Server selection for heterogeneous cloud video services

He Chang

Follow this and additional works at: https://repository.hkbu.edu.hk/etd_oa

Recommended Citation
Chang, He, "Server selection for heterogeneous cloud video services" (2017). Open Access Theses and Dissertations. 419. https://repository.hkbu.edu.hk/etd_oa/419
DATE: August 31, 2017

STUDENT'S NAME: CHANG He

THESIS TITLE: Server Selection for Heterogeneous Cloud Video Services

This is to certify that the above student's thesis has been examined by the following panel members and has received full approval for acceptance in partial fulfillment of the requirements for the degree of Master of Philosophy.

Chairman: Dr. Shiu Wai Chee
Associate Professor, Department of Mathematics, HKBU
(Designated by Dean of Faculty of Science)

Internal Members: Dr. Tam Hon Wah
Associate Professor, Department of Computer Science, HKBU
(Designated by Head of Department of Computer Science)

Prof. Leung Yiu Wing
Professor, Department of Computer Science, HKBU

External Members: Dr. Wong Eric Wing-ming
Associate Professor
Department of Electronic Engineering
City University of Hong Kong

Issued by Graduate School, HKBU
Server Selection
for Heterogeneous Cloud Video Services

CHANG He

A thesis submitted in partial fulfillment of the requirements
for the degree of
Master of Philosophy

Principle supervisor:
Prof. Leung Yiu Wing (Hong Kong Baptist University)

August 2017
DECLARATION

I hereby declare that this thesis represents my own work which has been done after registration for the degree of MPhil at Hong Kong Baptist University, and has not been previously included in a thesis or dissertation submitted to this or any other institution for a degree, diploma or other qualifications.

I have read the University’s current research ethics guidelines, and accept responsibility for the conduct of the procedures in accordance with the University’s Committee on the Use of Human & Animal Subjects in Teaching and Research (HASC). I have attempted to identify all the risks related to this research that may arise in conducting this research, obtained the relevant ethical and/or safety approval (where applicable), and acknowledged my obligations and the rights of the participants.

Signature: Cheng He

Date: August 2017
Server selection is an important problem of cloud computing in which cloud service providers direct user demands to servers in one of the multiple data centers located in different geographical locations. The existing solutions usually assume homogeneity of cloud services (i.e., all users request the same type of service) and handle user demands in an individual basis which incurs high computational overhead. In this study, we propose a new and effective server selection scheme in which diversities of cloud services are taken into account. We focus on a specific cloud service, i.e., online video service, and assume that different videos have different bandwidth requirements. We group users into clusters and handle user demands on a cluster basis for faster and more efficient process.

Firstly, we assume that user demands and bandwidth capacities of servers are given in the data centers, our problem is to assign the user demands to the servers under the bandwidth constraint, such that the overall latency (measured by the network distance) between the user clusters and the selected servers is minimized. We design a server selection system and formulate this problem as a linear programming formulation which can be solved by existing techniques. The system periodically executes our scheme and computes an optimal solution for server selection. User demands are assigned to the servers according to the optimal solution and the minimum overall latency can be achieved. The
simulation results show that our scheme is significantly better than the random algorithm and the YouTube server selection strategy.

Based on the first part, we take the storage capacities of servers constraint into consideration. In the second part, our new problem is to assign the user demands to the servers under the bandwidth and storage constraint, such that the function of overall latency (measured by the network distance) between the user clusters and the selected servers and standard deviation of traffic load of every server in the system is minimized. We design a server selection system and formulate this problem which can be solved by existing techniques. User demands are assigned to the servers according to the optimal solution and the two goals (minimum overall latency and the most balanced traffic load) can be achieved. The simulation results show the influence of different weights of these two goals on the user demands assigning.
Acknowledgements

I would like to thank my principle supervisor, Prof. Yiu-Wing Leung deeply, for his great support throughout my MPhil study at the Hong Kong Baptist University. Without his careful guidance and great encouragement, this thesis would never have been possible.

I would like to thank my co-supervisor, Dr. Hai Liu. At first, I didn’t know the research at all. Dr. Hai Liu taught me how to select helpful materials and how to form own research ideas. And he helps me modify my every research paper.

I would like to thank my co-supervisor, Dr. Xiaowen Chu. He gave me many insightful comments and suggestions.

I would also like to thank my many friends at HKBU. Lu Yu gave me many help suggestions in my research. Qin Bao helped me in many ways to adapt to life in Hong Kong.

Finally, I would like to thank my family especially my mother’s love and support of my study and research.
# Table of Contents

Declaration

Abstract

Acknowledgements

Table of Contents

List of Tables

List of Figures

Chapter 1    Introduction

1.1 Cloud Computing

1.2 Motivation

1.3 Main Contributions

1.4 Thesis Outline

1.5 Notations

Chapter 2    Background

2.1 A Brief Introduction of Server Selection in Cloud Computing

2.2 Some Existing Server Selection Approaches in Cloud Computing Environments
Chapter 3  Minimum Latency Server Selection for Heterogeneous Cloud Services

3.1 Introduction

3.2 System Design and Models

3.3 Optimal Server Selection

3.4 Numerical Examples

3.5 Simulation

3.6 Conclusion

Chapter 4  Two goals Server Selection for Heterogeneous Cloud Services

4.1 Introduction

4.2 System Design and Models

4.3 Problem Formulation of Server Selection

4.4 Numerical Examples

4.5 Simulation

4.6 Conclusion

Chapter 5  Conclusions and Future Work

Bibliography

CURRICULUM VITAE
List of Tables

1.1 Notations

2.1 Differences between Our Problem and the Existing Server Selection Problems

3.1 Notations in Minimum Latency Model

3.2 Bandwidth Requirements of 5 Videos

3.3 Access Probabilities of 5 Videos in 2 User Clusters

3.4 Network Distance between 2 User Clusters and 3 Servers

3.5 9 Groups of Bandwidth Capacities of Servers

3.6 Server Selection Solution ($Q_{U_tV_jS_t}$) for Group No.1

3.7 Three Types of Videos In YouTube

4.1 Notations in Two Goals Model

4.2 Bandwidth and Capacity Requirements of 5 Videos

4.3 Access Probabilities of 5 Videos in 2 User Clusters

4.4 Storage Capacity of Every 3 Server

4.5 Network Distance between 2 User Clusters and 3 Servers

4.6 9 Groups of Bandwidth Capacities of Servers

4.7 Server Selection Solution ($Q_{U_tV_jS_t}$) for Group No.1

4.8 Server Selection Solution ($Q_{U_tV_jS_t}$) for Group No.2

4.9 Server Selection Solution ($Q_{U_tV_jS_t}$) for Group No.3

4.10 Server Selection Solution ($Q_{U_tV_jS_t}$) for Group No.4

4.11 Server Selection Solution ($Q_{U_tV_jS_t}$) for Group No.5
4.12 Server Selection Solution \( (Q_{U,t,V_j,S_t}) \) for Group No.6

4.13 Server Selection Solution \( (Q_{U,t,V_j,S_t}) \) for Group No.7

4.14 Server Selection Solution \( (Q_{U,t,V_j,S_t}) \) for Group No.8

4.15 Server Selection Solution \( (Q_{U,t,V_j,S_t}) \) for Group No.9

4.16 Three Types of Videos in YouTube

4.17: The execution time from Fig.4.1 to Fig.4.3
List of Figures

3.1 System Architecture
3.2 Sum of Access Probabilities of the 5/10 Most Popular Videos with Increase of $\alpha$
3.3 Percentages of Total Workload of Servers
3.4 Overall Latency VS Bandwidth Capacities of Servers
3.5 Mean of Overall Latency VS Skew Factor $\alpha$
3.6 Mean of Overall Latency VS Number of Servers
4.1 Mean of Overall Latency, Normalized $F_2$ and Normalized $F$ VS Skew Factor $\alpha$
4.2 Mean of Overall Latency, Normalized $F_2$ and Normalized $F$ VS Number of Servers
4.3 Mean of Overall Latency, Normalized $F_2$ and Normalized $F$ VS $w_1$
Chapter 1

Introduction

Many measurement studies reveal that the Internet is increasingly a platform for various online services such as online video, online game and social network, distributed across multiple locations for better reliability and performance. With the development of network technologies, cloud computing provides a stable and flexible infrastructure for various online applications such as cloud storage, web searching, online gaming, online video, etc.

In the following, we introduce some basic concepts about cloud computing, firstly. Then, we present our motivation and summarize the main contributions of this thesis. At last, we outline the organization of the whole thesis and list some commonly used notations in this thesis.

1.1 Cloud Computing

Cloud Computing is a synonym for distributed computing over a network, and means the ability to run a program or application on many different geographic computers at the same time. In terms of architectures, many current researchers classify various cloud paradigms seem to converge to layered “everything as a service” (XaaS).

1) Software as a Service (SaaS): This layer is composed by all applications which run on the cloud and provide a direct service the customer. Based on the application level offered, this layer could be further subdivided. On top
layer, we have the actual applications which are basically the final service provided to an end customer, such as Google Docs or Microsoft Office Live. Obviously, they can be made up of lower layer services: Lenk et al.[1] classify them further into basic and composite application services.

2) **Platform as a Service (PaaS):** Customers of PaaS are offered with an application or development platform, which makes them create SaaS application/services. We can further decompose the PaaS layer into programming environments and execution environments. The programming environments provide programming-language-level environment and the execution environments could take care of automatic scaling and load balancing.

3) **Infrastructure as a Service (IaaS):** The lowest level offers the basic resources, i.e., storage, computing, and network, which PaaS/SaaS depend on. SaaS applications do not necessarily depend on the intermediate PaaS layer. The “resources” can refer to physical resources like servers, but these often are virtualized.

1.2 Motivation

With the development of network technologies, cloud computing provides a stable and flexible infrastructure for various online applications such as cloud storage, web searching, online gaming, online video, etc. In a typical framework of cloud computing, cloud service providers (CSP) manage datacenter hardware and software, and provide various services to cloud users all over the world. Due to the increasing demands for large-scale computing, modern data center architectures usually adopt multiple data centers which are geographically located in multiple regions [2]. With such distributed architecture, CSP is faced with the server
selection problem that how to allocate user demands to computing nodes of data centers (referred to as “servers”) located at various geo-locations [3]. To achieve high service quality, users are expected to be served by geographically close servers so as to decrease end-to-end latency. At the same time, CSP should ensure load-balancing among the servers. Otherwise, some popular servers may become overloaded which in turn degrades the quality of service to the users.

Basically, there are two approaches to the server selection problem in cloud computing environments [3]. The first is to grant the selection rights to users in selecting desired servers (e.g., [4]). With this approach, users usually select the closest servers for low latency service. However, this approach is lack of global scheduling and optimization since each user individually selects its desired server, which consequently harms the workload balance among the servers. The other approach is to let cloud providers allocate servers to users. This approach does not consider diversities of cloud services and make server selections for every single user.

In fact, there is a wide variety of cloud services and these services have distinct features. For instance, cloud storage is relatively less sensitive to latency while online gaming has strict latency requirement. E-mail and instant messaging have fewer requirements for the bandwidth while online video is eager for large bandwidth. Even for the same type of service, the user requirements could be different. Taking online video service as an example, high-definition videos require much more bandwidth than low-definition videos do.

User demands are handled on a cluster basis for faster and more efficient process. Users in the same clusters are assumed to have similar behavior or similar latency to cloud servers. For instance, the users in a geographic region (e.g.,
people in Miami) may be more interested in specific videos (e.g., NBA game of Miami Heat) than users in other areas. Moreover, users with the same IP prefix (e.g., coming from a same network) may have similar latency to access servers of a data center. In fact, this model is a generalization of existing ones [3] [5-7] where each user demand is individually handled by the system, given that each user can be treated as a single cluster.

To sum up, we hope to investigate a new server selection problem in cloud computing environments which makes server selections considering diversities of cloud services and correlation of users and group users into clusters.

1.3 Main Contributions

In this work, we investigate a new server selection problem in cloud computing environments which differs from existing problems in the following aspects.

1) We consider diversities of cloud services.

As we mentioned in the previous part, there is a wide variety of cloud services and these services have distinct features such as latency requirement and bandwidth requirement. In this thesis, we take online video service which has become one of the most popular applications in the Internet and it holds approximately 86% of global network traffics [8] as an example to consider the influence of diversities of online videos’ bandwidth requirement on the server selection results. The proposed solution in this work is applicable to general cases of cloud services.

2) We consider correlation of users and group users into clusters.

We group users into clusters and handle user demands on a cluster basis. We design a server selection system which interconnects video service provider (VSP),
Internet service provider (ISP) and end users. The system receives demands from clustered users and assigns the demands to the servers, such that the overall latency (measured by the network distance) between the user clusters and the selected servers and standard deviation of traffic load of every server in the system is minimized.

The contribution of this work is three-fold.

1) Generalized model: We consider diversities of cloud services in our problem. This model generalizes the model adopted in existing solutions which considers only homogeneous cloud services.

2) Efficient system: Our system handles user demands on a cluster basis, instead of on an individual basis. The users with similar behaviors/features could be grouped and are served with the same servers so as to improve the process efficiency and save the computational cost, especially in serving large-scale users.

3) Optimal solution: We formulate the server selection problem as a linear programing formulation which can be solved by existing techniques. The proposed scheme can achieve the minimum overall latency in serving users under the capacity constraint of the servers. The simulation results show that our solution is significantly better than the random algorithm and the YouTube server selection strategy.

1.4 Thesis Outline

The rest of this thesis is organized as follows:

Chapter 2 introduces some background knowledge, including a brief introduction of server selection in cloud computing and some existing server selection approaches in cloud computing environments.
Chapter 3 gives details of our server selection system design and models. And this chapter gives the solution of minimum latency server selection for heterogeneous cloud services. In this chapter, user demands and bandwidth capacities of servers in the data centers are given. Our problem is to assign the user demands to the servers under the bandwidth constraint, such that the overall latency (measured by the network distance) between the user clusters and the selected server is minimized. Then, we discuss the time complexity of our solution. Finally, we give the numerical examples and simulations of this problem.

Chapter 4 gives the solution of two goals server selection for heterogeneous cloud services. In this chapter, user demands, bandwidth and storage capacities of servers in the data centers are given. Our problem is to assign the user demands to the servers under the bandwidth and storage constraint, the function of overall latency (measured by the network distance) between the user clusters and the selected servers and standard deviation of traffic load of every server in the system is minimized. Then, we give the numerical examples and simulations of this new problem.

Chapter 5 concludes the thesis and proposes several potential directions for future research.
### 1.5 Notations

Some commonly used notations in this thesis are listed in Table 1.1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Physical Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U = {U_1, U_2 \ldots U_k}$</td>
<td>Set of given user clusters</td>
</tr>
<tr>
<td>$k$</td>
<td>Number of user clusters</td>
</tr>
<tr>
<td>$</td>
<td>U_l</td>
</tr>
<tr>
<td>$S = {S_1, S_2 \ldots S_i \ldots S_n}$</td>
<td>Set of servers</td>
</tr>
<tr>
<td>$n$</td>
<td>Number of servers</td>
</tr>
<tr>
<td>$B_{S_i}$</td>
<td>Available bandwidth capacity of $S_i$</td>
</tr>
<tr>
<td>$C_{S_i}$</td>
<td>Available storage capacity of $S_i$</td>
</tr>
<tr>
<td>$L_{S_i}$</td>
<td>Traffic load of $S_i$</td>
</tr>
<tr>
<td>$V = {V_1, V_2 \ldots V_j \ldots V_m}$</td>
<td>Set of videos in the system</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of videos</td>
</tr>
<tr>
<td>$B_{V_j}$</td>
<td>Bandwidth requirement of $V_j$</td>
</tr>
<tr>
<td>$C_{V_j}$</td>
<td>Storage size of $V_j$</td>
</tr>
<tr>
<td>$A_{V_j}^{&lt;l&gt;}$</td>
<td>Access probability of video $V_j$ in user cluster $U_l$</td>
</tr>
<tr>
<td>$P_{V_j}^{&lt;l&gt;}$</td>
<td>Expected popularity of video $V_j$ in user cluster $U_l$</td>
</tr>
<tr>
<td>$D_{U_lS_i}$</td>
<td>Network distance between user cluster $U_l$ and server $S_i$</td>
</tr>
<tr>
<td>$T_{V_jS_i}$</td>
<td>Decision function whether put video $V_j$ in server $S_i$ or not</td>
</tr>
<tr>
<td>$Q_{U_lV_jS_i}$</td>
<td>Proportion of user demands from cluster $U_l$ asking for video $V_j$ that is assigned to server $S_i$</td>
</tr>
<tr>
<td>$F_1$</td>
<td>Overall latency (measured by the network distance) between the user clusters and the selected servers</td>
</tr>
<tr>
<td>$F_2$</td>
<td>Standard deviation of traffic load of every server in the system</td>
</tr>
<tr>
<td>$w_1$</td>
<td>Weight of $F_1$</td>
</tr>
<tr>
<td>$w_2$</td>
<td>Weight of $F_2$</td>
</tr>
</tbody>
</table>
Chapter 2

Background

In this chapter, we briefly introduce some background knowledge of server selection in cloud computing and some existing server selection approaches in cloud computing environments.

2.1 A Brief Introduction of Server Selection in Cloud Computing

Cloud computing provides a stable and flexible infrastructure for various online applications such as cloud storage, web searching, online gaming, online video, etc. In a typical framework of cloud computing, cloud service providers (CSP) manage datacenter hardware and software, and provide various services to cloud users all over the world. Due to the increasing demands for large-scale computing, modern data center architectures usually adopt multiple data centers which are geographically located in multiple regions [2]. With such distributed architecture, CSP is faced with the server selection problem that how to assign user demands to computing nodes of data centers (referred to as “servers”) located at various geo-locations [3]. To achieve high service quality, users are expected to be served by geographically close servers so as to reduce the end-to-end latency. At the same time, CSP should ensure load-balancing among the servers. Otherwise, some popular servers may become overloaded which in turn degrades the quality of service to the users.
The server selection problem is different from the resource placement problem (e.g., [9] [10]). After the resources (e.g., servers and services) have been placed, the server selection starts and directs user demands to the servers.

2.2 Some Existing Server Selection Approaches in Cloud Computing Environments

To the best of our knowledge, there are two approaches to the server selection problem in cloud computing environments [3].

1) The first is to grant the selection rights to users in selecting desired servers (e.g., [4]). With this approach, users usually select the closest servers for low latency service. However, this approach is lack of global scheduling and optimization since each user individually selects its desired server, which consequently harms the workload balance among the servers.

2) The other approach is to let cloud providers allocate servers to users.

The largest centralized CDN (content distribution network), Akamai, was studied in [5]. In Akamai, an end user sends a service request to the local DNS (domain name service). The DNS analyzes the domain name and tries to find the server which meets the user’s demand and is likely to be close to the user.

Authors in [6] proposed a decentralized server (replica) selection system, in which requests of cloud users (clients) are handled by multiple mapping nodes (e.g., HTTP ingress proxies). Given the bandwidth capacities of servers, the mapping nodes route the user demands to appropriate servers so as to minimize the cost which is defined as the overall latency between the users and the servers. Each mapping node performs a local optimization based on its user demands. The local decisions of all the mapping nodes iteratively converge to a global optimum in the end.
A similar decentralized system was proposed in [3], where inter-domain transit traffic is taken into account. If a user and its selected server are within the same geographical location, the cost function of [3] is the same as that of [6]. Otherwise, it introduces a penalty, i.e., coefficient $k$, to the cost function so as to limit the inter-domain transit traffic in the minimum level.

V.K. Adhikari et al. [7] studied server selection strategy in YouTube. It shows that YouTube uses a hash-based policy to perform video server selection by analyzing DNS resolution and video playback traces. First, YouTube maps every video-ID to a unique hostname in every namespace of the hierarchical DNS based on the hash policy. Then it maps each DNS hostname to an IP address (i.e., a physical video cache) based on user geographical location and current video demand. During “normal” hours, YouTube redirects users to the graphically closest server. If this server has too many popular videos (i.e., “busy” hours), it redirects the user to a slightly farther location in order to balance the load on different servers.

H. Xu et al. [11] investigated a problem of joint request mapping and response routing with distributed servers. The problem was formulated as a general workload management optimization with a utility function of the overall latency and the electricity and bandwidth costs. A distributed algorithm was developed to solve this large-scale optimization.

Table 2.1 shows the difference between our problem and the existing server selection problems.
Table 2.1 Differences between Our Problem and the Existing Server Selection Problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Centralized /Decentralized</th>
<th>Consider the diversities of cloud services?</th>
<th>Handle user demands on a cluster basis?</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our problem</td>
<td>Centralized</td>
<td>YES</td>
<td>YES</td>
<td>Objective: minimize the overall latency between the user clusters and the selected servers</td>
</tr>
<tr>
<td>[5]</td>
<td>Centralized</td>
<td>NO</td>
<td>NO</td>
<td>Analyze the centralized hierarchical stable-marriage solution to match clients with CDN servers in Akamai</td>
</tr>
<tr>
<td>[7]</td>
<td>Centralized</td>
<td>NO</td>
<td>NO</td>
<td>Analyze DNS resolutions and video playback traces collected by half a million YouTube videos</td>
</tr>
<tr>
<td>[6]</td>
<td>Decentralized</td>
<td>NO</td>
<td>NO</td>
<td>Objective: minimize the overall latency between the users and the selected servers</td>
</tr>
<tr>
<td>[3]</td>
<td>Decentralized</td>
<td>NO</td>
<td>NO</td>
<td>Objective: minimize the overall latency between the users and the selected servers</td>
</tr>
<tr>
<td>[11]</td>
<td>Decentralized</td>
<td>NO</td>
<td>NO</td>
<td>Objective: minimize the overall latency in serving the request and the electricity and bandwidth costs</td>
</tr>
</tbody>
</table>
Chapter 3

Minimum Latency Server Selection for Heterogeneous Cloud Services

3.1 Introduction

This chapter is mainly composed of my previous publication [31].

In this chapter, we propose a new and effective server selection scheme in which diversities of cloud services are taken into account. We focus on a specific cloud service, i.e., online video service, and assume that different videos have different bandwidth requirements. We group users into clusters and handle user demands on a cluster basis for faster and more efficient process. Given user demands and bandwidth capacities of servers in the data centers, our problem is to assign the user demands to the servers under the bandwidth constraint, such that the overall latency (measured by the network distance) between the user clusters and the selected servers is minimized.

3.2 System Design and Models

For ease of understanding, we focus on a specific cloud service, i.e., online video service. We consider diversities of video services and assume that different videos have different bandwidth requirements. Notice that our system and the proposed scheme in this work can be easily extended to other types of cloud services.

A. Overview of Our System
Our system architecture is illustrated in Figure 3.1. There are three entities involved: video service provider (VSP), Internet service provider (ISP) and end users. Our system logically interconnects these three entities. The VSP holds multiple data centers (referred to as servers for simplicity) located in different geographical locations, and provides various videos to end users via our system as a middle layer. Based on ISP information (e.g., IP addresses [12] [13]), the system classifies the end users into several clusters and handles user demands accordingly for each cluster. Our system estimates the latency, in terms of the network distance [3] [14], between the servers and the clusters. The VSP provides real-time video information (e.g., bandwidth requirements and popularities of videos) and workload information (e.g., available bandwidth of servers) which are needed in
the sever selection module of our system. Users may request any videos that are provided by the VSP. For each cluster’s demand asking for the same video, the system performs sever selection and computes an optimal distribution among the servers based on the workload information, the video information and the latency information. For instance, the end users in user cluster $U_1$ asking for video $V_1$ send request $(U_1, V_1)$ to the system. The system computes an optimal distribution among the servers where $(U_1, V_1, S_1) = 40\%$ implies that 40% of the demands from user cluster $U_1$ asking for video $V_1$ should be directed to server $S_1$. We discuss each component of our system in details in the rest of this section.

B. ISP Information for User Classification

The ISP provides Internet connection for the end users and the VSP. The ISP information could be used for 1) classification of the end users, and 2) estimation of latency between the end users and the servers. We group users into clusters such that the users in a same cluster have the similar latency to a remote server. We introduce several sample ISP information for classification of the end users.

**Example 1.** End users can be divided into clusters according to their geographical regions/locations. For instance, the system can obtain IP addresses from the ISP database and determine the corresponding geographical region information (e.g. which city, which state, and which country) by the construction and maintenance of a location table [12] [13]. By analyzing the YouTube server selection strategy, authors in [7] pointed out that in most cases, the closeness in terms of network latency is highly related to the closeness in geographical distance. That is, the users in the same geographical region have the similar latency to the servers far away. Moreover, these users may have similar preference in watching videos. For example, people in Miami may be more interested in the videos of
NBA game of Miami Heat than people in other regions, and users coming from China may prefer the videos which are delivered in Chinese.

**Example 2.** The end users in the same Autonomous System (AS) can be grouped into a cluster. An AS is defined as a collection of connected IP routing prefixes under the control of one or more network operators which can decide to adopt routing protocols in its domain independently [15]. An AS can be a simple network or network groups under the control of one or more network operators. It was pointed out in [14] that some close IP hosts, in terms of both geographical locations and latency, may belong to different ASs, while some IP hosts that are very far apart may belong to the same AS. There are about 42000 ASs in late 2012 [16]. We can assume that the hosts in the same AS have the similar latency to the other hosts outside [14].

**Example 3.** End users associated with the same ISP can be grouped into a cluster. It is because that the users with the same ISP usually share the same gateways and the same topological routes, and experience the same congestion situations. Thus, these users have the same/similar latency to access the servers far away if they choose the same service package. For example, the users using AOL service can be grouped into a cluster.

**C. ISP Information for Latency Estimation**

Our system adopts the concept of *network distance* [14] [17] to measure the latency between end users and servers. Network distance is a measurement technique of estimating the latency between arbitrary two Internet hosts which cannot be directly measured by ping or traceroute because of high overhead [14]. Current methods of the network distance can be divided into infrastructure based methods (e.g., [14]) and coordinate based methods (e.g., [3] [17] [17-20]). With
the infrastructure based methods, hosts are aggregated according to their IP prefixes and some measurement infrastructures called *Tracers* are placed such that each cluster of aggregated hosts is close to one or more tracers. The tracers measure distance between each other and to their nearest host clusters, which are used to determine the distance between any pairs of host clusters. The coordinate based methods usually select a small set of hosts as landmarks which calculate their coordinates by minimizing the overall discrepancy between the measured distances and the computed distances. A host measures its latencies to the landmarks using ping and determines its own coordinate relative to the landmarks’ coordinates. The network distance is estimated by a function of the coordinates (e.g., [17]).

Considering the extra cost of measurement infrastructures (e.g., tracers) in the infrastructure based methods, we propose to use the coordinate based methods to estimate latency in this work. Notice that there are two roles, i.e., servers and end users, involved in our system. Works in [16] [18-21] consider only homogeneous hosts and thus the solutions therein are not suitable to our system. We employ a two-layer network distance estimation method proposed in [3]. It assumes two layers of the system: cloud layer and client layer. Accordingly, the system [3] includes an intra-cloud network coordinate system and a client-cloud network coordinate system. The intra-cloud network coordinate system computes coordinates of servers (i.e., cloud nodes) and measures the latency between any two servers by using the method in [21]. The client-cloud network coordinate system selects a subset of the servers as the landmarks to position the clients by using the matrix-factorization based method in [17]. Coordinate of either a server or a client is represented by an outgoing vector and an incoming vector. Once the
coordinates of the servers and the clients are determined, the latency between any
server and any client can be estimated as the production of their two vectors.

D. Video Information for Popularity Estimation

Let \( V = \{V_1, V_2, \cdots, V_m\} \) denote the set of \( m \) videos provided by the VSP.
Compared with bandwidth, storage space is usually not a bottleneck in the modern
data centers [22] [23]. So, we assume that all the videos are replicated in every
data center. For each video \( V_j, j=1, 2, \ldots, m \), we consider two attributes \( B_{V_j} \) and
\( P_{V_j} \), where \( B_{V_j} \) is the bandwidth requirement of video \( V_j \) and \( P_{V_j} \) is the expected
popularity of video \( V_j \) which is calculated as the expected request frequency of
video \( V_j \) from the users in a unit time. If an end user asking for video \( V_j \) is served
by a server, the available bandwidth of the server should be no less than \( B_{V_j} \) so as
to guarantee the smooth delivery of video \( V_j \) to the user [23].

Notice that the expected popularity of video \( V_j \) (i.e., \( P_{V_j} \)) could be different in
different user clusters. Let \( U = \{U_1, U_2, \cdots, U_k\} \) denote the set of user clusters
where \( U_l \) denotes the set of end users in cluster \( l, l=1, 2, \ldots, k \). Let \( P_{V_j}^{<l>} \) be the
expected popularity of video \( V_j \) in user cluster \( U_l \). That is, \( P_{V_j}^{<l>} \) denotes the
expected request frequency of video \( V_j \) from the users in user cluster \( U_l \) in a unit
time. To estimate \( P_{V_j}^{<l>} \), we introduce another variable \( A_{V_j}^{<l>} \) which denotes the
access probability of video \( V_j \) in user cluster \( U_l \). Obviously, we have

\[
P_{V_j}^{<l>} = A_{V_j}^{<l>} \times |U_l|, \quad (1)
\]

where \( |U_l| \) is the number of users in cluster \( U_l \). We assume that the access
probability of a video follows the Zipf’s distribution [24] :

\[
A_{V_j}^{<l>} = c/j^\alpha, \quad (2)
\]
where $c$ is a normalization constant and $\alpha$ is a skew factor. For a given video $V_j$ and a user cluster $U_t$, we have
\[
\sum_{j=1}^{m} A_{V_j}^{<t>} = 1. \tag{3}
\]

We suppose that there are 100 videos in the system. Figure 3.2 shows the sum of access probabilities of the 5 most popular videos and the sum of access probabilities of the 10 most popular videos as parameter $\alpha$ varies in range of $0.0 \leq \alpha < 2.0$. We can see that if $\alpha$ is larger, the distribution of the video’s popularities becomes more skewed. That is, some videos are much more popular than others if $\alpha$ is large. If $\alpha$ is small, popularities of the videos are more or less the same. Estimation of parameter $\alpha$ is as follows.

We first obtain historic data of $P_{V_j}^{<t>}$ from the system, and then calculate $A_{V_j}^{<t>}$ according to equation (1). Once $A_{V_j}^{<t>}$ are computed, parameter $\alpha$ can be estimated according to equations (2) and (3). We use the determined Zipf’s distribution in equation (2) to generate $A_{V_j}^{<t>}$ which is subsequently used to calculate the expected popularity of video $P_{V_j}^{<t>}$ with equation (1). Notice that both video information $B_{V_j}$ and $P_{V_j}^{<t>}$ should be periodically updated in the system.
Figure 3.2 Sum of Access Probabilities of the 5/10 Most Popular Videos with Increase of $\alpha$

E. Workload Information

We define $S = \{S_1, S_2, \ldots, S_i, \ldots, S_n\}$ as the set of given servers. As the development of the storage technologies, VSPs could spend less money to buy enough storage capacity [22]. Hence, we assume that the storage capacity of every server is sufficient and consider the bandwidth capacities of servers as the main bottleneck of servers’ workloads [23]. Each server $S_i$ has capacity limit, say $B_{S_i}$, which indicates the bandwidth capacity of server $S_i$. VSPs periodically update workload information $B_{S_i}$ which will be used for the optimal server selection in the next section.

3.3 Optimal Server Selection

In our system, the server selection module assigns demands of user clusters to the servers. The goal of our system is to minimize the overall latency, measured by
the network distance, between the user clusters and the selected servers. We consider the server workload, i.e., available bandwidth, as the main constraint because that overloaded servers cannot provide expected services to the users and unbalance workload of servers may further cause the risk of system breaking down [3]. In the system architecture shown in Figure 2.1, the optimal server selection is made based on the workload information, the video information for popularity estimation, and the ISP information for latency estimation. Specifically, the server selection module computes an optimal distribution of user demands among the servers, based on the following information.

1) Workload of server \( S_i \), denoted by \( B_{S_i} \), \( i=1, 2, ..., n \).

2) Expected popularity of video \( V_j \) in cluster \( U_l \), denoted by \( P_{V_j}^{<l>} \), \( j=1, 2, ..., m \), \( l=1, 2, ..., k \).

3) Network distance between user cluster \( U_l \) and server \( S_i \), denoted by \( D_{U_l,S_i} \), \( l=1, 2, ..., k \), \( i=1, 2, ..., n \).

First, we give the notations used in our model in Table 3.1.

Let \( Q_{U_l,V_j,S_i} \) be the proportion of user demands from cluster \( U_l \) asking for video \( V_j \) that is assigned to server \( S_i \). \( Q_{U_l,V_j,S_i} \) is a ratio in range \([0, 1]\). For example, if \( Q_{U_l,V_j,S_i} = 0.3 \), it implies that 30% of the demands from cluster \( U_l \) asking for video \( V_j \) are served by server \( S_i \). If \( Q_{U_l,V_j,S_i} = 1 \), it means that server \( S_i \) serves all the demands from cluster \( U_l \) asking for video \( V_j \).

The problem of server selection can be formulated as the following.

Minimize \( \sum_{i=1}^{k} \sum_{j=1}^{m} \sum_{l=1}^{n} Q_{U_l,V_j,S_i} \cdot D_{U_l,S_i} \cdot P_{V_j}^{<l>} \) \hspace{1cm} (4)

Subject to

\( \sum_{l=1}^{m} B_{V_j} \cdot (\sum_{i=1}^{k} Q_{U_l,V_j,S_i} \cdot P_{V_j}^{<l>}) \leq B_{S_i} \), \( i=1, 2, ..., n \); \hspace{1cm} (5)
\[
\sum_{l=1}^{n} Q_{U_l V_j S_i} = 1, l=1, 2, \ldots, k, j=1, 2, \ldots, m. \quad (6)
\]

\[
Q_{U_l V_j S_i} \leq 1, \forall i, j, l. \quad (7)
\]

Table 3.1 Notations in Minimum Latency Model

<table>
<thead>
<tr>
<th>Notation</th>
<th>Physical Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>User clusters</td>
<td>(U = {U_1, U_2 \cdots U_1 \cdots U_k}) Set of given user clusters</td>
</tr>
<tr>
<td></td>
<td>(k) Number of user clusters</td>
</tr>
<tr>
<td></td>
<td>(</td>
</tr>
<tr>
<td>Servers</td>
<td>(S = {S_1, S_2 \cdots S_i \cdots S_n}) Set of servers</td>
</tr>
<tr>
<td></td>
<td>(n) Number of servers</td>
</tr>
<tr>
<td></td>
<td>(B_{S_i}) Available bandwidth capacity of (S_i)</td>
</tr>
<tr>
<td>Videos</td>
<td>(V = {V_1, V_2 \cdots V_j \cdots V_m}) Set of videos in the system</td>
</tr>
<tr>
<td></td>
<td>(m) Number of videos</td>
</tr>
<tr>
<td></td>
<td>(B_{V_j}) Bandwidth requirement of (V_j)</td>
</tr>
<tr>
<td></td>
<td>(A_{V_j}^{&lt;\ell&gt;}) Access probability of video (V_j) in user cluster (U_l)</td>
</tr>
<tr>
<td></td>
<td>(P_{V_j}^{&lt;\ell&gt;}) Expected popularity of video (V_j) in user cluster (U_l)</td>
</tr>
<tr>
<td></td>
<td>(D_{U_l S_i}) Network distance between user cluster (U_l) and server (S_i)</td>
</tr>
<tr>
<td></td>
<td>(Q_{U_l V_j S_i}) Proportion of user demands from cluster (U_l) asking for video (V_j) that is assigned to server (S_i)</td>
</tr>
</tbody>
</table>

The above formulation is a typical linear programming formulation which can be solved by existing techniques such as the simplex algorithm and the LPSOLVE algorithm in [25]. The item in (4) is the overall latency between the user clusters and the servers. Inequation (5) implies that the total bandwidth that each server consumes cannot exceed its available bandwidth capacity. Equation (6) implies that all demands from cluster \(U_l\) asking for video \(V_j\) should be eventually satisfied by the servers. We assume that the total bandwidth capacities of the servers are sufficient to serve the users.

**Theorem 1:** The average time complexity of our solution is \(O((nk)^{5.5}(\log Z)^2))\), where \(n\) is the number of servers, \(k\) is the number of user
clusters, \( m \) is the number of videos, and \( Z \) is the largest coefficient in LP formulation (4)-(7).

**Proof:** Parameters \( D_{U_iS_i}, B_{V_j}, B_{S_i} \) and \( P_{V_j}^{<t>} \) should be determined to solve the LP formulation (4)-(7). Each of \( D_{U_iS_i}, B_{V_j} \) and \( B_{S_i} \) can be obtained in constant time, and thus the time complexity of computing all parameters \( D_{U_iS_i}, B_{V_j} \) and \( B_{S_i} \) is \( O(kn), O(m) \) and \( O(n) \), respectively. According to equations (1)-(3), each \( P_{V_j}^{<t>} \) can be computed in \( O(m) \). So, the total time to compute all \( P_{V_j}^{<t>} \) is \( kmO(m) = O(km^2) \).

The LPSOLVE algorithm is adopted to solve this linear programming problem. The complexity of this algorithm is \( O(N^{3.5}L^2) \) where \( N \) is dimension of the problem and \( L \) is the number of bits in the input [25]. In our formulation, \( N = nkm \) and \( L \sim O(kmn \times \log Z) \) where \( Z \) is the largest coefficient in LP formulation (4)-(7). Hence, the time complexity of our solution is

\[
O(kn)+O(m)+O(n)+O(km^2)+O((nkm)^{5.5}(\log Z)^2) \\
= O((nkm)^{5.5}(\log Z)^2)).
\]

Once all variables \( Q_{U_iV_jS_i} \) are determined, server selection module assigns user demands according to the optimal solution and the minimum overall latency is achieved. Notice that the workload information, the latency information and the expected popularities of videos should be periodically updated and the LP formulation is re-computed by the server selection module.

### 3.4 Numerical Examples

In this section, we present numerical examples for illustration of our server selection scheme. In our example, the number of user clusters is set to \( k = 2 \), and the numbers of users in the two clusters are set to \( |U_1| = 20 \) and \( |U_2| = 100 \),
respectively. We suppose there are five videos in the system (i.e., \( m = 5 \)) and bandwidth requirements of the five videos are shown in Table 3.2. For simplicity, we assume that the skew factor of equation (2) \( \alpha \) is estimated to be 0.7 based on the historic data. According to the model in equation (2), access probabilities of the five videos are calculated and shown in Table 3.3. Table 3.4 shows the network distances between 2 user clusters and 3 servers which are random numbers in [0, 100]. To ensure that there is a feasible solution, we set

\[
B_{S_1} = \sum_{j=1}^{5} B_{V_j} \left( \sum_{i=1}^{2} P_{V_j} \right) = 2.4 \times 10^5 \text{ kbps}. \tag{8}
\]

That is, bandwidth capacity of server \( S_1 \) is sufficient to serve all user demands. We consider nine groups of bandwidth capacities of servers, as shown in Table 3.5. For example, in group No.1, \( B_{S_1} = 2.4 \times 10^5 \text{ kbps} \) and \( B_{S_2} = B_{S_3} = 0.8 \times 10^5 \text{ kbps} \).

Table 3.2 Bandwidth Requirements of 5 Videos

<table>
<thead>
<tr>
<th>( V_j )</th>
<th>( V_1 )</th>
<th>( V_2 )</th>
<th>( V_3 )</th>
<th>( V_4 )</th>
<th>( V_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B_{V_j} ) (kbps)</td>
<td>728</td>
<td>1138</td>
<td>5362</td>
<td>736</td>
<td>3021</td>
</tr>
</tbody>
</table>

Table 3.3 Access Probabilities of 5 Videos in 2 User Clusters

<table>
<thead>
<tr>
<th>( V_j )</th>
<th>( U_1 )</th>
<th>( U_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_1 )</td>
<td>0.136203</td>
<td>0.359441</td>
</tr>
<tr>
<td>( V_2 )</td>
<td>0.221262</td>
<td>0.116506</td>
</tr>
<tr>
<td>( V_3 )</td>
<td>0.116506</td>
<td>0.166588</td>
</tr>
<tr>
<td>( V_4 )</td>
<td>0.166588</td>
<td>0.136203</td>
</tr>
<tr>
<td>( V_5 )</td>
<td>0.359441</td>
<td>0.221262</td>
</tr>
</tbody>
</table>
Table 3.4 Network Distance between 2 User Clusters and 3 Servers

<table>
<thead>
<tr>
<th></th>
<th>$U_1$</th>
<th>$U_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>73</td>
<td>97</td>
</tr>
<tr>
<td>$S_2$</td>
<td>83</td>
<td>38</td>
</tr>
<tr>
<td>$S_3$</td>
<td>63</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3.5 9 Groups of Bandwidth Capacities of Servers

<table>
<thead>
<tr>
<th>Group No.</th>
<th>$B_{S_1}/B_{S_2}/B_{S_3}$ (10^5 kbps)</th>
<th>Group No.</th>
<th>$B_{S_1}/B_{S_2}/B_{S_3}$ (10^5 kbps)</th>
<th>Group No.</th>
<th>$B_{S_1}/B_{S_2}/B_{S_3}$ (10^5 kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.4/0.8/0.8</td>
<td>4</td>
<td>2.4/1.6/0.8</td>
<td>7</td>
<td>2.4/2.4/0.8</td>
</tr>
<tr>
<td>2</td>
<td>2.4/0.8/1.6</td>
<td>5</td>
<td>2.4/1.6/1.6</td>
<td>8</td>
<td>2.4/2.4/1.6</td>
</tr>
<tr>
<td>3</td>
<td>2.4/0.8/2.4</td>
<td>6</td>
<td>2.4/1.6/2.4</td>
<td>9</td>
<td>2.4/2.4/2.4</td>
</tr>
</tbody>
</table>

Table 3.6 Server Selection Solution ($Q_{U_j,V_j,S_i}$) for Group No.1

<table>
<thead>
<tr>
<th></th>
<th>$V_1$</th>
<th>$V_2$</th>
<th>$V_3$</th>
<th>$V_4$</th>
<th>$V_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>1/0/0</td>
<td>1/0/0</td>
<td>1/0/0</td>
<td>1/0/0</td>
<td>1/0/0</td>
</tr>
<tr>
<td>$U_2$</td>
<td>0/0/1</td>
<td>0/0/1</td>
<td>0.47/0.53/0</td>
<td>0/0/1</td>
<td>0/0.51/0.49</td>
</tr>
</tbody>
</table>

Based on the data in Tables 3.2, 3.3, 3.4 and 3.5, we can compute the LP formulation and obtain optimal $Q_{U_j,V_j,S_i}$. Table 3.6 shows an optimal solution for group No.1 of bandwidth capacities of servers. In Table 3.6, we can determine the optimal proportion of user demands to be directed to each server. For instance, “1/0/0” in the second row and the second column of Table 3.6 implies that if users in user cluster $U_1$ ask for video $V_1$, all the demands should be directed to server $S_1$. “0.46/0.54/0” in the third row and the fourth column of Table 3.6 implies that 46%
of user demands from cluster $U_2$ asking for video $V_3$ should be served by server $S_1$ and the remaining demands (54%) should be served by server $S_2$. The CPU of our numerical examples environment is CPU E5-2670 and the memory is 64G. We solve our problem in Matlab 2015a. The execution time of Group No.1 to No.9 is 235s, 238s, 242s, 239s, 250s, 262s, 247s, 252s and 258s.

Fig. 3.3 shows the percentage of total workload of each server for serving all users. According to Table 3.4, server $S_3$ has the minimum latency in serving users in both cluster $U_1$ and $U_2$. Therefore, when bandwidth capacity of server $S_3$ incrementally increases from $0.8 \times 10^5$kbps to $2.4 \times 10^5$ kbps in groups No.1, No.2 and No.3 of Table 3.5, more and more user demands are assigned to server $S_3$, as shown in Fig. 3.3. For example, we observe that percentage of total workload of server $S_3$ is increasing from 33.67% in group No.1 to 75.23% in group No.2, and then to 94.39% in group No.3.

Fig. 3.4 shows the overall latency (i.e., equation (4)) versus the bandwidth capacities of servers $S_2$ and $S_3$ (bandwidth capacity of server $S_1$ is fixed). Since server $S_3$ has the minimum latency in serving users in both cluster $U_1$ and $U_2$, the overall latency with $B_{S_3} = 2.4 \times 10^5$kbps is the minimum while the overall latency with $B_{S_3} = 0.8 \times 10^5$kbps is the maximum. For example, when $B_{S_2} = 0.8 \times 10^5$kbps, we observe that the overall latency decreases from 3426.4ms to 2259.1ms, and then to 1858.6ms with the increase of bandwidth capacity of server $S_3$. 
Fig. 3.3 Percentages of Total Workload of Servers

Fig. 3.4 Overall Latency VS Bandwidth Capacities of Servers
3.5 Simulation

In this section, simulations are conducted for performance evaluation. We apply GPU computing supported by Matlab 2015 to implement the proposed method, so that it can make use of the many graphics processors in the graphics display card GTX 780 for parallel computing for faster execution. We assume that the number of users in every user cluster is uniformly distributed in interval [1,100]. For simplicity, the network distance between user cluster \( U_i \) and server \( S_i \) is set to be uniformly distributed in interval [1ms, 100ms]. We set number of videos \( m = 10^6 \) and the number of user clusters \( k = 5 \). The expected popularity of videos \( P_{ij}^{lt} \), \( 1 \leq l \leq k \) and \( 1 \leq j \leq m \), are generated according to equations (1)-(3). We vary skew factor \( \alpha \) from 0 to 1.9. Table 3.7 shows three typical types of online videos in YouTube [26]. For the common video, each set usually lasts 45 minutes and has a size of 240MB. Thus, bandwidth requirement of the common video is calculated as follows.

\[
\frac{240 \times 1024 \times 8}{45 \times 60} = 728 \text{kbps}.
\]

We can calculate bandwidth requirements of other two types of videos in the same way, as shown in Table 3.7. We assume that all videos in our system are evenly distributed in these three types and bandwidth requirements of videos follow the norm distributions \( N(728, 7.282) \), \( N(1136, 11.362) \) and \( N(5464, 54.642) \), respectively.

<table>
<thead>
<tr>
<th>Video Type</th>
<th>Episode Duration</th>
<th>Size</th>
<th>Bandwidth Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common</td>
<td>45min/set</td>
<td>240MB</td>
<td>728kbps</td>
</tr>
<tr>
<td>Common High Definition</td>
<td>1hr/set</td>
<td>500MB</td>
<td>1136kbps</td>
</tr>
<tr>
<td>Super High Definition MTV</td>
<td>5min/set</td>
<td>200MB</td>
<td>5464kbps</td>
</tr>
</tbody>
</table>
We compare our server selection scheme with a random algorithm and the server selection strategy adopted in YouTube. In the random algorithm, user demands are randomly assigned to a server as long as the available bandwidth capacity of the server is not zero. YouTube adopts a greedy algorithm to select the server which has the minimum latency in serving the given user demand. If the nearest server is too busy (i.e., its available bandwidth is not sufficient), it redirects the user demand to the second nearest server and so on [7].

Fig. 3.5 shows the mean of overall latency versus the skew factor $\alpha$. We set the number of servers $n = 10$. To ensure that there is a feasible solution, we set $B_{s_1}$ according to equation (8) in a similar way. The bandwidth capacities of other servers are uniformly set to be $0.1 \times B_{s_1}$. We vary skew factor $\alpha$ from 0 to 1.9 with the step size of 0.1. The results reported in Fig. 3.5 are the means of 100
separate runs. In Fig. 3.5, we can see that our scheme can significantly reduce the overall latency compared with the random algorithm and the YouTube server selection strategy. For example, when $\alpha = 0.7$, our scheme reduces 79.06% and 24.51% of the overall latency compared with the random algorithm and the YouTube server selection strategy, respectively. The execution time is 845s.

![Fig. 3.6 Mean of Overall Latency VS Number of Servers](image)

Fig. 3.6 shows the mean of overall latency versus the number of servers. We fix the skew factor $\alpha = 0.7$, and increase the number of servers from 10 to 100 with the step size of 10. We set the bandwidth capacities of servers $B_{s_1} = 3.2 \times 10^5 \text{ kbps}$, $B_{s_2} = B_{s_3} = \cdots = B_{s_n} = 0.32 \times 10^5 \text{ kbps}$. We can see that the performance of our scheme is significantly better than those of the random algorithm and the YouTube server selection strategy. For example, when $k = 10$, our scheme reduces 88.28% and 67.57% of the overall latency compared with the random algorithm and the YouTube server selection strategy. The execution time
is 910 seconds, which is fast enough for real-world deployment because the offline optimization problem is solved once per day or several-days.

3.6 Conclusion

In this work, we studied a new server selection problem in cloud computing environments which takes into account diversities of cloud services and handles user demands on a cluster basis. We focus on a specific cloud service, i.e., online video service. Given user demands and bandwidth capacities of servers in the data centers, our problem is to assign the user demands to the servers under the bandwidth constraint, such that the overall latency (measured by the network distance) between the user clusters and the selected servers is minimized. We designed a server selection system in which the server selection problem is formulated as a linear programming formulation. Our system receives user demands and computes an optimal distribution of the user demands among the servers so as to achieve the minimum overall latency. The simulation results have shown that our system can significantly reduce the overall latency, compared with the random algorithm and the YouTube server selection strategy. The proposed solution in this work is applicable to general cases of cloud services.
Chapter 4

Two goals Server Selection for Heterogeneous Cloud Services

4.1 Introduction

In this chapter, we give the solution of two goals server selection for heterogeneous cloud services. Same to the chapter 3, we propose a new and effective server selection scheme in which diversities of cloud services are taken into account. We also focus on a specific cloud service, i.e., online video service, and assume that different videos have different bandwidth requirements and different storage size. In this chapter, user demands, bandwidth and storage capacities of servers in the data centers are given. Our problem is to assign the user demands to the servers under the bandwidth and storage constraint, the function of overall latency (measured by the network distance) between the user clusters and the selected servers and standard deviation of traffic load of every server in the system is minimized. Then, we give the numerical examples and simulations of this new problem.

4.2 System Design and Model

In this chapter, our system architecture is the same as the previous problem which is illustrated in Figure 3.1. There are also three entities involved: video service provider (VSP), Internet service provider (ISP) and end users. The VSP holds multiple data centers (referred to as servers for simplicity) located in
different geographical locations, and provides various videos to end users via our system as a middle layer. Based on ISP information (e.g., IP addresses [12] [13]), the system classifies the end users into several clusters and handles user demands accordingly for each cluster. Our system estimates the latency, in terms of the network distance [3] [14], between the servers and the clusters. The VSP provides real-time video information (e.g., bandwidth requirements, storage size and popularities of videos) and workload information (e.g., available bandwidth of servers and available storage capacity of servers) which are needed in the server selection module of our system. Users may request any videos that are provided by the VSP. For each cluster’s demand asking for the same video, the system performs server selection and computes an optimal distribution among the servers based on the workload information, the video information and the latency information.

In this chapter, we not only take the bandwidth capacity constraint into account, but also add storage capacity constraint into account. We add the information of the storage size of every video $V_j$ as $C_{V_j}$ into VSP Info and add the information of the storage capacity of every server $S_i$ as $C_{S_i}$ into workload Info. If an end user asking for video $V_j$ is served by a server, the available storage capacity of the server should be no less than $C_{V_j}$ so as to guarantee the server has enough space to keep the video.

We could formulate our two goals server selection problem as following:

**Assumption:** we assume that each server has a fixed and given storage capacity, we could put one video copy in one or more servers.

**Video Information:** In video formulation part, we let $V = \{V_1, V_2, \cdots, V_m\}$ denote the set of $m$ videos provided by the VSP. Different from chapter 3, we
don’t assume that all the videos are replicated in every data center because storage
capacity constraint sometimes has to been considered when servers don’t have
enough storage space [27-30]. For each video $V_j, j=1, 2, ..., m$, we consider three
attributes $B_{V_j}, C_{V_j}$ and $P_{V_j}$. $B_{V_j}$ is the bandwidth requirement of video $V_j$. $C_{V_j}$ is
the storage size of video $V_j$ and $P_{V_j}$ is the expected popularity of video $V_j$ which is
calculated as the expected request frequency of video $V_j$ from the users in a unit
time. If an end user asking for video $V_j$ is served by a server, the available
bandwidth of the server should be no less than $B_{V_j}$ so as to guarantee the smooth
delivery of video $V_j$ to the user [23].

**Server Information:** In server formulation part, we define $S = \{S_1, S_2, \ldots, S_i, \ldots, S_n\}$ as the set of given servers. Sometimes, the storage capacities of
servers are not enough [27-30] and it will influence server selection strategy.
Hence, we consider two attributes of servers: $B_{S_i}$ and $C_{S_i}$. $B_{S_i}$ stands for the
bandwidth capacity of server $S_i$ and $C_{S_i}$ stands for the storage capacity of server $S_i$.

**User Information:** In user formulation part, let $U = \{U_1, U_2, \ldots, U_k\}$ denote
the set of user clusters where $U_l$ denotes the set of end users in cluster $l, l=1, 2, \ldots, k$. Let $P^{<\lambda>}_{V_j}$ be the expected popularity of video $V_j$ in user cluster $U_l$. That is, $P^{<\lambda>}_{V_j}$
denotes the expected request frequency of video $V_j$ from the users in user cluster
$U_l$ in a unit time. We use the same method as it has been explained in chapter 3 to
estimate $P^{<\lambda>}_{V_j}$.

There are two main differences in our new problem in this chapter in contrast
with the server selection problem which has been solved in chapter 3. Firstly, we
add the storage capacities of servers constraint into account because as the
development of video definition technology, some high definition videos or super
high definition MTVs occupy more and more storage space, thus servers’ storage capacities will limit the quality of service [27]. Secondly, if we select video servers entirely aimed to minimize the latency between the selected server and the user cluster, a possible result is that some servers hold many nearer and more popular videos but some servers hold few farther and less popular videos. This unbalanced partition will lead to network traffic load problem, which further cause the descend of the quality of video service and dissatisfaction of users’ experience. By the evidence of the study of X. Cheng et al. in [30] on a large collection of YouTube video data, they demonstrate the existence of this unbalanced traffic load phenomenon would bring great problem in cloud computing, from the perspective of network engineering. Thus, we add the balance of traffic load of every server as the other objective in our new server selection problem.

4.3 Problem Formulation of Server Selection

In our system, the server selection module assigns demands of user clusters to the servers. The goal of our system is to minimize the function of overall latency (measured by the network distance) between the user clusters and the selected servers and standard deviation of traffic load of every server in the system. We consider the server workload, i.e., available bandwidth and storage as the main constraints because that overloaded servers cannot provide expected services to the users and unbalance workload of servers may further cause the risk of system breaking down [3]. In the system architecture shown in Figure 3.1, the optimal server selection is made based on the workload information, the video information for popularity estimation, and the ISP information for latency estimation. Specifically, the server selection module computes an optimal distribution of user demands among the servers, based on the following information.
1) Workload of server $S_i$, denoted by $B_{S_i}$ and $C_{S_i}$, $i=1, 2, \ldots, n$.

2) Expected popularity of video $V_j$ in cluster $U_l$, denoted by $P_{V_j}^{<l>}$, $j=1, 2, \ldots, m$, $l=1, 2, \ldots, k$.

3) Network distance between user cluster $U_l$ and server $S_i$, denoted by $D_{U_lS_i}$, $l=1, 2, \ldots, k$, $i=1, 2, \ldots, n$.

First, we give the notations used in our model in Table 4.1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Physical Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>User clusters</td>
<td>$U = {U_1, U_2 \ldots U_l \ldots U_k}$ Set of given user clusters</td>
</tr>
<tr>
<td></td>
<td>$k$ Number of user clusters</td>
</tr>
<tr>
<td></td>
<td>$</td>
</tr>
<tr>
<td>Servers</td>
<td>$S = {S_1, S_2 \ldots S_l \ldots S_n}$ Set of servers</td>
</tr>
<tr>
<td></td>
<td>$n$ Number of servers</td>
</tr>
<tr>
<td></td>
<td>$B_{S_i}$ Available bandwidth capacity of $S_i$</td>
</tr>
<tr>
<td></td>
<td>$C_{S_i}$ Available storage capacity of $S_i$</td>
</tr>
<tr>
<td></td>
<td>$L_{S_i}$ Traffic load of $S_i$</td>
</tr>
<tr>
<td>Videos</td>
<td>$V = {V_1, V_2 \ldots V_l \ldots V_m}$ Set of videos in the system</td>
</tr>
<tr>
<td></td>
<td>$m$ Number of videos</td>
</tr>
<tr>
<td></td>
<td>$B_{V_j}$ Bandwidth requirement of $V_j$</td>
</tr>
<tr>
<td></td>
<td>$C_{V_j}$ Storage size of $V_j$</td>
</tr>
<tr>
<td></td>
<td>$A_{V_j}^{&lt;l&gt;}$ Access probability of video $V_j$ in user cluster $U_l$</td>
</tr>
<tr>
<td></td>
<td>$P_{V_j}^{&lt;l&gt;}$ Expected popularity of video $V_j$ in user cluster $U_l$</td>
</tr>
<tr>
<td>$D_{U_lS_i}$</td>
<td>Network distance between user cluster $U_l$ and server $S_i$</td>
</tr>
<tr>
<td>$T_{V_lS_i}$</td>
<td>Decision function whether put video $V_j$ in server $S_i$ or not</td>
</tr>
<tr>
<td>$Q_{U_lV_jS_i}$</td>
<td>Proportion of user demands from cluster $U_l$ asking for video $V_j$ that is assigned to server $S_i$</td>
</tr>
<tr>
<td>$F_1$</td>
<td>Overall latency (measured by the network distance) between the user clusters and the selected servers</td>
</tr>
<tr>
<td>$F_2$</td>
<td>Standard deviation of traffic load of every server in the system</td>
</tr>
<tr>
<td>$w_1$</td>
<td>Weight of $F_1$</td>
</tr>
<tr>
<td>$w_2$</td>
<td>Weight of $F_2$</td>
</tr>
</tbody>
</table>
Let $Q_{U_i V_j S_l}$ be the proportion of user demands from cluster $U_i$ asking for video $V_j$ that is assigned to server $S_l$. $Q_{U_i V_j S_l}$ is a ratio in range $[0, 1]$. For example, if $Q_{U_i V_j S_l} = 0.3$, it implies that 30% of the demands from cluster $U_i$ asking for video $V_j$ are served by server $S_l$. If $Q_{U_i V_j S_l} = 1$, it means that server $S_l$ serves all the demands from cluster $U_i$ asking for video $V_j$. As our new problem assumes that the storage capacities of servers are not always enough, we let $T_{V_j S_l}$ be the decision function whether put video $V_j$ in server $S_l$ or not. If $T_{V_j S_l} = 0$, it means the video $V_j$ isn’t assigned to the server $S_l$. If $T_{V_j S_l} = 1$, it means the video $V_j$ has a copy in the server $S_l$.

The problem of server selection can be formulated as the following.

Minimize \[ F = w_1 \text{Normalized}(F_1) + w_2 \text{Normalized}(F_2) \] (9)

\[ F_1 = \sum_{i=1}^{k} \sum_{j=1}^{m} \sum_{l=1}^{n} T_{V_j S_l} \cdot Q_{U_i V_j S_l} \cdot D_{U_l S_l} \cdot P_{V_j}^{<i>} \] (10)

\[ F_2 = \sqrt{\frac{\sum_{i=1}^{n} (L_{S_i} - \bar{L})^2}{n}} \] (11)

\[ L_{S_i} = \frac{\sum_{j=1}^{m} B_{V_j} \left( \sum_{l=1}^{k} T_{V_j S_l} \cdot Q_{U_i V_j S_l} \cdot P_{V_j}^{<i>} \right)}{B_{S_l}} \] (12)

Subject to

\[ \sum_{j=1}^{m} B_{V_j} \cdot \left( \sum_{l=1}^{k} T_{V_j S_l} \cdot Q_{U_i V_j S_l} \cdot P_{V_j}^{<i>} \right) \leq B_{S_l}, \quad i=1, 2, \ldots, n; \] (13)

\[ \sum_{j=1}^{m} C_{V_j} \cdot T_{V_j S_l} \leq C_{S_l}, \quad i=1, 2, \ldots, n; \] (14)

\[ Q_{U_i V_j S_l} \leq T_{V_j S_l}; \] (15)

\[ \sum_{l=1}^{n} Q_{U_i V_j S_l} = 1, \quad l=1, 2, \ldots, k, \quad j=1, 2, \ldots, m. \] (16)

\[ Q_{U_i V_j S_l} \leq 1, \quad \forall \ i, j, l. \] (17)

The above formulation is a nonlinear programming formulation which can be solved by existing techniques such as the simplex algorithm in [25]. The item in (9)
is the function of overall latency (measured by the network distance) between the user clusters and the selected servers and standard deviation of traffic load of every server in the system. We could adopt weight factor $w_1$ and $w_2$ to adjust the influences on the final result by overall latency and traffic load, respectively. Equation (10) implies the overall latency (measured by the network distance) between the user clusters and the selected servers. Equation (11) implies the standard deviation of traffic load of every server in the system. Equation (12) implies the traffic load of every server. Inequation (13) implies that the total bandwidth that each server consumes cannot exceed its available bandwidth capacity. Inequation (14) implies that the total storage that each server consumes cannot exceed its available storage capacity. Equation (16) implies that all demands from cluster $U_i$ asking for video $V_j$ should be eventually satisfied by the servers.

Once all variables $Q_{U_iV_jS_i}$ are determined, server selection module assigns user demands according to the optimal solution. Notice that the workload information, the latency information and the expected popularities of videos should be periodically updated and the LP formulation is re-computed by the server selection module.

### 4.4 Numerical Examples

In this section, we present numerical examples for illustration of our server selection scheme. In our example, the number of user clusters is set to $k = 2$, and the numbers of users in the two clusters are set to $|U_1| = 20$ and $|U_2| = 100$, respectively. We suppose there are five videos in the system (i.e., $m = 5$) and bandwidth/storage requirements of the five videos are shown in Table 4.2. For simplicity, we assume that the skew factor of equation (2) $\alpha$ is estimated to be 0.7
based on the historic data. According to the model in equation (2), access probabilities of the five videos are calculated and shown in Table 4.3. Table 4.4 shows the storage capacity of every 5 server. Table 4.5 shows the network distances between 2 user clusters and 3 servers which are random numbers in [0, 100]. To ensure that there is a feasible solution, we set

\[ B_{S_1} = \sum_{j=1}^{5} B_{V_j} \left( \sum_{i=1}^{2} P_{V_j}^{<i>} \right) = 2.4 \times 10^4 \text{ kbps.} \quad (18) \]

That is, bandwidth capacity of server \( S_1 \) is sufficient to serve all user demands. We consider nine groups of bandwidth capacities of servers, as shown in Table 4.6. For example, in group No.1, \( B_{S_1} = 2.4 \times 10^4 \text{ kbps} \) and \( B_{S_2} = B_{S_3} = 0.8 \times 10^4 \text{ kbps} \).

In this numerical example, we set the weight factor \( w_1 = w_2 = 0.5 \). It means that two goals (the overall latency and the balance of traffic load) have the same influence on the final optimization result.

<table>
<thead>
<tr>
<th>( V_j )</th>
<th>( V_1 )</th>
<th>( V_2 )</th>
<th>( V_3 )</th>
<th>( V_4 )</th>
<th>( V_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B_{V_j} ) (kbps)</td>
<td>728</td>
<td>1138</td>
<td>5362</td>
<td>736</td>
<td>3021</td>
</tr>
<tr>
<td>( C_{V_j} ) (GB)</td>
<td>0.8</td>
<td>1.2</td>
<td>3.0</td>
<td>1.0</td>
<td>2.4</td>
</tr>
</tbody>
</table>

Table 4.3 Access Probabilities of 5 Videos in 2 User Clusters

<table>
<thead>
<tr>
<th>( V_j )</th>
<th>( U_1 )</th>
<th>( U_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_1 )</td>
<td>0.136203</td>
<td>0.359441</td>
</tr>
<tr>
<td>( V_2 )</td>
<td>0.221262</td>
<td>0.116506</td>
</tr>
<tr>
<td>( V_3 )</td>
<td>0.116506</td>
<td>0.166588</td>
</tr>
<tr>
<td>( V_4 )</td>
<td>0.166588</td>
<td>0.136203</td>
</tr>
</tbody>
</table>
Based on the data in Tables 4.2, 4.3, 4.4, 4.5 and 4.6, we can compute the formulation and obtain optimal $Q_{u_i,v_j,s_i}$. Table 4.7 shows an optimal solution for group No.1 of bandwidth capacities of servers.

In Table 4.7, we can determine the optimal proportion of user demands to be directed to each server. For instance, “0/0/1” in the third row and the forth column of Table 4.7 implies that if users in user cluster $U_2$ ask for video $V_3$, all the

<table>
<thead>
<tr>
<th>$V_5$</th>
<th>0.359441</th>
<th>0.221262</th>
</tr>
</thead>
</table>

Table 4.4 Storage Capacity of Every 5 Server

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{S_i}(\text{GB})$</td>
<td>10</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4.5 Network Distance between 2 User Clusters and 3 Servers

<table>
<thead>
<tr>
<th>$U_1$/$S_i$ (ms)</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>73</td>
<td>83</td>
<td>63</td>
</tr>
<tr>
<td>$U_2$</td>
<td>97</td>
<td>38</td>
<td>70</td>
</tr>
</tbody>
</table>

Table 4.6 9 Groups of Bandwidth Capacities of Servers

<table>
<thead>
<tr>
<th>Group No.</th>
<th>$B_{S_1}/B_{S_2}/B_{S_3}$ (10^4kbps)</th>
<th>Group No.</th>
<th>$B_{S_1}/B_{S_2}/B_{S_3}$ (10^4kbps)</th>
<th>Group No.</th>
<th>$B_{S_1}/B_{S_2}/B_{S_3}$ (10^4kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.4/0.8/0.8</td>
<td>4</td>
<td>2.4/1.6/0.8</td>
<td>7</td>
<td>2.4/2.4/0.8</td>
</tr>
<tr>
<td>2</td>
<td>2.4/0.8/1.6</td>
<td>5</td>
<td>2.4/1.6/1.6</td>
<td>8</td>
<td>2.4/2.4/1.6</td>
</tr>
<tr>
<td>3</td>
<td>2.4/0.8/2.4</td>
<td>6</td>
<td>2.4/1.6/2.4</td>
<td>9</td>
<td>2.4/2.4/2.4</td>
</tr>
</tbody>
</table>
demands should be directed to server $S_3$. “0/0.02/0.98” in the second row and the third column of Table 4.7 implies that none of user demands from cluster $U_1$ asking for video $V_2$ should be served by server $S_1$, 2% of user demands from cluster $U_1$ asking for video $V_2$ should be served by server $S_2$ and the remaining demands (98%) should be served by server $S_3$. In contrast with the server selection solution only for the minimized overall latency objective, which is shown in Table3.6, we can find this solution has more balance assignment of the user requests as we add the traffic load goals in our objectives. Another server selection results for Group No.2 to Group No.9 are shown in Table 4.8 to Table 4.15.

The CPU of our numerical examples environment is E5-2670 and the memory is 64G. We solve our problem in Matlab 2015a. The execution time of Group No.1 to No.9 is 1788s, 1812s, 1754s, 1831s, 1788s, 1892s, 1910s, 1954s and 1977s.

Table 4.7 Server Selection Solution ($Q_{U_jV_jS_{ij}}$) for Group No.1

<table>
<thead>
<tr>
<th>$U_i$</th>
<th>$V_j$</th>
<th>$V_1$</th>
<th>$V_2$</th>
<th>$V_3$</th>
<th>$V_4$</th>
<th>$V_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>1/0/0</td>
<td>0/0/0.02/0.98</td>
<td>1/0/0</td>
<td>1/0/0</td>
<td>1/0/0</td>
<td>1/0/0</td>
</tr>
<tr>
<td>$U_2$</td>
<td>0/1/0</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0/0/1</td>
</tr>
</tbody>
</table>

Table 4.8 Server Selection Solution ($Q_{U_jV_jS_{ij}}$) for Group No.2

<table>
<thead>
<tr>
<th>$U_i$</th>
<th>$V_j$</th>
<th>$V_1$</th>
<th>$V_2$</th>
<th>$V_3$</th>
<th>$V_4$</th>
<th>$V_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>0/0/1</td>
<td>0/0/1</td>
<td>1/0/0</td>
<td>1/0/0</td>
<td>0.75/0/0.25</td>
<td>0/0/1</td>
</tr>
<tr>
<td>$U_2$</td>
<td>0/1/0</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0/0/1</td>
</tr>
</tbody>
</table>
Table 4.9 Server Selection Solution ($Q_{U_{i}V_{j}S_{l}}$) for Group No.3

<table>
<thead>
<tr>
<th>$U_{i}$</th>
<th>$V_{1}$</th>
<th>$V_{2}$</th>
<th>$V_{3}$</th>
<th>$V_{4}$</th>
<th>$V_{5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_{1}$</td>
<td>0/0/1</td>
<td>0/0/1</td>
<td>1/0/0</td>
<td>1/0/0</td>
<td>0.3/0/0.7</td>
</tr>
<tr>
<td>$U_{2}$</td>
<td>0/1/0</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0.01/0/0.99</td>
<td>0.01/0.98/0.01</td>
</tr>
</tbody>
</table>

Table 4.10 Server Selection Solution ($Q_{U_{i}V_{j}S_{l}}$) for Group No.4

<table>
<thead>
<tr>
<th>$U_{i}$</th>
<th>$V_{1}$</th>
<th>$V_{2}$</th>
<th>$V_{3}$</th>
<th>$V_{4}$</th>
<th>$V_{5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_{1}$</td>
<td>0.04/0/0.96</td>
<td>0.38/0/0.62</td>
<td>1/0/0</td>
<td>0.99/0.01/0</td>
<td>0.89/0.11/0</td>
</tr>
<tr>
<td>$U_{2}$</td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/1/0</td>
</tr>
</tbody>
</table>

Table 4.11 Server Selection Solution ($Q_{U_{i}V_{j}S_{l}}$) for Group No.5

<table>
<thead>
<tr>
<th>$U_{i}$</th>
<th>$V_{1}$</th>
<th>$V_{2}$</th>
<th>$V_{3}$</th>
<th>$V_{4}$</th>
<th>$V_{5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_{1}$</td>
<td>0/0/1</td>
<td>0/0/1</td>
<td>1/0/0</td>
<td>0.34/0/0.66</td>
<td>1/0/0</td>
</tr>
<tr>
<td>$U_{2}$</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/1/0</td>
</tr>
</tbody>
</table>

Table 4.12 Server Selection Solution ($Q_{U_{i}V_{j}S_{l}}$) for Group No.6

<table>
<thead>
<tr>
<th>$U_{i}$</th>
<th>$V_{1}$</th>
<th>$V_{2}$</th>
<th>$V_{3}$</th>
<th>$V_{4}$</th>
<th>$V_{5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_{1}$</td>
<td>0/0/1</td>
<td>0/0/1</td>
<td>0.86/0/0.14</td>
<td>0/0/1</td>
<td>1/0/0</td>
</tr>
<tr>
<td>$U_{2}$</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/1/0</td>
</tr>
</tbody>
</table>
4.5 Simulation

In this section, simulations are conducted in E5-2670, 64GB, GTX 780. We apply GPU computing supported by Matlab 2015a to implement the proposed method, so that it can make use of the many graphics processors in the graphics display card GTX 780 for parallel computing for faster execution. We assume that the number of users in every user cluster is uniformly distributed in interval [1,100]. For simplicity, the network distance between user cluster $U_i$ and server $S_j$ is set to be uniformly distributed in interval [1ms, 100ms]. We set number of videos $m = 10^6$ and the number of user clusters $k = 5$. The expected popularity

Table 4.13 Server Selection Solution ($Q_{U_iV_j,S_l}$) for Group No.7

<table>
<thead>
<tr>
<th>$U_i$</th>
<th>$V_j$</th>
<th>$V_1$</th>
<th>$V_2$</th>
<th>$V_3$</th>
<th>$V_4$</th>
<th>$V_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td></td>
<td>0.45/0/0.55</td>
<td>0.38/0/0.62</td>
<td>1/0/0</td>
<td>1/0/0</td>
<td>0.01/0.99/0</td>
</tr>
<tr>
<td>$U_2$</td>
<td></td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/1/0</td>
</tr>
</tbody>
</table>

Table 4.14 Server Selection Solution ($Q_{U_iV_j,S_l}$) for Group No.8

<table>
<thead>
<tr>
<th>$U_i$</th>
<th>$V_j$</th>
<th>$V_1$</th>
<th>$V_2$</th>
<th>$V_3$</th>
<th>$V_4$</th>
<th>$V_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td></td>
<td>0/0/1</td>
<td>0/0/1</td>
<td>1/0/0</td>
<td>0.75/0/0.25</td>
<td>0.32/0.68/0</td>
</tr>
<tr>
<td>$U_2$</td>
<td></td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/1/0</td>
</tr>
</tbody>
</table>

Table 4.15 Server Selection Solution ($Q_{U_iV_j,S_l}$) for Group No.9

<table>
<thead>
<tr>
<th>$U_i$</th>
<th>$V_j$</th>
<th>$V_1$</th>
<th>$V_2$</th>
<th>$V_3$</th>
<th>$V_4$</th>
<th>$V_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td></td>
<td>0/0/1</td>
<td>0/0/1</td>
<td>0.93/0/0.07</td>
<td>0/0.01/0.99</td>
<td>0.52/0.48/0</td>
</tr>
<tr>
<td>$U_2$</td>
<td></td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/0/1</td>
<td>0/1/0</td>
<td>0/1/0</td>
</tr>
</tbody>
</table>
of videos $P_{ij}^{<\tau}$, $1 \leq l \leq k$ and $1 \leq j \leq m$, are generated according to equations (1)-(3). We vary skew factor $\alpha$ from 0 to 1.9. Table 4.16 shows three typical types of online videos in YouTube [26].

We assume that all videos in our system are distributed in these three types and storage requirements of videos follow the norm distributions $N(240,24), N(500,50), N(200,20)$, respectively. For the common video, each set usually lasts 45 minutes and has a size of 240MB. Thus, bandwidth requirement of the common video is calculated as follows.

$$\frac{240 \times 1024 \times 8}{45 \times 60} = 728 \text{kbps}.$$  

We can calculate bandwidth requirements of other two types of videos in the same way, as shown in Table 4.8. We assume that all videos in our system are evenly distributed in these three types and bandwidth requirements of videos follow the norm distributions $N(728, 7.282), N(1136, 11.362)$ and $N(5464, 54.642)$, respectively.

<table>
<thead>
<tr>
<th>Video Type</th>
<th>Episode Duration</th>
<th>Size</th>
<th>Bandwidth Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common</td>
<td>45min/set</td>
<td>240MB</td>
<td>728kbps</td>
</tr>
<tr>
<td>Common High Definition</td>
<td>1hr/set</td>
<td>500MB</td>
<td>1136kbps</td>
</tr>
<tr>
<td>Super High Definition MTV</td>
<td>5min/set</td>
<td>200MB</td>
<td>5464kbps</td>
</tr>
</tbody>
</table>

We compare our server selection scheme with a random algorithm and the server selection strategy adopted in YouTube. In the random algorithm, user demands are randomly assigned to a server as long as the available bandwidth and storage capacity of the server is not zero. YouTube selects the server which has the minimum latency in serving the given user demand if the server’s available storage capacity is sufficient. If the nearest server is too busy (i.e., its available
bandwidth is not sufficient), it redirects the user demand to the second nearest server and so on [7].

Fig. 4.1 shows the mean of overall latency, normalized $F_2$ and normalized $F$ versus the skew factor $\alpha$. We set the number of servers $n = 10$. To ensure that there is a feasible solution, we set

$$C_{s_1} = m \cdot \max \left(C_{v_j}\right) = 10^6 \times 550\text{MB} = 5.5 \times 10^5\text{GB}$$  \hspace{1cm} (19)

That is, storage capacity of server $S_1$ is sufficient to serve all user demands. The storage capacities of other servers are uniformly set to be $0.1 \times C_{s_1}$. We set $B_{s_1}$ according to equation (18) in a similar way. The bandwidth capacities of other servers are uniformly set to be $0.1 \times B_{s_1}$. We vary skew factor $\alpha$ from 0 to 1.9 with the step size of 0.1. Finally, we set the weight factor of two goals $w_1 = w_2 = 0.5$. That means the objective of minimizing the overall latency and the objective of balancing the traffic load have the same influence on the final optimization result. The results reported in Fig. 4.1 are the means of 100 separate runs. The execution time is 79659s.

In Fig. 4.1(A), we can see that our scheme can significantly reduce the overall latency compared with the random algorithm and the YouTube server selection strategy. For example, when $\alpha = 0.7$, our scheme reduces 89.77% and 73.58% of the overall latency compared with the random algorithm and the YouTube server selection strategy, respectively.

In Fig. 4.1(B), we can see that our scheme can significantly reduce the normalized $F_2$ compared with the random algorithm and the YouTube server selection strategy. For example, when $\alpha = 0.7$, our scheme reduces 89.00% and 91.81% of the normalized $F_2$ compared with the random algorithm and the
YouTube server selection strategy, respectively. That means our scheme has more balanced traffic load among servers.

In Fig. 4.1(C), we can see that our scheme can significantly reduce the normalized $F$ compared with the random algorithm and the YouTube server selection strategy. For example, when $\alpha = 0.7$, our scheme reduces 89.5% and 70.19% of the normalized $F$ compared with the random algorithm and the YouTube server selection strategy, respectively. That means our scheme could reduce overall latency and it has more balanced traffic load among servers compared with the random algorithm and the YouTube server selection strategy.

(A) Mean of Overall Latency VS Skew Factor $\alpha$
Fig. 4.1 Mean of Overall Latency, Normalized $F_2$ and Normalized $F$ VS Skew Factor $\alpha$
Fig. 4.2 shows the mean of overall latency, normalized $F_2$ and normalized $F$ versus the number of servers. We fix the skew factor $\alpha = 0.7$, and increase the number of servers from 10 to 100 with the step size of 10. To ensure that there is a feasible solution, we set the storage capacities of servers $C_{s_1} = 5.5 \times 10^5 GB$, $C_{s_2} = C_{s_3} = \cdots = C_{s_n} = 5.5 \times 10^4 GB$. In the same way, we set the bandwidth capacities of servers $B_{s_1} = 3.2 \times 10^5 kbps$, $B_{s_2} = B_{s_3} = \cdots = B_{s_n} = 0.32 \times 10^5 kbps$. Finally, we set the weight factor of two goals $w_1 = w_2 = 0.5$. That means the objective of minimizing the overall latency and the objective of balancing the traffic load have the same influence on the final optimization result.

The results reported in Fig. 4.2 are the means of 100 separate runs. The execution time is 80325s.

In Fig. 4.2(A), we can see that our scheme can significantly reduce the overall latency compared with the random algorithm and the YouTube server selection strategy. For example, when $k = 10$, our scheme reduces 80.00% and 70.14% of the overall latency compared with the random algorithm and the YouTube server selection strategy, respectively.

In Fig. 4.2(B), we can see that our scheme can significantly reduce the normalized $F_2$ compared with the random algorithm and the YouTube server selection strategy. For example, when $k = 10$, our scheme reduces 69.9% and 52.95% of the normalized $F_2$ compared with the random algorithm and the YouTube server selection strategy, respectively. That means our scheme has more balanced traffic load among servers.

In Fig. 4.2(C), we can see that our scheme can significantly reduce the normalized $F$ compared with the random algorithm and the YouTube server
selection strategy. For example, when $k = 10$, our scheme reduces 81.45% and 57.59% of the normalized $F$ compared with the random algorithm and the YouTube server selection strategy, respectively. That means our scheme could reduce overall latency and it has more balanced traffic load among servers compared with the random algorithm and the YouTube server selection strategy.

(A) Mean of Overall Latency VS Number of Servers
Fig. 4.2 Mean of Overall Latency, Normalized $F_2$ and Normalized $F$ VS Number of Servers

(B) Normalized $F_2$ VS Number of Servers

(C) Normalized $F$ VS Number of Servers
Fig. 4.3 shows the mean of overall latency, normalized $F_2$ and normalized $F$ versus the weight factor of $F_1$: $w_1$. We fix the skew factor $\alpha = 0.7$. And we set the number of servers $n = 10$. To ensure that there is a feasible solution, we set $C_{s_1} = 5.5 \times 10^5 \text{GB}$. That is, the storage capacity of server $S_1$ is sufficient to serve all user demands. The storage capacities of other servers are uniformly set to be $0.1 \times C_{s_1}$. We set $B_{s_1}$ according to equation (18) in a similar way. The bandwidth capacities of other servers are uniformly set to be $0.1 \times B_{s_1}$. We vary $w_1$ from 0 to 1 with the step size of 0.1. When $w_1 = 0$, it means the only one objective of balancing the traffic load has the influence on the final optimization result. When $w_1 = 1$, it means the only one objective of minimizing the overall latency has the influence on the final optimization result. When $w_1 = 0.5$, it means the objective of minimizing the overall latency and the objective of balancing the traffic load have the same influence on the final optimization result. The results reported in Fig. 4.3 are the means of 100 separate runs. The execution time is 79873s.

In Fig. 4.3(A), we can see that our scheme can significantly reduce the overall latency compared with the random algorithm and the YouTube server selection strategy. For example, when $w_1 = 0.1$, our scheme reduces 96.45% and 49.86% of the overall latency compared with the random algorithm and the YouTube server selection strategy, respectively.

In Fig. 4.3(B), we can see that our scheme can significantly reduce the normalized $F_2$ compared with the random algorithm and the YouTube server selection strategy. For example, when $w_1 = 0.1$, our scheme reduces 97.31% and 87.89% of the normalized $F_2$ compared with the random algorithm and the YouTube server selection strategy, respectively. That means our scheme has more balanced traffic load among servers.
In Fig. 4.3(C), we can see that our scheme can significantly reduce the normalized $F$ compared with the random algorithm and the YouTube server selection strategy. For example, when $w_1 = 0.1$, our scheme reduces 98.72% and 96.38% of the normalized $F$ compared with the random algorithm and the YouTube server selection strategy, respectively. That means our scheme could reduce overall latency and it has more balanced traffic load among servers compared with the random algorithm and the YouTube server selection strategy.

(A) Mean of Overall Latency VS $w_1$
Fig. 4.3 Mean of Overall Latency, Normalized $F_2$ and Normalized $F$ VS $w_1$
Table 4.17 shows the execution time of the proposed algorithm in Fig.4.1, Fig.4.2 and Fig.4.3. We can see that the proposed algorithm takes about 22 hours, which should be acceptable in real-world deployment because the offline optimization problem is solved once per day or several-days.

Table 4.17: The execution time from Fig.4.1 to Fig.4.3

<table>
<thead>
<tr>
<th>No.</th>
<th>n (number of servers)</th>
<th>$B_{S_1}$ (kbps)</th>
<th>$C_{S_1}$ (GB)</th>
<th>Execution time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig.4.1</td>
<td>10</td>
<td>$B_{S_1} = Max(B)$ Others=0.1$B_{S_1}$</td>
<td>$C_{S_1} = 5.5 \times 10^6$, others=0.1$C_{S_1}$</td>
<td>79659</td>
</tr>
<tr>
<td>Fig 4.2</td>
<td>10-100</td>
<td>$B_{S_1} = Max(B)$ Others=0.1$B_{S_1}$</td>
<td>$C_{S_1} = 5.5 \times 10^6$, others=0.1$C_{S_1}$</td>
<td>80325</td>
</tr>
<tr>
<td>Fig 4.3</td>
<td>10</td>
<td>$B_{S_1} = Max(B)$ Others=0.1$B_{S_1}$</td>
<td>$C_{S_1} = 5.5 \times 10^6$, others=0.1$C_{S_1}$</td>
<td>79873</td>
</tr>
</tbody>
</table>

4.6 Conclusion

In this work, we studied a new server selection problem in cloud computing environments which takes into account diversities of cloud services and handles user demands on a cluster basis. We focus on a specific cloud service, i.e., online video service. In this chapter, user demands, bandwidth and storage capacities of servers in the data centers are given. Our problem is to assign the user demands to the servers under the bandwidth and storage constraint, the function of overall latency (measured by the network distance) between the user clusters and the selected servers and standard deviation of traffic load of every server in the system is minimized. We designed a server selection system in which the server selection problem is formulated as a nonlinear programming formulation. Our system receives user demands and computes an optimal distribution of the user demands among the servers so as to achieve the minimum overall latency. The simulation
results have shown that our system can significantly reduce the overall latency, compared with the random algorithm and the YouTube server selection strategy. The proposed solution in this work is applicable to general cases of cloud services.
Chapter 5

Conclusions and Future Work

In this work, we investigate a new server selection problem in cloud computing environments which differs from existing problems in the following aspects.

1) We consider diversities of cloud services.

As we mentioned in the previous part, there is a wide variety of cloud services and these services have distinct features such as latency requirement and bandwidth requirement. In this thesis, we take online video service which has become one of the most popular applications in the Internet and it holds approximately 86% of global network traffics [8] as an example to consider the influence of diversities of online videos’ bandwidth requirement on the server selection results. The proposed solution in this work is applicable to general cases of cloud services.

2) We consider correlation of users and group users into clusters.

We group users into clusters and handle user demands on a cluster basis. We design a server selection system which interconnects video service provider (VSP), Internet service provider (ISP) and end users. The system receives demands from clustered users and assigns the demands to the servers, such that the overall latency (measured by the network distance) between the user clusters and the selected servers and standard deviation of traffic load of every server in the system is minimized.
Firstly, we give details of our server selection system design and models. And we give the solution of minimum latency server selection for heterogeneous cloud services. In this work, user demands and bandwidth capacities of servers in the data centers are given. Our problem is to assign the user demands to the servers under the bandwidth constraint, such that the overall latency (measured by the network distance) between the user clusters and the selected server is minimized. We designed a server selection system in which the server selection problem is formulated as a linear programming formulation. Our system receives user demands and computes an optimal distribution of the user demands among the servers so as to achieve the minimum overall latency. The simulation results have shown that our system can significantly reduce the overall latency, compared with the random algorithm and the YouTube server selection strategy. The proposed solution in this work is applicable to general cases of cloud services.

Secondly we give the solution of two goals server selection for heterogeneous cloud services. In this work, user demands, bandwidth and storage capacities of servers in the data centers are given. Our problem is to assign the user demands to the servers under the bandwidth and storage constraint, the function of overall latency (measured by the network distance) between the user clusters and the selected servers and standard deviation of traffic load of every server in the system is minimized. We designed a server selection system in which the server selection problem is formulated as a nonlinear programming formulation. Our system receives user demands and computes an optimal distribution of the user demands among the servers so as to achieve the minimum overall latency. The simulation results have shown that our system can significantly reduce the overall latency,
compared with the random algorithm and the YouTube server selection strategy.
The proposed solution in this work is applicable to general cases of cloud services.

In our current work, we consider the latency, the bandwidth capacity and the
storage capacity in our server selection problem. In future research, it would be
worthy to investigate the energy factor because of the huge energy consumption in
data centers with many server machines. If the energy consumption can be
reduced or better controlled, this would save the operation cost of the provider and
be more environment-friendly. When the energy factor is taken into account, there
are two crucial factors: latency (or network distance) and energy consumption.
There are three possible ways to handle these two crucial factors:

1. The objective is to minimize the energy consumption while a constraint
   ensures that the latency is not larger than a given and acceptable value.
2. The objective is to minimize the latency while a constraint ensures that the
   energy consumption is not larger than a given and acceptable value.
3. The objectives are to: i) minimize the latency, and ii) minimize the energy
   consumption. In this case, the problem is a bi-objective optimization problem.
   Since it may not be possible to optimize both objectives simultaneously, we
could apply the existing multi-objective optimization methodologies by which
the network designers could make tradeoff between these two objectives (e.g.,
either weighted sum of objective functions or Pareto frontier).
Bibliography


CURRICULUM VITAE

Academic qualifications of the thesis author, Miss. CHANG He:

• Received the degree of Bachelor of Engineering from East China Normal University, June 2012.

August 2017