Research Article

Binary Integer Programming Approach to Optimal Content Placement in Cloud-based Content Delivery Networks

Nay Myo Sandar*
Department of International College, Modern Trade Business Management, Panyapiwat Institute of Management, Nonthaburi, Thailand

* Corresponding author. E-mail: naymyo@pim.ac.th

DOI: 10.14416/j.asep.2021.01.006
Received: 27 January 2020; Revised: 9 July 2020; Accepted: 16 November 2020; Published online: 27 January 2021

Abstract
Over the last decades, Content Delivery Networks (CDNs) have been developed to overcome the limitation of user perceived latency by replicating contents from the origin server to its content servers around the globe close to clients. As some contents occupy most of the storage capacity and processing power in traditional private content servers, cloud computing can provide a pool of storage and processing power resources for caching contents. By adopting cloud computing to CDN, the content provider can use the cloud infrastructure by distributing the contents to cloud servers which will then deliver to near clients. In this paper, we propose a cloud-based CDN framework designed by two schemes, including 1) UDP/TCP-based content distribution from the origin server to cloud servers and 2) SDN-based cloud server coordination. In addition, we also formulate the optimal content placement problem using binary integer programming to minimize the total cost of renting resources including storage, processing power, and network bandwidth in cloud providers for hosting contents from the origin server. Then, the optimal solution obtained from binary integer programming is evaluated by greedy algorithm and simulations. The proposed framework help the content provider to offer high quality services to clients while minimizing the cost of rented cloud resources.

Keywords: Cloud-based Content Delivery Networks, User Datagram Protocol, Transmission Control Protocol, Software Defined Networking, Binary integer programming, Greedy algorithm

1 Introduction

In recent years, content delivery networks have distributed the contents (including images, real-time video, media clips, advertisements, web content, etc.) from content servers scattered in multiple locations so that clients can retrieve the contents from nearby content servers to reduce access latency instead of retrieving from remote origin server [1]. Today, there are many companies applied content delivery networks, for example, Akamai [2], Limelight [3], CoralCDN [3], CoDeen [4], etc. Although CDN can provide services to the clients by reducing access latency, there are still remaining serious issues of processing power and cache space limitation in traditional private content servers due to some large file downloads such as software patches, high quality videos, real - time video conferences, etc.

To achieve a higher amount of processing power and storage capacity, cloud computing provides unlimited access to computational resources such as processing power, storage, and network bandwidth [5]. With cloud computing, the content provider can use a pool of large resources and allocates loads of contents to cloud servers. Then, the clients in different geographical locations can access the contents from the near cloud providers. In this paper, we propose a scalable cloud based content delivery networks framework designed by two schemes, including 1) UDP/TCP-based content distribution from the origin server to cloud servers and 2) SDN-based cloud server coordination.

First, a UDP/TCP-based content distribution scheme is proposed to provide fast and reliable content distribution from the origin server to multiple cloud servers. In particular, the content provider (or origin server) distributes the contents with high - speed UDP (User Datagram Protocol) protocol to cloud servers deployed in several locations. However, the nature of UDP is unreliable which can result in loss of contents. Therefore, TCP (Transmission Control Protocol) connection is further supplemented to provide a reliable connection by acknowledging the received contents between the origin server and cloud servers.

Second, an SDN-based cloud server coordination scheme is proposed to manage efficiently for cloud servers. Once the client sends a content request to one of the cloud servers and if that cloud server does not have that requested content by the client, SDN (Software Defined Networking) technique allows the controller to monitor, which other cloud servers have the content request. Once the content request is found in the other cloud server, the controller executes the routing with the smallest latency and writes a control program or instruction for content the content among the cloud servers in hop by hop manner.

By integrating these two schemes in the proposed cloud-based CDN framework, it can achieve fast and reliable content distribution from the origin server to cloud servers and efficient coordination between cloud servers. In addition, we also consider the cost-effectiveness of the content provider. Although cloud computing can reduce the infrastructure cost for the content provider, it still leaves the problem of optimal distribution of numerous contents among multiple cloud servers since it has impact the monetary cost to provision resources (e.g., processing power, storage, and network bandwidth). To achieve the optimal content placement in cloud servers with the minimum total cost, we formulate an optimization model using binary integer programming and obtain the optimal solution. Then, we applied a greedy algorithm and performed simulations to evaluate the optimal solution from binary integer programming. With the optimization model, the content provider can optimally distribute content to cloud servers at reasonable prices.

The key contributions of this paper are to:

Propose scalable cloud-based CDN framework designed by two schemes UDP/TCP-based content distribution and SDN-based cloud server coordination.

Formulate binary integer programming model to achieve cost effectiveness for content provider by optimally allocating contents to cloud servers for proposed framework.

and Apply greedy algorithm and perform simulations to evaluate the optimal solution obtained from binary integer programming.

The remainder of this paper is organized as follows. The related works are reviewed in Section 2. The proposed scalable cloud-based content delivery networks framework is proposed in Section 3. The system model and assumption are presented in Section 4. The problem formulation based on binary integer programming is derived in Section 5. The numerical results are studied in Section 6. Finally, the contributions of this paper are concluded in Section 7.

2 Related Work

Recently, a number of studies have investigated CDNs Among those works the researchers developed a new technique called cluster shutdown to save energy by turning off the servers in CDN during off-peak hours [6]. A new CDN architecture called server clusterrs was introduced improve system performance by grouping the content servers and exploiting the benefits of server cooperation [1]. A prototype of simple application independent data lockers (SAILOR) was designed and implemented, which provides a shared in-network storage infrastructure for content distribution to improve network efficiency and application performance [7].

Nowadays, content delivery networks (CDNs) using cloud computing for different concerns have started to emerge. For example, the work in [8] proposed and implemented a CloudSeal scheme for secure content storage and delivery via the public cloud. The work in [9] presented a novel CDN architecture called ActiveCDN which utilizes cloud computing to enhance content delivery services. The work in [10] proposed an Elastic Video Endpoint (EVE) to dynamically provision cloud resources when they are required.

In this paper, we propose a scalable cloud-based content delivery networks framework designed by two schemes, including 1) UDP/TCP-based content distribution and 2) SDN-based cloud server coordination. By using both UDP and TCP protocols, they can provide fast and reliable data transmission. The previous work in [11] proposed a Reliable Blast UDP (RBUDP) scheme which applies UDP protocol and TCP protocol to achieve reliable and fast large data transmission. Likewise, the use of SDN technique can manage the network efficiently with the control program. The recent work in [12] proposed an SDN-based architecture to design a dynamic and highly efficient bulk data transfers among geo-distributed multiple datacenters. Based on the previous works, it demonstrated that the aforementioned network techniques can offer several advantages applied in different fields. By applying the advantages of those network techniques in our proposed framework, we expect that it can provide several benefits of fast, reliability, and efficiency for content delivery networks. Furthermore, we formulate a binary integer programming model for the problem of optimal content placement to cloud servers to minimize the total cost of storage, processing power, and network bandwidth for the content provider.

Compared to the earlier works, we realize that lots of researchers explored various challenges and proposed various approaches to improve CDN service. However, to the best of our knowledge, they still lack of consideration to combine the aforementioned network techniques to completely perfect the scalability of content delivery networks. Moreover, we also apply the optimization technique based on binary integer programming to minimize the total cost of cloud resources for the content provider in the proposed cloud-based CDN framework.

3 Proposed Framework

In this section, we present a scalable cloud-based content delivery networks framework as shown in Figure 1. The proposed framework in Figure 1 consists of three basic components : content distribution, server coordination, and content delivery. First, the content distribution component distributes the contents from the origin server (or content provider) to cloud servers. After that, the cloud servers provide a large amount of storage and processing power for the received contents for various services.

- Content database service: This service is used for organizing and managing the contents.
- Content replica service: This service is used for replicating the same contents in cloud servers.
- Content compression service: Some content might be large so that this service can compress the size of content for efficient storage and transfer to customers.

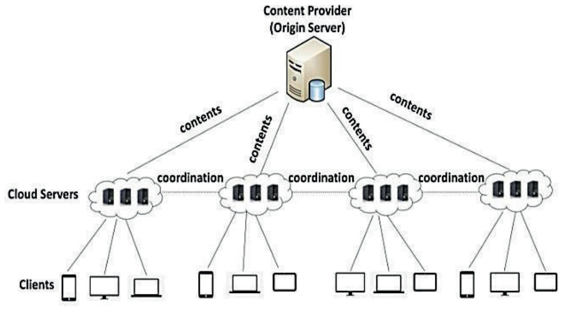

Figure 1: Scalable cloud-based content delivery networks.

- Content rendering service: This service is used for displaying the contents on clients' devices.

In addition, the cloud servers coordinate with each other to share the contents. In the case where one of the servers cannot fulfill the client's request, the content delivery component delivers the requested content from the nearest cloud server.

To improve the scalability of the proposed framework, each component is designed by several schemes as presented in the following subsections.

3.1 *UDP/TCP-based content distribution*

As illustrated in Figure 2, this section presents the concept of UDP/TCP-based content distribution scheme [13].

1) Frist, the origin server distributes the contents (e.g., movie, music, image, TV shows, etc.) to one cloud server using User Datagram Protocol (UDP) protocol which provides a faster connection than Transmission Control Protocol (TCP). However, UDP protocol is unreliable and results in loss of contents.

2) Therefore, after distributing contents, the origin server sends a Done signal with a list of contents to that cloud server over TCP protocol which is more reliable than UDP protocol.

3) The cloud server receives and checks a list of contents. Then, it sends an Acknowledged signal back to the origin server which contents are received and which contents are missed over TCP protocol.

4) After that, the origin server redistributes the missing contents to the cloud server. In this way, this

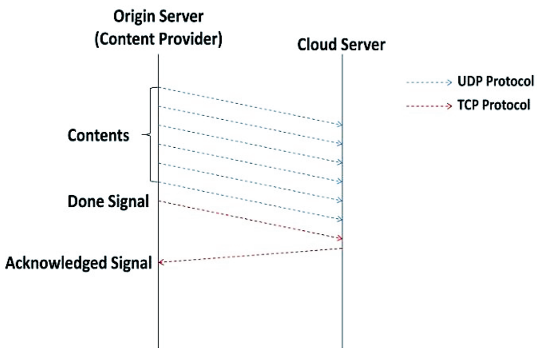

Figure 2: UDP/TCP-based content distribution scheme.

proposed scheme can provide fast and reliable content distribution.

3.2 *SDN-based cloud server coordination*

This section presents an SDN-based cloud server coordination scheme as shown in Figure 3. This proposed scheme utilizes the developing technology called software defined networking (SDN). This network technology allows network administrators or controllers to provide programmable network paths [14]. We present the step by step routing procedure for this proposed scheme below.

1) There are multiple cloud servers in different locations $\{j_1, j_2, j_3, \ldots\}$.

2) There are a set of contents $\{k_1, k_2, k_3, \ldots\}$ in different cloud servers.

3) There is a controller which is centralized and controlled among cloud servers.

4) If a client requests content k_1 to cloud server j_1, then j_1 (if it has) transfers content k_1 to client.

5) If j_1 does not have the content k_1 requested by the client, the controller monitors the other cloud servers $\{j_2, j_3, j_4, \ldots\}$.

6) If content k_1 is found in the other cloud server j_4, the controller performs routing execution from j_1 to j_4 in terms of smallest latency.

Then, the command dispatcher in the controller sends the signal to the cloud server j_1 with the generated routing decision.

The signal handler in cloud server j_1 handles the signal sent by the controller via the control path.

After that, the forwarding information base in cloud server j_1 forwards the name of requested content k_1 to the other cloud server j_4 based on the routing

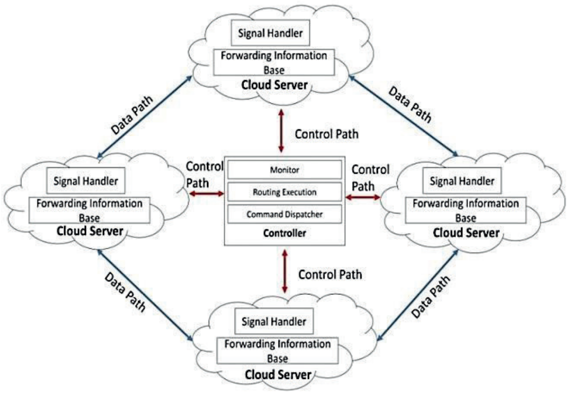

Figure 3: SDN-based cloud server coordination scheme.

execution by the controller.

The other cloud server j_4 will return back the content k_1 requested by a client to the cloud server j_1 by following the reverse path.

By using SDN approach, this proposed scheme can efficiently manage and control the routing of content sharing among cloud servers with low latency. However, SDN can sometimes have the risk of software errors so that it can result in unreliable network paths for sharing contents between cloud servers. For this issue, the controller must be frequently upgraded to fix the errors and improve network performance.

4 System Model and Assumption

As proposed in Figure 1, the system model of cloud-based CDN framework consists of four main entities (i.e., content provider, contents, cloud servers, and clients). A content provider has a set of contents (e.g., audio, video file, etc.) denoted by $k \in K$. The content provider needs to distribute contents to cloud servers in advance before delivering the contents to its clients. The same content can be distributed to more than one cloud server. The content provider rents storage, processing power, and network bandwidth offered by cloud providers for distributing content. Let e_k denotes the processing power required by each content k (CPU-hours). Let h_k denotes the storage capacity required by each content k (MB). Let n_k denotes the network bandwidth to transfer for each content k (MB/sec).

Let $j \in J$ denotes a set of cloud servers. Let a_j denotes the maximum processing power from each cloud server j (CPU-hours) to compute contents k. Let r_j denotes the maximum storage capacities from each cloud server j (GB) to store contents k. Let b_j denotes the maximum network bandwidth from each cloud server (GB/sec) to distribute contents k. Let cp_j denote unit cost to process contents k charged by cloud servers j ($/CPU-hours). Let cs_j denotes unit cost to store contents k charged by cloud servers j ($/GB). Let cb_j denotes unit cost to distribute contents k charged by cloud servers j ($/GB-sec).

In this system model, we propose the problem for content providers how to distribute the contents to the cloud servers in order to minimize the total cost (including processing power, storage, and network bandwidth) subject to the constraints for processing power, storage, and delay are defined as well. Let X_{kj} denotes the decision variable which indicates that content k is stored in cloud servers j.

5 Problem Formulation

To solve the optimal content placement problem, we formulate the optimization approach based on binary integer programming as follows.
Minimize:

$$\sum_{k\in K} \sum_{j\in J} [(e_k \times cp_j \times X_{kj}) + (h_k \times cs_j \times X_{kj}) + (n_k \times cb_j \times X_{kj})] \qquad (1)$$

Subject to:

$$\sum_{j\in J} X_{kj} \geq 1, \quad \forall k \in K \qquad (2)$$
$$e_k X_{kj} \leq a_j, \quad \forall k \in K, j \in J \qquad (3)$$
$$h_k X_{kj} \leq r_j, \quad \forall k \in K, j \in J \qquad (4)$$
$$n_k X_{kj} \leq b_j, \quad \forall k \in K, j \in J \qquad (5)$$
$$X_{kj} \in \{0,1\}, \quad \forall k \in K, j \in J \qquad (6)$$

Objective function in (1) is to minimize the total cost for processing power, storage capacity, and network bandwidth for distribution of contents. The constraint in (2) ensures that the content provider must replicate each content k at least one cloud server j. The constraint in (3) ensures that the processing power required by each content e_k must not exceed the maximum processing power offered by each cloud server a_j. The constraint in (4) ensures that the storage capacity required by each content h_k must not exceed the maximum storage capacity offered by cloud each server r_j. The constraint in (5) ensures that the network bandwidth required by each content n_k must not exceed the maximum network bandwidth offered by each cloud server b_j. The last constraint in (6) indicates that decision variable X_{kj} takes binary integer numbers (i.e., 0 and 1). The meaning of 0 represents that content is not allocated to cloud servers while the meaning of 1 represents that content is allocated to cloud servers.

6 Parameter Setting and Numerical Results

To solve the formulations in Section 5, we define parameters and generate numerical results. In the system model shown in Figure 1, it is considered to simplify that there are four contents (i.e., $\{k_1, k_2, k_3, k_4\}$) and four cloud servers (i.e., $\{j_1, j_2, j_3, j_4\}$). The processing power required by each content k_1, k_2, k_3, and k_4 is 50, 52, 53, and 55 (CPU-hours), respectively. The storage capacity required by each content k_1, k_2, k_3, and k_4 is 73, 74, 75, and 78 (MB), respectively. The network bandwidth required by each content k_1, k_2, k_3, and k_4 is 5.69, 6.25, 7.87, and 10.34 (MB/sec), respectively.

The maximum processing power offered by cloud servers j_1 and j_2 is 1200 (CPU-hours), j_3 is 1500 (CPU-hours) and j_4 is 1000 (CPU-hours), respectively. The maximum storage capacity offered by cloud servers j_1 and j_4 are 1000 (GB), j_2 is 2000 (GB), and j_3 is 3000 (GB), respectively. The maximum network bandwidth offered by each cloud server j_1 is 33.33 (GB/sec), j_2 is 67.68 (GB/sec), j_3 is 48.84 (GB/sec), and j_4 is 50.25 (GB/sec), respectively. The unit costs of processing power charged by each cloud server j_1 is 1.280, j_2 is 1.320, j_3 is 1.150, and j_4 is 1.250 ($/CPU-hours), respectively. The unit costs of storage capacity charged by each cloud server j_1 are 0.930, j_2 is 1.200, j_3 is 1.250, and j_4 is 0.950 ($/MB), respectively. The unit costs of network bandwidth charged by each cloud server j_1 are 0.080, j_2 is 0.083, j_3 is 0.089, and j_4 is 0.090 ($/MB-sec), respectively.

After that, we input the defined parameters to the formulations from (1)–(6) and solve them by GAMS/CPLEX solver [15]. The numerical results of optimal content placement to cloud servers obtained from GAMS/CPLEX solver are described in Table 1.

Table 1: Numerical results of optimal content placement

Decision Variable (X_{kj})	Optimal Solution
$X_{k_1j_1}$	1
$X_{k_1j_2}$	0
X_{k_1j3}	0
$X_{k_1j_4}$	0
$X_{k_2j_1}$	0
$X_{k_2j_2}$	0
$X_{k_2j_3}$	0
$X_{k_2j_4}$	1
$X_{k_3j_1}$	0
$X_{k_3j_2}$	0
$X_{k_3j_3}$	0
$X_{k_3j_4}$	1
$X_{k_4j_1}$	1
$X_{k_4j_2}$	0
$X_{k_4j_3}$	0
$X_{k_4j_4}$	0
Total Cost	$550.183

In Table 1, the optimal solution is obtained from the optimization approach where the content provider can allocate the contents $k_1$to cloud server j_1, k_2 to cloud server j_4, k_3 to cloud server j_4, and k_4 to cloud server j_1 with the minimum total cost of $550.183.

6.1 *Greedy algorithm*

In this section, we applied the greedy algorithm to evaluate the optimal solution obtained by binary integer programming as presented in Algorithm

Algorithm 1: Greedy method for optimization problem

```
1. Select $X \leftarrow$ Input Domain Set $\{C\}$
2. If (feasible ($X$)) then go to step 3 else go to step 1
3. X $\rightarrow$ solve model in (1)
4. X $\cap$ Solution Set $\{S\}$
5. Repeat step 1
```

Generally, a greedy algorithm solves the optimization problem by making choices that seem to be the best at the particular moment [16]. As shown in Algorithm 1, the greedy algorithm proceeds step by step. First, we select an input X from the input domain set $\{C\}$. Next, we check the selected input X whether it is feasible or not by checking the problem's constraints. If the input X satisfies the constraints, it is called a feasible solution and assigned to objective function. Then, this feasible input is added to the solution set $\{S\}$. On the other hand, if the input X violates one constraint; it is called an infeasible solution and never considered. Then we select new input from input domain set $\{C\}$ and perform through a sequence of steps.

Based on objective function (either maximize or minimize), we choose a local optimal solution from solution set $\{S\}$ which looks best at the moment among all feasible candidates. To better understanding the above algorithm, we demonstrate how it works for optimization problem as follow. Table 2 shows the input domain set $\{C\}$ which contains a set of n inputs or decision variables (i.e., X_{kj}) represented by binary numbers 0 and 1. In our work, we made 10 attempts to find an optimal solution by examining each decision variable from Table 2 on the following page.

Each decision variable is first checked its feasibility in given constraints (2)–(5). As can be seen in Table 2, the feasible solutions are highlighted with yellow color whereas infeasible solutions are highlighted with red color.

After that, we compute the feasible solutions into the objective function (1) and add them to the solution set $\{S\}$ as presented in Table 3 on the following page. In Table 3, each feasible solution yields a different objective function. Intuitively, we highlight the optimal solution with green color which is the same as the one obtained from GAMS/CPLEX solver with the minimum objective function value.

It also looks best local choice among all feasible solutions until that attempt.

To be concluded, a greedy algorithm can be applied to an optimization problem which is easy to implement. However, it can sometimes fail to find the global optimal solution because it makes a locally optimal choice and does not operate exhaustively on all the inputs.

6.2 *Simulation with different scenarios*

In this section, more simulation is performed in GAMS/CPLEX solver for evaluating the performance of using optimization approach. In the simulation, different parameter settings for a number of contents, storage capacity of contents, processing power of contents, and network bandwidth of contents are studied in three different scenarios.

Table 2: Input domain set $\{C\}$

Attempts	Decision Variable (X_{kj})															
	$X_{k_1j_1}$	$X_{k_1j_2}$	$X_{k_1j_3}$	$X_{k_1j_4}$	$X_{k_2j_1}$	$X_{k_2j_2}$	$X_{k_2j_3}$	$X_{k_2j_4}$	$X_{k_3j_1}$	$X_{k_3j_2}$	$X_{k_3j_3}$	$X_{k_3j_4}$	$X_{k_4j_1}$	$X_{k_4j_2}$	$X_{k_4j_3}$	$X_{k_4j_4}$
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
2	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
3	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
4	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
6	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
7	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
8	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0	0
9	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
10	0	1	1	0	0	1	1	0	0	0	0	0	0	1	1	0
11	0	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0
12	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
13	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
14	1	1	0	0	1	1	0	0	0	0	0	0	1	1	0	0
15	1	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0
⋮	⋮															
n	n															

Table 3: Solution set $\{S\}$

Feasible Solutions for Decision Variable (X_{kj})																Objective Function Value ($)
$X_{k_1j_1}$	$X_{k_1j_2}$	$X_{k_1j_3}$	$X_{k_1j_4}$	$X_{k_2j_1}$	$X_{k_2j_2}$	$X_{k_2j_3}$	$X_{k_2j_4}$	$X_{k_3j_1}$	$X_{k_3j_2}$	$X_{k_3j_3}$	$X_{k_3j_4}$	$X_{k_4j_1}$	$X_{k_4j_2}$	$X_{k_4j_3}$	$X_{k_4j_4}$	
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	550.214
0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	619.183
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1169.397
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1189.916
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	550.183

6.2.1 First scenario

For the first scenario, it is assumed that there are five contents, denoted by $|K| = 5$. The required storage capacity of each content k_1, k_2, k_3, k_4, and k_5 are 20, 25, 30, 35, and 40 (MB), respectively. The required processing power of each content k_1, k_2, k_3, k_4, and k_5 are 30, 35, 40, 45, and 50 (CPU-hours), respectively. The required network bandwidth of each content k_1, k_2, k_3, k_4, and k_5 are 2.69, 3.25, 4.87, 5.34, and 6.29 (MB/sec). Then, the numerical result of allocating contents to each cloud server for the first scenario is shown in Table 4.

In Table 4, the optimal solution with optimization approach for the first scenario shows that the contents k_1 are allocated to cloud server j_1, k_2 is allocated to cloud server j_4, k_3 is allocated to cloud server j_4, k_4 is allocated to cloud server j_4, and k_5 is allocated to cloud server j_4 with the minimum total cost of $379.493. The values without optimization approach show that the contents k_1 is allocated to cloud server j_2, k_2 is allocated to cloud server j_1, k_3 allocated to cloud server j_3, k_4 is allocated to cloud server j_1, and k_5 is allocated to cloud server j_4 with the higher total cost of $397.510.

6.2.2 Second scenario

For the second scenario, it is assumed that there are six contents, denoted by $|K| = 6$. The required storage capacity of each content k_1, k_2, k_3, k_4, k_5 and k_6 are 30, 35, 40, 45, 50, and 55 (MB), respectively. The required processing power of each content k_1, k_2, k_3, k_4, k_5, and k_6 are 40, 45, 50, 55, 60, and 65 (CPU-hours), respectively. The required network bandwidth of each

Table 4: Numerical result for first scenario

Decision Variable (X_{kj})	With Optimization	Without Optimization
$X_{k_1 j_1}$	1	0
$X_{k_1 j_2}$	0	1
$X_{k_1 j3}$	0	0
$X_{k_1 j_4}$	0	0
$X_{k_2 j_1}$	0	1
$X_{k_2 j_2}$	0	0
$X_{k_2 j_3}$	0	0
$X_{k_2 j_4}$	1	0
$X_{k_3 j_1}$	0	0
$X_{k_3 j_2}$	0	0
$X_{k_3 j_3}$	0	1
$X_{k_3 j_4}$	1	0
$X_{k_4 j_1}$	0	1
$X_{k_4 j_2}$	0	0
$X_{k_4 j_3}$	0	0
$X_{k_4 j_4}$	1	0
$X_{k_5 j_1}$	0	0
$X_{k_5 j_2}$	0	0
$X_{k_5 j_3}$	0	0
$X_{k_5 j_4}$	1	1
Total Cost	$379.493	$397.510

Table 5: Numerical result for second scenario

Decision Variable (X_{kj})	With Optimization	Without Optimization
$X_{k_1 j_1}$	0	0
$X_{k_1 j_2}$	0	0
$X_{k_1 j3}$	0	1
$X_{k_1 j_4}$	1	0
$X_{k_2 j_1}$	0	0
$X_{k_2 j_2}$	0	1
$X_{k_2 j_3}$	0	0
$X_{k_2 j_4}$	1	0
$X_{k_3 j_1}$	0	0
$X_{k_3 j_2}$	0	0
$X_{k_3 j_3}$	0	0
$X_{k_3 j_4}$	1	1
$X_{k_4 j_1}$	0	1
$X_{k_4 j_2}$	0	0
$X_{k_4 j_3}$	0	0
$X_{k_4 j_4}$	1	0
$X_{k_5 j_1}$	0	0
$X_{k_5 j_2}$	0	1
$X_{k_5 j_3}$	0	0
$X_{k_5 j_4}$	1	0
$X_{k_6 j_1}$	0	0
$X_{k_6 j_2}$	0	0
$X_{k_6 j_3}$	0	1
$X_{k_6 j_4}$	1	0
Total Cost	$621.150	$676.444

content k_1, k_2, k_3, k_4, k_5, and k_6 are 3.15, 4.29, 5.67, 6.24, 7.50, and 8.15 (MB/sec). Then, the numerical result of allocating contents to each cloud server for the second scenario is shown in Table 5.

In Table 5, the optimal solution with optimization approach for the second scenario shows that the contents k_1 is allocated to cloud server j_4, k_2 is allocated to cloud server j_4, k_3 is allocated to cloud server j_4, k_4 is allocated to cloud server j_4, k_5 is allocated to cloud server j_4, and k_6 is allocated to cloud server $j_4$with the minimum total cost of $621.150. The values without optimization approach show that the contents k_1 is allocated to cloud server j_3, k_2 is allocated to cloud server j_2, and k_3 is allocated to cloud server j_4, k_4 is allocated to cloud server j_1, k_5 is allocated to cloud server j_2, and k_6 is allocated to cloud server j_3 with the higher total cost of $676.444.

6.2.3 Third scenario

For the third scenario, it is assumed that there are seven contents, denoted by $|K| = 7$. The required storage capacity of each content k_1, k_2, k_3, k_4, k_5, k_6, and k_7 are 40, 45, 50, 55, 60, 65, 70 (MB), respectively. The required processing power of each content k_1, k_2, k_3, k_4, k_5, k_6, and k_7 are 45, 50, 55, 60, 65, 70, and 75 (CPU-hours), respectively. The required network bandwidth of each content k_1, k_2, k_3, k_4, k_5, k_6, and k_7 are 4.30, 5.34, 6.80, 7.35, 8.60, 9.85, and 10.25 (MB/sec). Then, the numerical result of allocating contents to each cloud server for the third scenario is shown in Table 6.

In Table 6, the optimal solution with optimization approach for the third scenario shows that the contents k_1 is allocated to cloud server j_4, k_2 is allocated to cloud server j_4, k_3 is allocated to cloud server j_4, k_4 is allocated to cloud server j_4, k_5 is allocated to cloud server j_4, k_6 is allocated to cloud server j_4, and k_7 is allocated to cloud server j_4 with the minimum total cost of $884.974. The values without optimization approach show that the contents k_1 is allocated to cloud server j_1, k_2 is allocated

Table 6: Numerical result for third scenario

Decision Variable (X_{kj})	With Optimization	Without Optimization
$X_{k_1j_1}$	0	1
$X_{k_1j_2}$	0	0
X_{k_1j3}	0	0
$X_{k_1j_4}$	1	0
$X_{k_2j_1}$	0	0
$X_{k_2j_2}$	0	0
$X_{k_2j_3}$	0	1
$X_{k_2j_4}$	1	0
$X_{k_3j_1}$	0	0
$X_{k_3j_2}$	0	0
$X_{k_3j_3}$	0	0
$X_{k_3j_4}$	1	1
$X_{k_4j_1}$	0	0
$X_{k_4j_2}$	0	1
$X_{k_4j_3}$	0	0
$X_{k_4j_4}$	1	0
$X_{k_5j_1}$	0	0
$X_{k_5j_2}$	0	0
$X_{k_5j_3}$	0	1
$X_{k_5j_4}$	1	0
$X_{k_6j_1}$	0	1
$X_{k_6j_2}$	0	0
$X_{k_6j_3}$	0	0
$X_{k_6j_4}$	1	0
$X_{k_7j_1}$	0	0
$X_{k_7j_2}$	0	0
$X_{k_7j_3}$	0	0
$X_{k_7j_4}$	1	1
Total Cost	$884.974	$928.467

to cloud server j_3, and k_3 is allocated to cloud server j_4, k_4 is allocated to cloud server j_2, k_5 is allocated to cloud server j_3, k_6 is allocated to cloud server j_1, and k_7 is allocated to cloud server j_4 with higher total cost of $928.467.

According to the simulation results, it can be deducted that our optimization approach can provide the optimal solution to allocate contents to cloud servers with the minimum total cost for different data sets in different scenarios.

6.3 *Scalability Experiment with a Large Number of Contents*

This experiment was repeated 50 times using different scenarios with different combinations of contents $\{k_1, k_2, k_3,\ldots, k_{50}\}$. Each content has different requirements of processing power, storage capacity, and network bandwidth. There are four cloud servers $\{j_1, j_2, j_3, j_4\}$. The maximum processing power offered by cloud servers j_1 and j_2 is 1200 (CPU-hours), j_3 is 1500 (CPU-hours) and j_4 is 1000 (CPU-hours), respectively. The maximum storage capacity offered by cloud servers j_1 and j_4 is 1000 (GB), j_2 is 2000 (GB), and j_3 is 3000 (GB), respectively. The maximum network bandwidth offered by each cloud server j_1 is 33.33 (GB/sec), j_2 is 67.68 (GB/sec), j_3 is 48.84 (GB/sec), and j_4 is 50.25 (GB/sec), respectively. The unit costs of processing power charged by each cloud server j_1 is 1.280, j_2 is 1.320, j_3 is 1.150, and j_4 is 1.250 ($/CPU-hours), respectively. The unit costs of storage capacity charged by each cloud server j_1 is 0.930, j_2 is 1.200, j_3 is 1.250, and j_4 is 0.950 ($/MB), respectively. The unit costs of network bandwidth charged by each cloud server j_1 is 0.080, j_2 is 0.083, j_3 is 0.089, and j_4 is 0.090 ($/MB-sec), respectively.

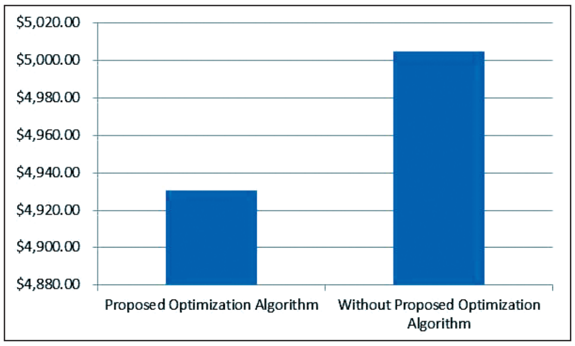

Figure 4: SDN-based cloud server coordination scheme.

The pre-defined parameters were used as input to the GAMS/CPLEX solver to run the experiment. The total cost with and without using the proposed algorithm is shown in Figure 4.

According to the result shown in Figure 4, the proposed optimization algorithm can address the scalability issue and provides a lower total cost than the algorithm without optimization.

7 Conclusions

In this paper, we proposed a scalable cloud based CDN framework by developing two schemes, i.e., UDP/TCP for content distribution from a content provider to cloud servers and SDN for cloud server coordination. Our framework is expected to offer various benefits of fast, reliable, and efficient content delivery networks.

Moreover, we have applied the optimization model of binary integer programming to obtain the optimal solution for the problem of content placement among multiple cloud servers to minimize the total cost of rented resources (storage, processing power, and internal bandwidth) for a content provider. According to the numerical results, the use of the optimization approach can save the budget of the content provider by optimally distributing contents to appropriate cloud servers. Then, we have evaluated the optimal solution for content placement by using the greedy algorithm and performing simulations.

The research direction for future work is to evaluate the performance of two schemes for the proposed framework by using simulation.

References

[1] Z. Xu, Y. Hu, and L. Bhuyan, "Efficient server cooperation mechanism in content delivery network," presented at the IEEE International Conference Performance Computing and Communications, Phoenix, USA, Apr. 10–12, 2006.

[2] Akamai, "Security, cloud delivery, performance," 2019. [Online]. Available: https://www.akamai.com/

[3] Limelight Network, "Content Delivery Network (CDN) service provider," 2019. [Online]. Available: https://www.limelight.com

[4] CoDeeN, "A CDN on PlanetLab," 2019. [Online]. Available: http://codeen.cs.princeton.edu/

[5] M. Wang, P. Jayaraman, R. Ranjan, and K. Mitra, "An overview of cloud-based content delivery networks: Research dimensions and state-of-the-art," in *Transactions on Large-Scale Data- and Knowledge-Centered Systems XX*, Berlin, Germany: Springer, 2015, pp. 131–158.

[6] V. Mathew, R. Sitaraman, and P. Shenoy, "Energy-efficient content delivery networks using cluster shutdown," in *Proceedings Green Computing*, 2013, pp. 1–10.

[7] R. Allan Alimi, L. Chen, D. Kutscher, H. Liu, A. Rahman, H. Song, R. Yang, D. Zhang, and N. Zong, "An open content delivery infrastructure using data lockers," in *Proceedings ICN Workshop on Information-centric Networking*, 2012, pp. 25–30.

[8] H. Xiong, X. Zhang, D. Yao, X. Wu, and Y. Wen, "Towards end-to-end secure content storage and delivery with public cloud," in *Proceedings Data and Application Security and Privacy*, 2012, pp. 257–266.

[9] S. Srinivasan, J. Lee, D. Batn, and H. Schulzrinne, "ActiveCDN: Cloud computing meets content delivery networks," Columbia University, New York, Computer Science Tech. Rep. CUCS-045-11, Nov. 2011.

[10] A. Blair, G. Parr, P. Morrow, B. Scotney, A. McConnell, S. Appleby, and M. Nilsson, "Cloud-based dynamically provisioned multimedia delivery: An elastic video endpoint," in *Proceedings Cloud Computing, GRIDs, and Virtualization*, 2012, pp. 260–265.

[11] E. He, J. Leigh, O. Yu, and T. Defainti, "Reliable blast UDP: Predictable high performance bulk data transfer," in *Proceedings Cluster Computing*, 2002, pp. 1–8.

[12] Y. Wu, Z. Zhang, C. Wu, C. Guo, Z. Li, and F. Lau, "Orchestrating bulk data transfers across geo-distributed datacenters," *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 112–125, Jan. 2015.

[13] M. P. Sarma, "Performance measurement of TCP and UDP using different queuing algorithm in high speed local area network," *International Journal of Future Computer and Communication*, vol. 2, no. 6, pp. 682–686, Dec. 2013.

[14] W. Han, H. Hu, and G. Ahn, "LPM: Layered policy management for software-defined networks," in *Proceedings 28th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*, 2014, pp. 1–9.

[15] GAMS, "Optimization Solvers," 2019. [Online]. Available: https://www.gams.com/optimization-solvers/

[16] A. Malik, N. Goyat, and V. Saroha, "Greedy algorithm: Huffman algorithm," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 7, pp. 296–303, Jul. 2013.