging service. Amazon SNS makes it simple and cost-effective to push notifications to Apple, Google, Fire OS, and Windows devices, as well as Android devices in China with Baidu Cloud Push. You can also use Amazon SNS to push notifications to Internet-connected smart devices, as well as other distributed services. Besides pushing cloud notifications directly to mobile devices, Amazon SNS can also deliver notifications by SMS text message or email to Amazon SQS queues, or to any HTTP endpoint. To prevent messages from being lost, all messages published to Amazon SNS are stored redundantly across multiple Availability Zones.

Amazon Cognito

Amazon Cognito is a service that makes it easy to save user data, such as app preferences or game state, in the AWS cloud without writing any backend code or managing any infrastructure. You can save data locally on users' devices allowing your applications to work even when the devices are offline. You can also synchronize data across a user's devices so that their app experience will be consistent regardless of the device they use. With Amazon Cognito, you can focus on creating great app experiences instead of having to worry about building and managing a backend solution to handle identity management, network state, storage, and sync.

Amazon Mobile Analytics

Amazon Mobile Analytics is a service that lets you easily collect, visualize, and understand app usage data at scale. Many mobile app analytics solutions deliver usage data several hours after the events occur. Amazon Mobile Analytics is designed to deliver usage reports within 60 minutes of receiving data from an app so that you can act on the data more quickly. Amazon Mobile Analytics is built to scale with your app, allowing you to collect and process billions of events per day from millions of users. It's easy to get started with Amazon Mobile Analytics. You simply add the AWS Mobile SDK to your app and publish the app using your existing distribution mechanism (such as the iTunes Store, Google Play, or Amazon Appstore), and you can start accessing reports in the AWS Management Console. Amazon Mobile Analytics automatically starts to collect metrics on active users, sessions, and retention, and you can add reporting on in-app revenue or any custom event you choose. Amazon Mobile Analytics helps you spend more time on creating great apps rather than the undifferentiated heavy lifting of setting up and managing an analytics system. Amazon Mobile Analytics is included in the AWS Mobile SDK, which supports iOS, Android, and Fire OS, or you can use the Amazon Mobile Analytics REST API directly.

AWS Mobile SDK

The AWS Mobile SDK helps you build high quality mobile apps quickly and easily. It provides access to AWS Mobile services, mobile-optimized connectors to popular AWS data and storage services, and easy access to a broad array of other AWS services. The AWS Mobile SDK includes libraries, code samples, and documentation for iOS, Android, and Fire OS so you can build apps that deliver great experiences across devices and platforms.

3. Enterprise Applications

Amazon WorkSpaces

Amazon WorkSpaces is a managed desktop computing service in the cloud. Amazon WorkSpaces allows customers to easily provision cloud-based desktops that allow end-users to access the documents, applications and resources they need with the device of their choice, including laptops, iPad, Kindle Fire, Android tablets, and zero clients. With a few clicks in the AWS Management Console, customers can provision a high-quality cloud desktop experience for any number of users at a cost that is highly competitive with traditional desktops and half the cost of most virtual desktop infrastructure (VDI) solutions.

Amazon Zocalo

Amazon Zocalo is a fully managed, secure enterprise storage and sharing service with strong administrative controls and feedback capabilities that improve user productivity. Users can comment on files, send them to others for feedback, and upload new versions without having to resort to emailing multiple versions of their files as attachments. Users can take advantage of these capabilities wherever they are, using the device of their choice, including PCs, Macs, and tablets. Amazon Zocalo offers IT administrators the option of integrating with existing corporate directories, flexible sharing policies, audit logs, and control of the location where data is stored.

References

1. Wang, L., Pai, V., Peterson, L.: The effectiveness of request redirection on CDN robustness.
web.cs.wpi.edu/*rek/DCS/D04/CDN_Redirection.ppt. Accessed 09,October 2014.
2. D. C. Verma. Content Distribution Networks: An engineering approach. Wiley Inter-Science, 2002.
3. C. M. Chen, Y. Ling, M. Pang, W. Chen, S. Cai, Y. Suwa, O. Altintas, “Scalable Request-Routing with Next- Neighbor Load Sharing in Multi-Server Environments,” In Proceedings of the 19th International Conference on Advanced Information Networking and Applications, IEEE Computer Society, Washington, DC, USA, pp. 441-446, 2005.
4. S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. Van Steen, “Replication of Web Hosting Systems,” ACM Computing Surveys, Vol. 36, No. 3, ACM Press, NY, USA, 2004.
5. F. Douglis, and M. F. Kaashoek, “Scalable Internet Services,” IEEE Internet Computing, Vol. 5, No. 4, 2001, pp.36-37.
6. Akamai Technologies Inc., “Akamai-The Business Internet - A Predictable Platform for Profitable E-Business,” 2004.
7. R. Brussee, H. Eertink, W. Huijsen, B. Hulsebosch, M. Rougoor, W. Teeuw, M. Wibbels, and H. Zandbelt, “Content Distribution Network State of the Art,” Telematica Instituut, June 2001.
8. Buyya, Rajkumar, Al-Mukaddim Pathan, James Broberg, and Zahir Tari. "A Case for Peering of Content Delivery Networks." IEEE Distributed Systems Online IEEE Distrib. Syst. Online 7.10 (2006).
9. Website: <http://www.sitepoint.com/7-reasons-to-use-a-cdn/>.
10. Pallis, G., Vakali, A.: Insight and perspectives for content delivery networks. Commun. ACM 49(1), 101–106 (2006).
11. Pathan, A.K., Buyya, R.: A taxonomy and survey of content delivery networks. Technical Report, GRIDS-TR-2007-4, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 12 Feb 2007
12. Li, L., Ma, X., Huang, Y.: CDN cloud: A novel scheme for combining CDN and cloud computing. In: 2013 International Conference on Measurement, Information and Control (ICMIC), vol. 01, pp. 687, 690, 16–18 August 2013
13. Akamai Technologies, Inc. (2014). www.akamai.com

14. A case for the Accountable Cloud. <http://aws.amazon.com/cn/cloudfront/>
15. Krauter, K., Buyya, R., et al.: A taxonomy and survey of grid resource management systems for distributed computing. *Softw. Pract. Exp. (SPE)* 32, 135–164 (2002)
16. Microsoft Azure Cloud Services. <http://www.windows.azure.com>. Accessed 09 October 2014
17. Rackspace Cloud Files. <http://www.rackspace.com.au/cloud/files>. Accessed 06 August 2014
18. Amazon CloudFront. <http://aws.amazon.com/cloudfront/>. Accessed 09 October 2014
19. Broberg, J., Buyya, R., Tari, Z.: MetaCDN: Harnessing ‘Storage Clouds’ for highBperformance content delivery. *J. Netw. Comput. Appl.* 32(5), 1012–1022 (2009). ISSN: 1084-8045. <http://dx.doi.org/10.1016/j.jnca.2009.03.004>
20. NoSQL. <http://nosql-database.org/>. Accessed 06 August 2014
21. Limelight Orchestrate CDN. <http://www.limelight.com/services/orchestrate-content-delivery>. Html. Accessed 06 August 2014
22. Georgakopoulos, D., Ranjan, R., Mitra, K., Zhou, X.: MediaWise - Designing a smart media cloud. In: *Proceedings of the International Conference on Advances in Cloud Computing (ACC 2012)*, Bangalore, India, 26–28 July (2012). <http://arxiv.org/ftp/arxiv/papers/1206/1206.1943.pdf>
23. Ranjan, R., Mitra, K., Georgakopoulos, D.: MediaWise cloud content orchestrator. *J. Internet Serv. Appl.* 4, 2 (2013)
24. Mell, P., Grance, T.: The NIST definition of cloud computing (draft). NIST special publication, vol. 800, p. 145 (2011)
25. Gong, C., Liu, J., Zhang, Q., Chen, H., Gong, Z.: The characteristics of cloud computing. In: *2010 39th International Conference on Parallel Processing Workshops (ICPPW)*, pp. 275–279 (2010)
26. Cisco: Cisco Visual Networking Index: Forecast and Methodology, 2013–2018. http://www.isco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html. Accessed 09 October 2014
27. IDC: Data Growth, Business Opportunities, and IT Imperatives. <http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm>. Accessed 09 October 2014
28. Akamai: The Importance of Delivering A Great Online Video Experience. http://www.akamai.com/dl/reports/jupiter_onlinevideoexp.pdf. Accessed: 09 October 2014
29. IneoQuest: The Case for Leveraging the Cloud for Video Service Assurance. http://www.ineoquest.com/wp-content/uploads/2013/10/Whitepaper_Cloud_Services.pdf. Accessed 09 October 2014

30. BigData & CDN: <http://www.slideshare.net/pavlobaron/bigdata-cdnoop2011-pavlobaron>. Accessed 07 March 2014
31. Yin, H., Liu, X., Min, G., Lin, C.: Content delivery networks: A bridge between emerging applications and future IP networks. *Network, IEEE*, 24(4), 52, 56, July–August 2010
32. Wang, L., Kunze, M., Tao, J., Laszewski, G.: Towards building a cloud for scientific applications. *Adv. Eng. Softw.* 42(9), 714–722 (2011)
33. Wang, L., Fu, C.: Research advances in modern cyber infrastructure. *New GenerComput.* 28 (2), 111–112 (2010)
34. Mastin, P.: Is the cloud a CDN killer. <http://cloudcomputing.sys-con.com/node/2628667>. Accessed 07 March 2014
35. Compton, K.: Marching towards cloud CDN. <http://blogs.cisco.com/sp/marching-towardscloud-cdn/>. Accessed 07 March 2014
36. Peng, G.: CDN: Content distribution network. Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY, 2003. <http://citeseer.ist.psu.edu/peng03cdn.html>
37. Vakali, A., Pallis, G.: Content delivery networks: Status and trends. *IEEE Internet Comput.*, 68–74. IEEE Computer Society, November–December 2003
38. Dille, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., Wehl, B.: Globally distributed content delivery. *IEEE Internet Comput.* 6(5), 50–58 (2002)
39. Kung, H.T., Wu, C.H.: Content networks: Taxonomy and new approaches. In: Park, K., Willinger, W. (eds.) *The Internet as a Large-Scale Complex System*. Oxford University Press (2002)
40. Vakali, A. and Pallis, G. Content delivery networks: Status and trends. *IEEE Internet Computing* 7, 6 (2003), 68–74.
42. Content Delivery Networks (CDNs) – A Reference Guide, Matthew Liste from Cisco.
- 43.M. Pathan. Content distribution networks (cdn) - research directory, May 2007. URL <http://www.cs.mu.oz.au/~apathan/CDNs.html>.
- [4] Jaison Paul Mulerikkal. An Architecture for Distributed Content Delivery Network. MSC thesis submitted to the Science, Engineering, and Technology Portfolio, in the Royal Melbourne Institute of Technology. Melbourne, Victoria, Australia. 2007.
44. Mao, Z.M., Cranor, C.D., Boughs, F., Rabinovich, M., Spatscheck, O., Wang, J.: A precise and efficient evaluation of the proximity between web clients and their local DNS servers. In: *Proceedings of the USENIX 2002 Annual Technical Conference*, Monterey, CA, USA, pp. 229–242, June 2002

45. Armbrust, M., et al.: A view of cloud computing. *Commun. ACM Mag.* 53(4), 50–58 (2010). doi:10.1145/1721654.1721672. ACM Press.
46. Patterson, D.A.: Technical perspective: The data center is the computer. *Commun. ACM Mag.* 51(1), 105–105 (2008). ACM Press
47. Hui, L.: Realistic workload modeling and its performance impacts in large-scale eScience grids. *IEEE Trans. Parallel Distrib. Syst.* 21, 480–493 (2010)
48. Wang, C., Ranjan, R., Zhou, X., Mitra, K., Saha, S., Meng, M., Georgakopoulos, D., Wang, L., Thew, P.: A cloud-based collaborative video story authoring and sharing platform. *CSI J. Comput.* 1(3), 66–76 (2012)
49. Ranjan, R., Mitra, K., Saha, S., Georgakopoulos, D., Zaslavsky, A.: Do-it-yourself content delivery network orchestrator. In: Wang, X., Cruz, I., Delis, A., Huang, G. (eds.) *WISE 2012. LNCS*, vol. 7651, pp. 789–791. Springer, Heidelberg (2012)
50. Zhang, X., Du, H., Chen, J.-Q., Lin, Y., Zeng, L.-J.: Ensure data security in cloud storage. In: *2011 International Conference on Network Computing and Information Security (NCIS)*, vol. 1, pp. 284, 287, 14–15 May 2011
51. Stefan S, Krishna P. G., Richard J. D., Steven D. G., and Henry M. L. *An Analysis of Internet Content Delivery Systems*. University of Washington. 2004.
52. N. Bartolini, E. Casalicchio, and S. Tucci, “A Walk Through Content Delivery Networks,” In *Proceedings of MASCOTS 2003, LNCS Vol. 2965/2004*, pp. 1-25, April 2004.
53. Pathan, Mukaddim, and Rajkumar Buyya. "A Taxonomy of CDNs." *Content Delivery Networks Lecture Notes Electrical Engineering*: 33-77.
54. Wen, Y., Zhu, X., Rodrigues, J.J.P.C., Chen, C.W.: Cloud mobile media: Reflections and outlook. *IEEE Trans. Multimedia* 16(4), 885, 902, June 2014
55. Shaikh, A., Tewari, R., Agrawal, M.: On the effectiveness of DNS-based server selection. In: *Proceedings of IEEE INFOCOM, Anchorage, AK, USA*, pp. 1801–1810, April 2001 *An Overview of Cloud Based Content Delivery Networks: Research Dimensions* 157
56. Russo, W., Mastroianni, C., Palau, C.E., Fortino, G.: CDN-Supported collaborative media streaming control. *IEEE MultiMedia* 14(2), 60–71 (2007)
57. Jin, Y., Wen, Y., Shi, G., Wang, G., Vasilakos, AV.: CoDaaS: An experimental cloud-centric content delivery platform for user-generated contents. In: *2012 International Conference on Computing, Networking and Communications (ICNC)*, pp. 934, 938, 30 Jan – 2 Feb 2012
58. Li, H., Zhong, L., Liu, J., Li, B., Xu, B.: Cost-effective partial migration of VoD services to content clouds. In: *2011 IEEE International Conference on Cloud Computing (CLOUD)*, pp. 203, 210, 4–9 July 2011

59. Wang, L., Chen, D., Zhao, J., Tao, J.: Resource management of distributed virtual machines. *IJAHUC* 10(2), 96–111 (2012)
60. Wang, L., Jie, W.: Towards supporting multiple virtual private computing environments on computational Grids. *Adv. Eng. Softw.* 40(4), 239–245 (2009)
61. Wang, L., von Laszewski, G., Younge, A.J., He, X., Kunze, M., Tao, J., Cheng, F.: Cloud computing: A perspective study. *New Generation Comput.* 28(2), 137–146 (2010)
62. Wang, L., Chen, D., Yangyang, H., Ma, Y., Wang, J.: Towards enabling cyberinfrastructure as a service in clouds. *Comput. Electr. Eng.* 39(1), 3–14 (2013)
63. M. Dodani, "The Silver Lining of Cloud Computing", *J. Object Technology*, vol. 8(2), pp 29-38, 2009. [4] P. Mell and T. Grance, —Cloud Computing Definition‡, National Institute of Standards and Technology, Version 15, 10-7-09.
64. Craig Balding, —Assessing the Security Benefits of Cloud Computing‡, <http://cloudsecurity.org/blog/2008/07/21/assessing-the-security-benefits-of-cloud-computing.html>
65. Sujay. R —Hybrid Cloud: A New Era‡, *International Journal of Computer Science and Technology*, ISSN: 2229-4333(Print) ISSN :0976 -8491 (Online), June 2011. [7] Rupinder Kaur, —Cloud computing‡, *International Journal of Computer Science and Technology*, *IJCST* Vol. 2, Issue 3, September 2011.

Some Important Question and Answer:

What Is a Content Delivery Network?

Content delivery networks use multiple servers in many geographic locations that improve deliveries of static and streaming content. Global content requests automatically get routed to the closest servers, speeding up page loads, maximizing bandwidth and providing identical content regardless of Internet- or site-traffic spikes. Depending on traffic and number of nodes, the network's algorithms select the best routing options to deliver optimum performance and avoid bottlenecks.

Users with high-speed connections often experience choppiness, loading lags and poor quality, especially when viewing live events or if they are located far from the hosting servers. CDNs minimize latency issues that cause image jitters, optimize delivery speeds and maximize available bandwidth for each viewer.

Advantages of Using CDN:

- ✓ CDN provides content with high performance & high availability.
- ✓ Content delivery at high speed.
- ✓ Not only provides content of content provider but also provides them some degree of protection
- ✓ Reduce bandwidth cost
- ✓ Decrease page load time i.e. loads data faster
- ✓ Increase global availability of the content
- ✓ From any geographical location user can get content with the high speed from nearby server.
- ✓ Provides backup services online.
- ✓ Performance booster and money saver
- ✓

Disadvantages of CDNs

Disadvantages include additional costs, but large companies often save money by delivering their content along shorter routes. Some drawbacks include the following trouble spots:

New Points of Failure: CDNs create new points of potential failures along the delivery chains.

Sharing Resources: Delivery networks have many clients, and response times could vary due to the volume of website traffic of other CDN customers.

Geographical Choice Considerations: Website owners need to research their clients and choose the CDNs that offer the most convenient server locations near where they get the most business. If a website gets most of its customers from North America, then server locations in Europe and Asia would offer fewer benefits.

Content Management Problems: Companies might need to consider using content management systems to manage and update content throughout the networks. Some customers might need special widgets, content players or specific applications. Customer interactions could have different requirements in certain regions.

Lack of Direct Control: Changes to content must be made through CDN providers instead of directly, which could pose problems for editors and developers.

Samples of Popular Content Delivery Networks

Free content delivery networks include Coral Content Distribution Network, FreeCast, CloudFare and Incapsula. Popular commercial content delivery networks include the following companies:

Akamai
Amazon CloudFront
BitGravity
CacheFly
CDNetworks
CloudFlare
EdgeCast
EdgeStream
Limelight Networks
LocalMirror
MaxCDN
Mirror Image
Panther Express

Choosing a Content Delivery Network

Websites owners face questions about which type of network to choose. Customers can choose from Internet-based or private networks, and some networks integrate with Online Video Platform, enabling features that enhance video distribution. Other factors customers should study include the following:

High definition video has become increasingly popular, but not all networks support it.

Transcoding changes code to make videos compatible on certain players like Silverlight, Flash, Quicktime and iDevice. Clients need to choose compatible players for their content.

Mobile delivery has become very important to consumers, so website owners need to know if their network supports mobile delivery.

Licensed or protected material, such as Pay-Per-View, requires Digital Rights Management.

Analytics capabilities of content delivery networks become important in real-time for many applications, so owners need to assess potential networks for their analytic and reporting abilities.

What is the purpose of a CDN?

Quickly deliver websites and applications to end users anywhere in the world in a secure and reliable environment. The less time it takes to transmit web content to the end user, the faster websites load/respond. In addition to accelerating content delivery, CDNs also serve as a fail-safe system to ensure the site is always available during traffic peaks and server outages.

How does a CDN work to solve web performance issues?

Providing an interconnected network tuned for speed, reliability and scalability. CDNs enable web entities to bypass the inefficiencies of the standard internet, ensuring faster and more reliable web content delivery. Site owners can rely on CDNs to efficiently balance traffic

loads, dynamically scale to accommodate any traffic surges, and even detect and mitigate security threats.

Why should I use a CDN and not multiple datacenters?

The primary reasons are it is practical and it saves you cost and time. Managing multiple, globally distributed datacenters can be extremely expensive and resource intensive, requiring fixed up-front costs and a multitude of ongoing operating costs. In addition to cost, it can take months or even up to years to build, set up and deploy a fully functional network of datacenters.

CDNs provide a quick deployment strategy and eliminate huge investments associated with building and maintaining an internal infrastructure. Most organizations will see positive net results in a few months, versus a typical two year period before seeing any returns on deploying an internal solution.

Can CDN's accelerate dynamic content/applications?

Yes, but not all CDNs are capable of accelerating dynamic content. Most CDNs only have the ability to cache (duplicate) content throughout their network for quicker delivery. However, dynamic content such as games, applications, online trading transaction any data that is generated in real-time cannot be cached and must be delivered directly from the origin server. Dynamic acceleration is typically only provided by tier-one CDNs configured to streamline data requests and delivery between the origin server and the end user. Dynamic acceleration enables data from the origin server to bypass standard internet nodes and be delivered through a more direct and efficient path, drastically reducing load time and improving responsiveness.

Are CDNs valuable for video streaming?

Yes, CDN video streaming can accelerate the delivery of media files in a more secure and reliable environment. In addition to lag free viewing experiences, CDNs also ensure the asset's availability at all times, even during huge demand spikes.

Can CDNs use SPDY?

Google's SPDY is a protocol for transporting web content, designed specifically to address latency. Early tests found that SPDY delivered load time improvements ranging between 27% and 60% over HTTP.

SPDY has not been widely adopted by the content delivery industry. In a recent podcast interview, Google's Ilya Grigorik discussed why widespread CDN adoption of SPDY would be a huge boon, not just for SPDY, but for optimization best practices in general, and ultimately for end users.

How Does CDN Works ?

CDN stores and provides facility of delivering **content** to end –users and for that CDN operators are paid by the “**Content Providers**” for delivering the content. Besides this, CDN has to pay to **ISP : Internet Service Providers**, to network operators as they allows hosting of servers in their multiple data centers.

Multiple copies of the content are available on different server (aka **Content Replication**)

and CDN provides an identical content at a high speed. When any user make a request for a specific page, content or file, the closest server will respond and delivers the identical copy of content i.e. users reside at different location can access the same content but from different servers. CDN acts as an service provider on the internet and it is a Performance Booster.

Type of CDN:

Content can be static or Dynamic so CDN can be of two type first CDN that deals with **static content** and another that deals with **dynamic content**.

Some websites use CDN for hosting images and static **CSS** and **JavaScript** files, some uses CDN for streaming **HD** (High Definition) videos or HD content while some (Websites having large databases) uses CDN for long term storage. This is less expensive as well as needs less bandwidth.

Another type of CDN is which deals with Dynamic content or we can say “Web Cache”. For example CMS: Content Management System.

Content Routing:

Content routing redirect the user to best device in network based upon the well define user polices. Content routing can deliver the content quickly as possible using high availability techniques and fast server responses. It is network load balancing technology, content router perform redirects, load balancing on multiple remote user’s request, so that the user requests for content sources get the fastest response.

Content Caching:

Using content caching you can cache the frequently access content. Cisco has “Cisco content Engine” for providing content caching. You can place the Cisco content Engine (CCE) on suitable location for content caching, for example if you have video streaming users in your network you can use Cisco Content Engine for those users for providing them quick services. Caching of Web content gives you the flexibility of serving documents locally from local cache server. The content caching feature allows you to store commonly accessed Web content on local server and users can get much faster response from that local server.

Content Switching:

Content switching is used for web internet, internet delivery and for e-commerce modules. You can perform the content switching with Cisco Content Distributor.

Content switching is based on content availability and server availability you can perform the content switching by POP cache server, using application layer switching, and flow splitting using different load balancing techniques.

Using content switching you can configure the applications to redirect requests to different servers with different content on the basis of different clients attributes.

You can redirect the different traffic according to nature of request for example you can send the HTTP traffic to a specific Web server. If the request came from an IP phone, you can direct to a server or call manager that is capable of handling content that the user can view on his phone. A computer’s request is directed to a different server that is capable of handling content related to computer.

For web traffic in different languages you can sends the request to different servers that serves content in that language. For example if a user has configured his browser to request

the content in Spanish you can send this user to server that can handle the Spanish language request.

Is your content mostly dynamic or static?

Your needs for a CDN may vary based on the type of files that you need to host. Some files may be better left in house, like your web pages themselves or any files that get updated frequently.

Conversely, CDNs are great for files like images or style sheets that can stay static without many changes. A CDN can get these files closer to your end user so that they can load faster.

If you do choose to host most of your files through a CDN, understand that your content will remain cached. If you have a lot of dynamic content or highly personalized content, find a CDN that is able to quickly adapt to any changes you need.

Can CDN stop DDoS?

DDoS attacks are one of the biggest threats to the security on the internet, since the users of controlled computer systems are usually not aware of the attack performed. And since packets are not coming from a single source, they cannot be stopped by simply blocking a single IP address. Not that using one method of prevention alone can actually protect you. However, by using a combination of a few, you have a better chance of defending your business space.

Content delivery networks (CDNs) use servers located at different data centers. Not only one, but many communication channels are used. Since the emergence of cloud computing, CDNs are employed not only as a tool for unclogging the internet, but also as a tool for mitigating (avoiding) DDoS attacks. CDNs will absorb less-sophisticated DDoS attacks, simply with bandwidth. With CDNs, you gain the advantage of – size.

How can you see if CDN is really that fast in specific location?

To measure CDN performance several methods can be used to get analysis results.

Server side performance monitoring: simulates end user data requests, and then measures how quickly a webpage responds to the request

Synthetic transaction monitoring: incorporates emulators and real world browsers to test predefined data requests from many different locations

Measuring the performance of a few users: taking a selection of your end users and measuring how long it takes for them to access your pages

Measuring the performance of every end user: the most accurate method, by measuring all actual transaction times across a network and from all users, you can record response times from a server, network and application perspective

Number of POPs: the more points of presence a content delivery network provider has, the more bandwidth and customers it can potentially handle

What is a difference between CDN, dynamic site acceleration, video delivery, mobile delivery, multi-CDN, cloud?

A **content delivery network (CDN)** is a system surrogate web servers distributed across different data centers in different geographical regions to deliver content to end-users, based on proximity of this user.

The dynamic content of the World Wide Web 2.0 needs to get out there faster, and remain

personalized. Personalized content as well as real-time info cannot be cached. Moreover, the emergence of cloud layers such as SaaS (Software as a Service), and PaaS (Platform as a Service) and so on, require more efficient models of delivering online content. Besides caching. This is why Content Delivery Networks (CDNs) include Dynamic Site Acceleration services.

Sometimes, using more than one CDN provider is advised because CDN providers perform differently in different geographical regions, and not all CDN providers are reliable as others. This is known as multiple content delivery network or multiple CDN, multi-CDN.

Content Delivery Network platforms for the delivery of video and multimedia content acceleration, are on the rise. The new emerging technologies are busily developed and discussed among all the stakeholders – video content owners, service providers, telecom carriers and industry vendors. All are struggling for a flawless delivery of clips and video streams all over the world, at any time, as the user becomes more impatient, more mobile, more critical of services provided, and more picky.

Finally, mobile is the defining change in industries, building websites with a responsive design. The shift towards a mobile and app-based internet is rapidly changing the role of the CDN. Today, content delivery network for mobile delivery must be highly customized to further maximize the end user experience.

The term “cloud” stands for a set of remote computers that run the applications, store the data, and use a server system, and are owned by companies. The real storage and the software thus does not exist on your computer, but exist instead – on the cloud.

The most popular services of the cloud are that of infrastructure, platform, software, or storage and are largely improved by CDNs.

Does CDN solve security issues?

Since CDNs use surrogate servers located across different data centers in different regions around the globe, this scattered infrastructure provides a more secure network.

CDNs are known to absorb less-sophisticated DDoS attacks, simply with bandwidth. With CDNs, you gain the advantage of – size. The overload caused by DDoS attack is processed on different PoPs according to their origin, which helps to prevent server saturation.

Many CDN providers will also block threats and limit abusive bots and crawlers from wasting your bandwidth and server resources. This will result in the decrease of spam and hack attacks. Again, this depends of a service that your CDN provider is offering.

Is CDN a cloud?

The development of CDN networks sought to deal with extreme bandwidth pressures, first as video streaming was growing in demand as well as the number of content providers. That was in the past. Now, CDNs are a continual trend, with the emergence of cloud computing, improving all the layers of cloud computing:

SaaS (Software as a service), e.g. Google Docs

IaaS (Infrastructure as a service), e.g. Amazon

PaaS (Platform as a service) e.g. Google App Engine

BPaaS (Business Process as a service) e.g. advertisements, payments

However, CDNs are not “the cloud” as this is the umbrella term for all the remote computers owned by companies that run applications, store data, offer platforms etc. CDNs are content delivery networks, a worldwide network that stores static files such as video, pictures and pages that don’t change. It’s faster to load such files from somewhere locally which causes the page load improvement.

CDN would require actual server to be placed on different places around the world, and cloud computing would require large amount of hardware resource to provide as service.

What is POP?

POPs (also, points of presence, edge caches) are distributed servers where static resources are stashed across a region or worldwide, thereby bringing resources closer to users and reducing round trip time.

What kind of problems CDN solve

Implementing a CDN affects everything, from your internal architecture to the cost of your IT staff, performance management and more. True power of a content delivery network is yet to be revealed in future decades, as it soars by 20% each year, but so far, the major advantages of using one are:

- ✓ significantly reduced page load time of your website
- ✓ increased revenue by 1% for every 100 ms of improvement to your page load time
- ✓ retaining more customers (they are more satisfied)
- ✓ more manageable traffic
- ✓ maximum availability of your product
- ✓ more secure network
- ✓ no geographical barriers
- ✓ easy delivery of video, audio rich content
- ✓ build more interactive website at no cost of losing visitors due to latencies
- ✓ reaching mobile customers with ease
- ✓ branching out to new markets, regions
- ✓ easy management of traffic peaks
- ✓ more scalability to your business, you can grow it as much as you want to
- ✓ less or now downtimes
- ✓ setting your own criteria to enable the best possible performance for your website

What kind of hardware does the CDN use?

A CDN using faster, more powerful hardware to build their network can offer your company better page load speeds and better site performance. For instance, look for CDNs that use solid-state drives instead of traditional spinning drives. With solid-state drives, a CDN can fetch cached objects faster and handle more requests per second, which means users get your content more quickly.

What is Amazon EC2 service?

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable (scalable) computing capacity in the cloud. You can use Amazon EC2 to launch as many virtual servers you need. In Amazon EC2 you can configure security and networking, and manage storage.

What are the features of Amazon EC2 service?

As Amazon EC2 service is a cloud service so it has all the cloud features. Amazon EC2 provides the following features:

Virtual computing environment (known as instances)

Pre-configured templates for your instances (known as Amazon Machine Images – AMIs)
Amazon Machine Images (AMIs) is package that you need for your server (including the operating system and additional software)

Amazon EC2 provides various configuration of CPU, memory, storage and networking capacity for your instances (known as instance type)

Secure login information for your instances using key pairs (AWS stores the public key and you store the private key in a secure place)

Storage volumes for temporary data that's deleted when you stop or terminate your instance (known as instance store volumes)

Amazon EC2 provides persistent storage volumes (using Amazon Elastic Block Store – EBS)
A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups

Static IP addresses for dynamic cloud computing (known as Elastic IP address)

Amazon EC2 provides metadata (known as tags)

Amazon EC2 provides virtual networks that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network (known as virtual private clouds – VPCs)

What is Amazon Machine Image and what is the relation between Instance and AMI?

Amazon Web Services provides several ways to access Amazon EC2, like web-based interface, AWS Command Line Interface (CLI) and Amazon Tools for Windows Powershell. First you need to signed up for an AWS account and you can access Amazon EC2.

Amazon EC2 provides a Query API. These requests are HTTP or HTTPS requests that use the HTTP verbs GET or POST and a Query parameter named Action.

What is Amazon Machine Image (AMI)?

An Amazon Machine Image (AMI) is a template that contains a software configuration (for example, an operating system, an application server, and applications). From an AMI, we launch an instance, which is a copy of the AMI running as a virtual server in the cloud. We can launch multiple instances of an AMI.

What is the relation between Instance and AMI?

We can launch different types of instances from a single AMI. An instance type essentially determines the hardware of the host computer used for your instance. Each instance type offers different compute and memory capabilities.

After we launch an instance, it looks like a traditional host, and we can interact with it as we

would any computer. We have complete control of our instances; we can use sudo to run commands that require root privileges.

Explain storage for Amazon EC2 instance.

Amazon EC2 provides many data storage options for your instances. Each option has a unique combination of performance and durability. These storage can be used independently or in combination to suit your requirements.

There are mainly four types of storage provided by AWS.

Amazon EBS: Its durable, block-level storage volumes that you can attach to a running Amazon EC2 instance. The Amazon EBS volume persists independently from the running life of an Amazon EC2 instance. After an EBS volume is attached to an instance, you can use it like any other physical hard drive. Amazon EBS encryption feature supports encryption feature.

Amazon EC2 Instance Store: Storage disk that is attached to the host computer is referred to as instance store. Instance storage provides temporary block-level storage for Amazon EC2 instances. The data on an instance store volume persists only during the life of the associated Amazon EC2 instance; if you stop or terminate an instance, any data on instance store volumes is lost.

Amazon S3: Amazon S3 provides access to reliable and inexpensive data storage infrastructure. It is designed to make web-scale computing easier by enabling you to store and retrieve any amount of data, at any time, from within Amazon EC2 or anywhere on the web.

Adding Storage: Every time you launch an instance from an AMI, a root storage device is created for that instance. The root storage device contains all the information necessary to boot the instance. You can specify storage volumes in addition to the root device volume when you create an AMI or launch an instance using block device mapping.

What are the Security Best Practices for Amazon EC2?

There are several best practices for secure Amazon EC2. Following are few of them. Use AWS Identity and Access Management (IAM) to control access to your AWS resources. Restrict access by only allowing trusted hosts or networks to access ports on your instance. Review the rules in your security groups regularly, and ensure that you apply the principle of least

Privilege — only open up permissions that you require.

Disable password-based logins for instances launched from your AMI. Passwords can be found or cracked, and are a security risk.

Explain Stopping, Starting, and Terminating an Amazon EC2 instance?

Stopping and Starting an instance: When an instance is stopped, the instance performs a normal shutdown and then transitions to a stopped state. All of its Amazon EBS volumes remain attached, and you can start the instance again at a later time. You are not charged for additional instance hours while the instance is in a stopped state.

Terminating an instance: When an instance is terminated, the instance performs a normal shutdown, then the attached Amazon EBS volumes are deleted unless the volume's delete On

Termination attribute is set to false. The instance itself is also deleted, and you can't start the instance again at a later time.

Lab 1: Introduction to Amazon Elastic Compute Cloud (EC2)




Lab Details:

- 1.This lab walks you through the steps to launch and configure a virtual machine in the Amazon cloud.
- 2.You will practice using Amazon Machine Images to launch Amazon EC2 Instances and use key pairs for SSH authentication to log into your instance. You will create a web page and publish it.
- 3.Duration: 00:30:00 Hrs
- 4.AWS Region: US East (N. Virginia)

Tasks:

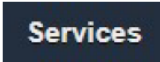

- 1.Login to AWS Management Console.
- 2.Create an Amazon Linux Instance from an Amazon Linux AMI
- 3.Find your instance in the AWS Management Console.
- 4.SSH into your instance.
- 5.Install a Web server on the server
- 6.Create and publish a sample test.html file.
- 7.Test the page with public IP address of EC2 Instance created.




Launching Lab Environment

- 1.Launch lab environment by clicking on . This will create an AWS environment with the resources required for this lab.
- 2.Once your lab environment is created successfully,  will be active. Click on , this will open your **AWS Console Account** for this lab in a new tab.
- 3.If you face any issues, please go through **FAQs and Troubleshooting for Labs**.

Steps:

Launching an EC2 Instance

- 1.Navigate to EC2 by clicking on the  menu in the top, then click on **EC2** in the  **Compute** section.

2. Click on 
3. Choose an Amazon Machine Image (AMI):  Amazon Linux **Amazon Linux 2 AMI (HVM), Arm)**
4. Choose an Instance Type: select  **Free tier eligible** and then click on  the
5. **Configure Instance Details:** No need to change anything in this step, click on 
6. **Add Storage:** No need to change anything in this step, click on 
7. **Add Tags:** Click on 
 - Key: **Name**
 - Value: **MyEC2Server**
 - Click on 
8. **Configure Security Group:**
 - To add **SSH**,
 - Choose Type: 
 - Source: Custom (Allow specific IP address) or Anywhere (From ALL IP addresses accessible).
 - For **HTTP**,
 - Click on 
 - Choose Type: **HTTP**
 - Source:  (Allow specific IP address) or  (From ALL IP addresses accessible).
 - For **HTTPS**,
 - Click on 
 - Choose Type: **HTTPS**

- Source: (Allow specific IP address) or (From ALL IP addresses accessible).

Review and Launch

- After that click on

Launch

9.**Review and Launch** : Review all settings and click on

10.**Key Pair** : This step is most important, Create a new key Pair and click

Download Key Pair

Launch Instances

on after that click on

11.**Launch Status**: Your instance is now launching, Click on the instance ID and wait for

complete initialization of instance till status change to **running**

Name	Instance ID	Instance Type	Availability Zone	Instance State
	i-01f66d9a39c8f1699	t2.micro	us-east-1a	running

12.Note down the sample IPv4 Public IP Address of the EC2 instance. A sample is shown in below screenshot.

?

Instance: **i-01f66d9a39c8f1699** Public DNS: **ec2-107-21-198-65.compute-1.amazonaws.com**

Description | Status Checks | Monitoring | Tags

Instance ID	i-01f66d9a39c8f1699	Public DNS (IPv4)	ec2-107-21-198-65
Instance state	running	IPv4 Public IP	107.21.198.65
Instance type	t2.micro	IPv6 IPs	-


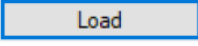
SSH into EC2 Instance

1.For MAC Users:

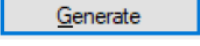
- Open Terminal
- Enter the following command:
 - Syntax: **ssh -i ec2-user@publicIPAddress keypairname.pem**
 - Sample: **ssh -i ec2-user@107.21.198.65 keypairname.pem --> Click Enter**
 - Type **yes** and enter, you will be successfully logged into EC2 Instance.

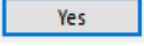
2.For Windows Users:

- Download **putty** and **puttygen** from this link : <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>


•To convert your key pair **.pem** to **.ppk**, Open  PuTTYgen to convert **.pem** into **.ppk**. Click on the  button.



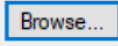
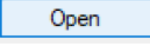
•Select the **.pem** file.

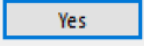
•Click  to get the **.ppk** file.

•Click  to the warning and save the file with the same name as the pem file.

•Go the Amazon EC2 instance page and get the public IP of the machine.

•Open  PuTTY Desktop app and put the public IP address in the Host Name field.

•Click  after that select  and click on  to select the private key (**.ppk**). Click on .

•Select  to connect to the machine.

•Enter user name **ec2-user** and hit Enter.

•You will see the console after a successful login.

?

Install an Apache Server

1.Switch to root user: **sudo -s**

2.Now run the updates using the following command:

•**yum -y update**

3.Once completed, lets install and run an apache server

•Install the Apache web server:

•**yum install httpd**

•On prompt Press "Y" to confirm.

•Start the web server

•**systemctl start httpd**

•Now enable httpd:

•**systemctl enable httpd**

•Check the webserver status

•**systemctl status httpd**

•You can see Active status is running.

- You can test that your web server is properly installed and started by entering the public IP address of your EC2 instance in the address bar of a web browser. If your web server is running, then you see the Apache test page. If you don't see the Apache test page, then verify whether you followed above steps properly and check your inbound rules for the security group that you created.

Create and publish page

1.Navigate to html folder where we will put our html page to be published.

- cd /var/www/html/**

2.Create a sample **test.html** file using nano editor:

- nano test.html**

3.Enter sample HTML content provided below in the file and save the file with Ctrl+X (Y).

- <HTML>Hi Simplex, I am a public page</HTML>**

4.Restart the web server by using the following command:

- systemctl start httpd**

5.Now enter the file name after the public IP which you got when you created ec2 instance in the browser, and you can see your HTML content.

- Sample URL:107.21.198.65/test.html**

6.If you can see the above text in the browser, then you have successfully completed the lab.

Completion and Conclusion

- You have successfully created and launched Amazon EC2 Instance.
- You have successfully logged into EC2 instance by SSH.
- You have successfully created a webpage and published it.

Lab 2: Introduction to Amazon Simple Storage Service (S3)

Lab Details:

1.This lab walks you through to Amazon Simple Storage Service. Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web. In this lab we will demonstrate aws S3 by creating a sample S3 bucket, uploading an object to S3 bucket and setup bucket permission and policy.

2.Duration: 00:30:00 Hrs

3.AWS Region: US East (N. Virginia)

Tasks:

- 1.Login to AWS Management Console.
- 2.Create a S3 bucket.
- 3.Upload an object to S3 Bucket.
- 4.Access the object on browser.
- 5.Setup bucket policy and permission and test the object accessibility.

Steps:

- 1.Launch your lab environment by logging to your account in aws management console.
2. Now click on **Console Login** button, this will open your **AWS Console** Account for this lab in a new tab.
- 3.Navigate to S3 by clicking on the “services” menu in the top, then click on “S3” (in the “Storage” section).

4.Create Bucket :

- 1.On the S3 dashboard click on **+ Create bucket** button and fill the bucket details.
- 2.**Bucket name** :Enter a bucket name.
- 3.**Region** :Select US East (N. Virginia)
- 4.No need to change anything just click on **Create** button.
- 5.Once you see your bucket name the list , click on it.
- 6.You will see this message "This bucket is empty. Upload new objects to get started", Now we are uploading an object into our bucket.

7.Upload Object :

- 1.Download the image from any website of your choice link to upload into the bucket you created.
- 2.Click on **upload** button.
- 3.Click on **Add files** button.
- 4.Browse the image say **abc.jpg** which you downloaded previously.
- 5.Click on **Upload** button.

6. You can watch the progress of the upload from within the Transfer panel at the bottom of the screen. Since this is a very small file, you might not see the transfer. Once your file has been uploaded, it will be displayed in the bucket.

8. Make Object Public :

1. Click on image name, You will see the image details like Owner, size, link etc.
2. Open image Link in a new tab.
3. You will see **Access Denied** message, means the object is not publicly accessible.
4. Keep this browser tab open, but return to the web browser tab with the S3 Management Console, and goto your bucket and open your uploaded image again.
5. click the **Permissions** tab, then configure:
 - i. Under the **Public access** section, select **Everyone**.
 - ii. Select **Read object**
 - iii. Click **Save**
6. Return to the browser tab that displayed **Access Denied** and refresh the page.
7. You can see your image is loaded successfully and publically accessible now.

9. Create a Bucket Policy :

1. In the previous step, you granted read access only to a specific object. If you wish to grant access to an entire bucket, you should need to create a **bucket policy**
2. Go to bucket list and click on your bucket name.
3. click the **Permissions** tab, then configure:
 - i. In the **Permissions** tab, click on **Bucket Policy**
 - ii. A blank **Bucket policy editor** is displayed.
 - iii. Before creating the policy, you will need to copy the ARN (Amazon Resource Name) of your bucket.
 - iv. Copy the **ARN** of your bucket to the clipboard. It is displayed at the top of the policy editor its look like "ARN: arn:aws:s3:::your-bucket-name"
 - v. In below policy update your bucket ARN in Resource key value and copy the policy code.

```
{
  "Id": "Policy1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": "replace-this-string-from-your-bucket-arn/*",
      "Principal": "*"
    }
  ]
}
```

4. Paste the bucket policy into the **Bucket policy editor**.
5. Click on **save**

10. Upload new object and test :

1. Download the image from website to upload into your bucket.
2. Click on **upload** button.

3. Click on **Add files** button.
4. Browse the image say abc.png which you downloaded previously.
5. Click on **Upload** button.
6. Once the image uploaded successfully, copy the image link and open in to the browser.
7. You can see your image is loaded successfully and publicably accessible.
11. You have successfully completed the lab.
12. Once you completed the steps click on End Lab from your whizlabs dashboard.

Lab 3: Creating AMI From EC2 Instance




Lab Details:

1. This lab walks you through the steps to create AMI from Amazon EC2 Instance. You will practice using Amazon Machine Images to launch Amazon EC2 Instance and Create AMI of that EC2 Instance.
2. Duration: 00:30:00 Hrs
3. AWS Region: US East (N. Virginia)

Tasks:

1. Login to AWS Management Console.
2. Create an EC2 Instance.
3. Create a new AMI using the EC2 Instance.

Launching Lab Environment

1. Launch lab environment by clicking on . This will create an AWS environment with the resources required for this lab.
2. Once your lab environment is created successfully,  will be active. Click on , this will open your **AWS Console Account** for this lab in a new tab.
3. If you face any issues, please go through **FAQs and Troubleshooting for Labs**.

Steps:

1. Navigate to **Services** menu in the top, then click on **EC2** in the **Compute** section.

2. Make sure you are in **N. Virginia** Region.

3. Click on **Launch Instance**

4. Choose an Amazon Machine Image

 **Amazon Linux 2 AMI (HVM, Arm)**
(AMI) : Amazon Linux

5. Choose an Instance Type : Select **t2.micro** **Free tier eligible** and click

on **Next: Configure Instance Details**

6. Configure Instance Details:

- Number of instances: 1
- Auto-assign Public IP: Select Enable

• Click on **Advanced Details**

• Under **User data** section, Enter the following script, which creates an HTML page served by Apache httpd web server.

```
#!/bin/bash -ex

sudo yum update -y # update packager
sudo yum -y install httpd # install apache httpd
sudo service httpd start # start apache httpd
sudo usermod -a -G apache ec2-user
sudo chown -R ec2-user:apache /var/www
sudo chmod 2775 /var/www

find /var/www -type d -exec sudo chmod 2775 {} \;
find /var/www -type f -exec sudo chmod 0664 {} \;

echo "<html><h1>Welcome to Whizlabs</h1></html>" >>
/var/www/html/index.html
```

• Click on **Next: Add Storage**

• **Add Storage**: No changes needed, click on **Next: Add Tags**

• **Add Tags** : For identification of your instances, you can add a tag with key pair combination

• Key : Name

• Value: MyEC2Server

• Click on **Next: Configure Security Group**

• **Configure Security Group** : Select **Create a new security group**,

• Security group name: MyEC2SecurityGroup

• Description : **My EC2 Security Group**

• To add **SSH**,

• Choose Type: SSH

• Source: Anywhere

• For **HTTP**, Click on **Add Rule**

• Choose Type: **HTTP**

• Source: Anywhere

• For **HTTPS**, Click on **Add Rule**

• Choose Type: **HTTPS**

• Source: Anywhere

• Click on **Review and Launch**

• **Review and Launch** : Review all your select settings and click on the **Launch**

• **Key Pair**: This step is most important, Create a new key Pair and click

on **Download Key Pair** and store them in your local.

• Click on **Launch Instances**

• **Launch Status**: Your instance is now launching, Navigate to **Instances** page from left menu and wait till status of EC2 Instance status change to running.


<input type="checkbox"/>	Name	Instance ID	Instance	Availability	Instance Status
<input type="checkbox"/>	MyEC2Server	i-04c9d253cf3385fde	t2.micro	us-east-1c	● running

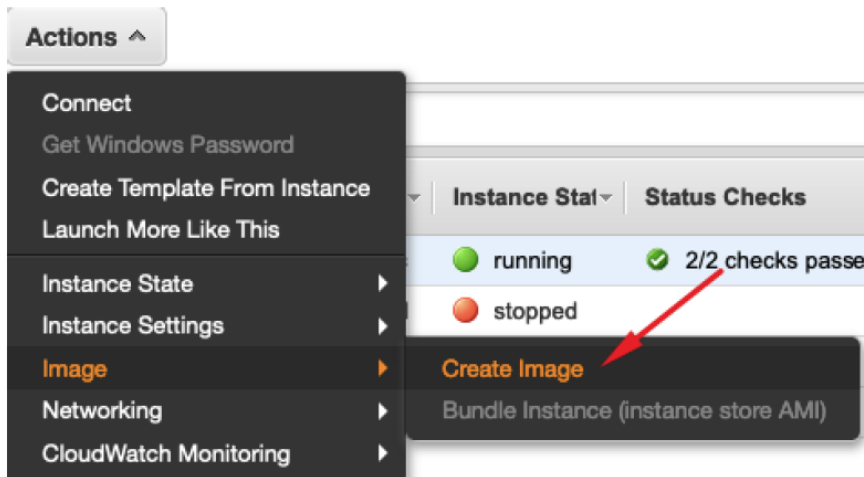
14. Note down the sample IPv4 Public IP Address of the EC2 instance.

Instance: i-04c9d253cf3385fde (MyEC2Server)		Public DNS: ec2-3-95-186-128.compute-1.amazonaws.com	
<div style="display: flex; justify-content: space-between;"> Description Status Checks Monitoring Tags </div>			
Instance ID	i-04c9d253cf3385fde	Public DNS (IPv4)	ec2-3-95-186-128.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	3.95.186.128 →
Instance type	t2.micro	IPv6 IPs	-


15. Enter the IP Address in Browser.

Creating AMI from the EC2 Instance

1. Select the MyEC2Server. Click on .
2. Under **Image**, Click on **Create Image**.



3. In the pop up window enter the following details:

- Image Name : MyEC2Image
- Image Description : My EC2 Image
- Leave other details as default.
- Click on .

Check the newly created Image

1. Navigate to **AMI's** under **Images** in left menu.
2. You can see that the Image is getting generated and status is **pending**.

	Name	AMI Name	AMI ID	Source	Owner	Visibility	Status
	MyEC2Image		ami-02516d21ccf560b...	757712384777/MyEC2Im...	757712384777	Private	pending

3. Once the process is completed, status becomes **available**.

<input type="checkbox"/>	Nam	AMI Name	AMI ID	Source	Owner	Visibility	Status
<input type="checkbox"/>		MyEC2Image	ami-02516d21ccf560b...	757712384777/MyEC2Im...	757712384777	Private	available

4. Now we can use this Image AMI to create brand new instances.


5. When you create a new EC2 instance with it, you will be able to see HTML page displaying message **Welcome to Simplex**. This shows that the data in new instance is the same as the one in first instance created.

Completion and conclusion

1. You have successfully created an EC2 instance.
2. You have successfully created an image directly from EC2 instance.

End Lab

1. You have successfully completed the lab.

2. Once you have completed the steps click on  from your whizlabs dashboard.

Lab 4: How to enable versioning Amazon S3

Lab Details:

- 1.This lab walks you through to the steps how to Enables Versioning to a AWS S3 Bucket. Versioning enables you to keep multiple versions of an object in one bucket. In this lab we learn how to enable object versioning on a S3 bucket.
- 2.Duration: 00:30:00 Hrs
- 3.AWS Region: US East (N. Virginia)

Tasks:

- 1.Login to AWS Management Console.
- 2.Create a S3 bucket.
- 3.Enable object versioning on bucket.
- 4.Upload an text file to S3 Bucket.
- 5.Test object versioning with update text file and upload.

Steps:

- 1.Login to aws management console with your account.
2. Now click on **Console Login** button, this will open your **AWS Console** Account for this lab in a new tab.
- 3.Navigate to S3 by clicking on the “services” menu in the top, then click on “S3” (in the “Storage” section).

4.Create Bucket :

- 1.On the S3 dashboard click on **+ Create bucket** button and fill the bucket details.
- 2.**Bucket name** :Enter a bucket name.
- 3.**Region** :Select US East (N. Virginia)
- 4.No need to change anything just click on **Create** button.

5.Enable Versioning :

- 1.Go to bucket list and click on your bucket name.
 - 2.Click On **Properties** tab.
 - 3.Choose **Versioning**
 - 4.Choose **Enable versioning** and then choose Save.
- 6.Once you see your bucket name the list , click on it.
- 7.You will see this message "This bucket is empty. Upload new objects to get started", Now we are uploading an object into our bucket.

8.Upload Object :

- 1.Download the text file from any website link to upload into the bucket you created.
- 2.Click on **upload** button.
- 3.Click on **Add files** button.

4. Browse the text file say **sample.txt** which you downloaded previously.
5. Click on **Upload** button.
6. You can watch the progress of the upload from within the Transfer panel at the bottom of the screen. Since this is a very small file, you might not see the transfer. Once your file has been uploaded, it will be displayed in the bucket.

9. Make Bucket Public with Bucket Policy :

1. In the previous step, you granted read access only to a specific object. If you wish to grant access to an entire bucket, you should need to create a **bucket policy**
2. Go to bucket list and click on your bucket name.
3. click the **Permissions** tab, then configure:

- i. In the **Permissions** tab, click on **Bucket Policy**
- ii. A blank **Bucket policy editor** is displayed.
- iii. Before creating the policy, you will need to copy the ARN (Amazon Resource Name) of your bucket.
- iv. Copy the **ARN** of your bucket to the clipboard. It is displayed at the top of the policy editor its look like "ARN: arn:aws:s3:::your-bucket-name"
- v. In below policy update your bucket ARN in Resource key value and copy the policy code.

```
{
  "Id": "Policy1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmnt1",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": "replace-this-string-from-your-bucket-arn/**",
      "Principal": "*"
    }
  ]
}
```

4. Paste the bucket policy into the **Bucket policy editor**.
5. Click on **save**

10. Now open the text file link into browser you can see the text in file "This is version 1"

11. Upload text file version 2 :

1. Update the text of your sample.txt file.
2. Click on **upload** button.
3. Click on **Add files** button.
4. Browse the **sample.txt** file which you downloaded previously.
5. Click on **Upload** button.
6. Once the file uploaded successfully, copy the image link and open in to the browser.
7. You can see the latest version 2 of sample.txt file which you uploaded.
8. **Now testing the old version:** For enable the old version of file we need to delete the latest version of file.
9. Click on file name and you can see the details of file.
10. On top section just next to the object name you can find a drop down of all version of your object called **Latest version**

11. Click on **Latest version** drop down and delete the latest version of sample.txt
12. Now refresh you S3 object URL you can see the older version of your sample.txt which you uploaded first time. You have successfully completed the lab.

Index

A

Advanced Encryption Standard _____ 97, 101, 157
Amazon Web Services 76, 83, 88, 97, 107, 111, 112, 118,
139, 157
American Institute of Certified Public Accountants __ 79,
157
Anycast domain name _____ 16, 157
Application Access Points _____ 44, 157
Application Delivery Network (ADN) _____ 44, 157
Autonomous Systems _____ 60, 157

B

Backup and Recovery Solutions _____ 45, 157
Broadband Services Forum _____ 9, 157

C

Cache Array Routing Protocol _____ 157
Cloud-based CDNs _____ 9, 36, 38, 157
Common Internet File System _____ 99, 157
Content Delivery Network _____ 157
Content Management and Online Reporting _____ 157
Cooperative Media On-Demand on the Inter
Net(COMODIN) _____ 157

D

Data Security Standard _____ 79, 157
Department of Defense _____ 80, 157
Distributed Database (DDB) _____ 157
Distributed Denial-Of-Service (DDoS) _____ 157
Distributed Sloppy Hash Table (DSHT) _____ 50, 157
Distribution Hash Table _____ 63, 157
Domain Name Service (DNS) _____ 157

E

Elastic Load Balancing (ELB) _____ 84, 157

F

Federal Information Security Management Act (FISMA)
_____ 157

G

Global Server Load Balancing (GSLB) __ 13, 69, 70, 157

H

Hypertext Caching Protocol (HTCP) _____ 22, 57, 157

I

Identity and Access Management (IAM) _ 101, 105, 114,
140, 157
Internet Cache Protocol (ICP) _____ 22, 27, 56, 57, 157
Internet Engineering Task Force (IETF) _____ 9, 157
Internet of things (IoT) _____ 113, 157

L

Lightweight Directory Access Protocol (LDAP) ____ 158

M

MediaWise Cloud content orchestrator (MCCO)_ 43, 158

N

Network Element Control Protocol (NECP) _____ 56, 158
Network File System (NFS) _____ 99, 158
Not Only SQL (NoSQL) _____ 158

P

Payment Card Industry (PCI) _____ 79, 158
peer-to-peer (P2P) _____ 17, 23, 158
Point of Presence (POP) _____ 11, 158
Public Key Infrastructures (PKI) _____ 158

Q

Quality of Service (QoS) _____ 8, 158

R

Return On Investment (ROI) _____ 158
RFCs (Request For Comments) _____ 9, 158

S

Secure File Transfer Appliances (SFTA) _____ 45, 158
Secure Sockets Layer (SSL) _____ 101, 158
Small to medium enterprises (SME) _____ 158
Small to Medium Enterprises (SME) _____ 158
Software Streaming Transfer Protocol (SSTP) ____ 45, 158
Solid State Drive (SSD) _____ 158

T

Time To Live (TTL) _____ 14, 158
Transport Control Protocol (TCP) _____ 20, 158

U

User Acceptance Testing (UAT) _____ 100, 158
User Generated Videos (UGV) _____ 11, 158

V

Video-on-Demand (VoD) _____ 9, 158
Virtual Data Centers (VDCs) _____ 44, 158
Virtual Desktop Infrastructure (VDI) _____ 158
Virtual Private Cloud (VPC) _____ 158
Virtual Private Network (VPN) _____ 158

W

Web Cache Coordination Protocol (WCCP) _____ 56

