THE HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF COMPUTING

Towards Migrating User Applications to the Cloud

Kunfeng LAI

A Thesis Submitted in Partial Fulfillment of

the Requirements for the Degree of

Doctor of Philosophy

June 2012

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____(Signature)

Kunfeng LAI (Name of Student)

ABSTRACT

The cloud computing has become an increasingly popular paradigm for Internet service hosting. It increases the operational efficiency by freeing the users from the trivial matters, such as software and hardware configuration, electricity power management, to name but a few. To run their applications on the cloud, enterprises should migrate their data, computing, and operations to the cloud side.

Though this migration trend is clear, there are many difficulties. The applications migrated to the clouds usually face multiple online users with widely diverse requirements; and the number of users can sharply increase or decrease. Therefore, it is challenging for an enterprise to migrate its applications to the clouds while providing stable services to the users with minimum infrastructure costs and minimum resource consumption.

In this thesis, we conduct a systematic study on this problem. First, we conduct studies on user traffic pattern as we believe the migration is tightly coupled with user behavior. Second, we design a stable workload management scheme that maintains service quality and adapts to user traffic dynamics. Third, we study how to deploy applications to the cloud servers with minimized deployment cost.

Our thesis completes the cycle with user behavior understanding, user service quality maintenance and cost minimization. Our results show that the user traffic of the Internet applications can be affected by various factors. For example, the external links may increase the video popularity in the video sharing site like Youku by 15%. Our workload management scheme can stably optimize the resource allocation under the dynamics of user traffic, and our deployment scheme can save the costs by 30% as compared to the traditional deployment scheme.

In detail, we firstly study how one special factor of the video sharing sites, external links, affects the video sharing sites popularity. These external links is provided by video sharing sites, for videos to be embedded into external web pages. Clearly, the purpose of such function is to increase the distribution of the videos. Does this function fulfill its purpose? In this part of our thesis, we provide a comprehensive measurement study and analysis on these external links to answer this question. With the traces collected from two major video sharing sites, YouTube and Youku of China, we show that the external links have impacts on the popularity of the video sharing sites. More specifically, for videos that have been uploaded for eight months in Youku, around 15% of views can come from external links. We also show that if a video is popular itself, it is likely to have a large number of external links.

In the second scenario, we design a stable workload management scheme to adapt for user traffic dynamics. To assure the service quality, the Service Level Agreement (SLA) is widely adopted. However, as the resource of the service provider is limited and the user requests are dynamic and online, the SLA can be violated. This will incur a loss, in money or in the number of users. As such, an important objective of the service provider is to minimize such loss. A widely adopted scheme for SLA maintenance is based on the prediction of the request arrival rate. In the study of this part, we will show that such scheme cannot always achieve a good minimization as they neglect the close loop between the predicted request arrival rate and response time. We propose an improved workload management scheme based on Iterative Extended Kalman Filter that can optimize and stabilize the system under SLA constraints. Based on analysis and real experiments, we show that our scheme can minimize the loss from SLA violation and achieve stability under fluctuating request arrival rates.

Finally, we further consider the deployment schemes with the objective of minimization the deployment cost. To deploy the Internet applications, nowadays, end users purchase from one service provider. As there are an increasing number of platforms that can support

applications to run cross different providers, we propose a multi-provider scheme to select different service instances from multiple providers to save costs in this part of our thesis. We formulate this multi-provider selection problem. We show that the problem is NP-complete. We develop an optimal algorithm for a special case and a non-trivial $(1 + \epsilon)$ approximation algorithm for the general problem. We further study two variants and develop efficient algorithms. We systematically evaluate our algorithms under different configurations and we conduct a case study for a real online MMORPG web-game. We show that by sharing services on Amazon, GoGrid and Rackspace, we can reduce the bill by 30%.

PUBLICATIONS

Journal Papers

1. **K. Lai**, and Dan Wang, "Understanding the External Links of Video Sharing Sites: Measurement and Analysis", Accepted in *IEEE Transactions on Multimedia*, 2012.

Conference Papers

1. **K. Lai**, D. Wang, and L. Zhang, "Minimizing the Bills of Cloud Services by Multi-Provider Selection", submitted to *IEEE ICNP'12*, Texas, the USA, Oct. 30 - Nov. 2, 2012.

2. **K. Lai**, W. Wang, H. Wu, and D. Wang, "Kalman Filter-based Scheduling for Stable Differentiated Service in the Clouds", submitted to *IEEE ICNP'12*, Texas, the USA, Oct. 30 - Nov. 2, 2012.

3. S. Ding, **K. Lai** and D. Wang, "A Study on the Characteristics of the Data Traffic of Online Social Networks", in *Proc. IEEE ICC'11*, Kyoto, Japan, Jun. 5 - 9, 2011.

4. **K. Lai** and D. Wang, "Towards Understanding the External Links on Video Sharing Sites: Measurement and Analysis", in Proc. ACM NOSSDAV'10, Amsterdam, Netherland, Jun. 2 - 4, 2010.

5. **K. Lai** and D. Wang, "A Measurement Study of the External Links of YouTube", in *Proc. IEEE GlobeCom'09*, Honolulu, Hawaii, Nov. 30 - Dec. 4 , 2009.

Book Chapters

1. X. Cheng, **K. Lai**, D. Wang, and J. Liu, “UGC Video Sharing: Measurement and Analysis”, *Intelligent Multimedia Transmission: Techniques and Applications*, Springer-Verlag, edited by Changwen Chen, Zhu Li, and Shiguo Lian, 2010.

ACKNOWLEDGEMENTS

First and foremost, I should give my special gratitude to my chief supervisor, Prof. Dan Wang, for his understanding, rigorous supervision, and patience during my research. To be more important, Prof. Wang not only shares me with his vast knowledge in our research areas, but also instructs me with his methodologies to learn, to work, and to express, which can be of benefits in a life long time. It is a great happiness to be a student of Prof. Wang, and I should sincerely thank his helps and encourages in my work. Without these helps, it is impossible for me to make such progress these years and complete this work.

I also would like to thank my co-supervisor, Prof. Jiannong Cao. His rigidity and advices inspire me these years. I should also acknowledge my special appreciation to Prof. Jiangchuan Liu at the Simon Fraser University for his wise advices and great helps during my PhD. studies. I also need to thank Prof. Rocky Chang, Dr. Zhu Li, Prof. Bin Xiao, and Prof. Qixin Wang at the Hong Kong Polytechnic University for giving me valuable advices for my researches. Furthermore, I also need to express my gratitude for Bo Li, Meijia Hou, Dr. Yang Liu, Dr. Dongmin Guo, Dr. Xiaocui Sun, Dr. Yi Wang, Weichao Li, Yi Yuan, and Qing Li, for the assistances for my progresses and giving me so many pleasures.

I should acknowledge Dr. Wenjie Wang at the PPLive Inc. and Dr. Haishan Wu at the IBM Research Center at China for the opportunity to visit IBM as well as their patient guidance for me. Here I also want to thank the friends in IBM for their friendship during my visit and their indirect helps.

Finally, I should sincerely thank my family for their encouragement and support. Without their unconditional love during my most difficult time, I think it is impossible for me to complete this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

These years, the cloud computing has become increasingly popular for the Internet service offering. It greatly increases the operation efficiency of the users by providing infrastructure services as an alternative to implementing the real infrastructure. By leveraging cloud computing, the users can avoid the financial and operational costs from the efforts such as hardware and software configuration, and the electricity power management. According to the evaluations in [83] and [113], cloud computing can actually reduce the costs other than just transferring the expenditures from the systems to the cloud service providers.

Since the technical and economical benefits are seductive, cloud computing become a promising solution for the enterprise decision makers to host their Internet applications [110] [111]. However, migrating applications to the cloud has difficulties, and the frequently asked questions are as follows: 1) what are the scales of the number of users; 2) to be cost-efficient, which cloud servers should we choose and how many of them should we buy to accommodate those users; 3) how can we maintain optimal and stable quality of service under the fluctuation in the number of users? This is especially important when the number of users goes beyond the service capacities, and causes the degradation of the quality of service. However, these questions are still far from the reaches of the non-technology based enterprises, which can be possible cloud users. Developing an overall model to solve these frequently asked questions can clear critical barriers restricting the widespread uses of cloud computing.

In these thesis, we systematically work on the universal procedures of migrating an applications to the cloud servers, which aims to help the non-technology based enterprise users to do cloud migration. To do this, we should do a complete researches from the whole

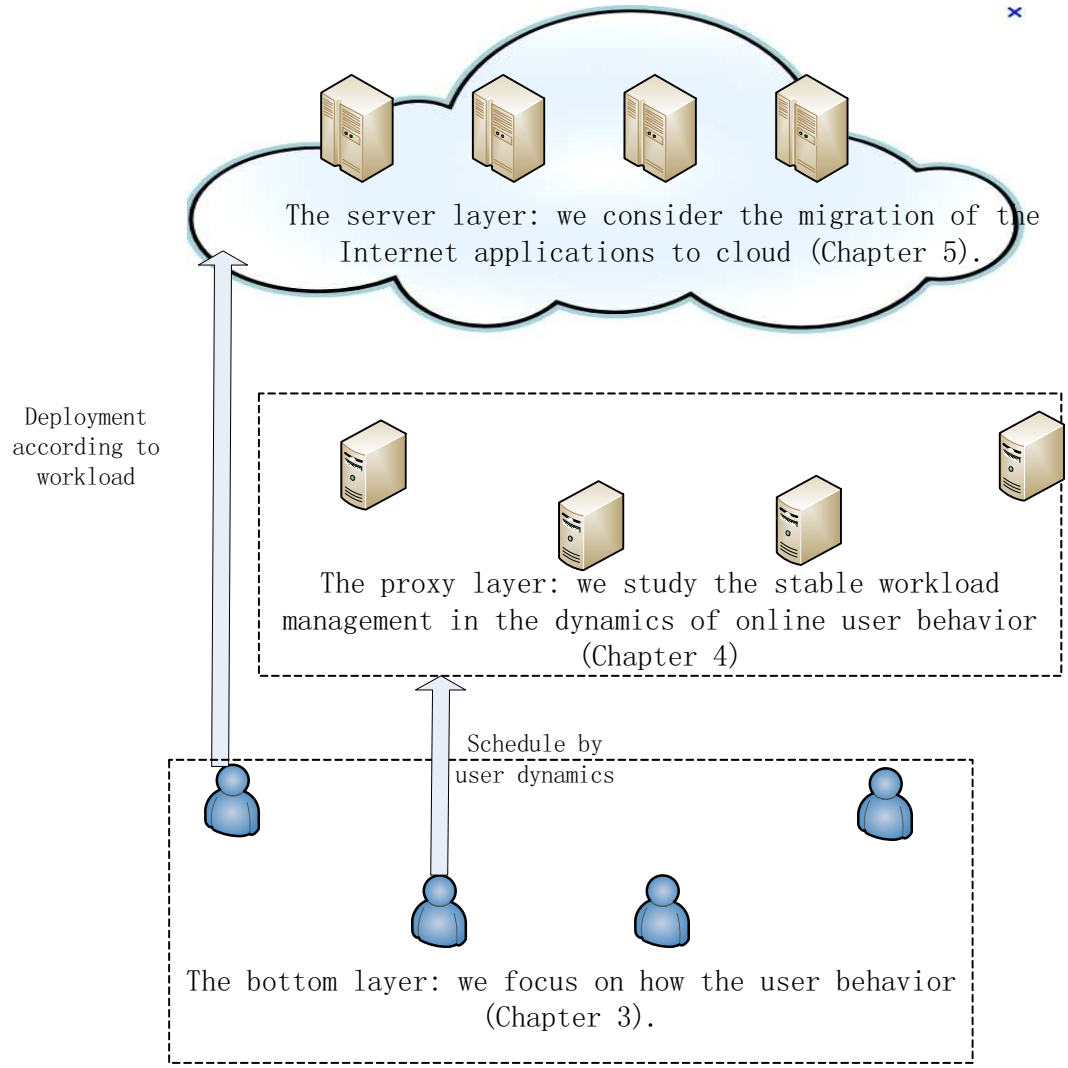


Figure 1.1. The research architecture of our thesis

cloud architecture. A widely adopted one [88] is shown as as Fig. 1.1. The whole system contains three layers. The bottom layer is comprised by the end users, which delivers their requests to the proxies of the cloud servers. In the middle layer, the proxies redirect the requests from users to the application servers in the top layer, with the objective to do the workload management. At last, the application servers in the top layer actually execute the applications and make responses to the end users.

To solve the previous addressed problems, we firstly measure and estimate the user traffic, and then do the workload management and the deployment according to result of the

estimation. In detail, the procedure is as follows:

- For the end users in the bottom layer, we study the how much a specific advertising scheme increase the user traffic. For his, we do a measurement and estimation. We analyze one type of special advertising features of the video sharing site, the external links, and try our best to characterize how much they affect the video popularity on the video sharing sites. Besides that, we also study their correlation with video popularity and other factors in the video sharing sites. We present this study in Chapter 3.

- For the proxies, we think about the workload management. Since that no estimation algorithm can assure 100situation of failing to accurately estimate the scale of the user traffic. However, underestimating the scale of the user traffic can cause the system overloaded and then degrade the user experience. For this, we study the workload management schemes to assure the user experience as well as stabilize the system performance (e.g. the request response time) under the fluctuating user traffic. This study is given in Chapter 4.

- For the application servers in the top layer, we study on a cost-efficient deployment scheme, according to the measurement and estimation result provided by the study in Chapter 3. The innovation of this part of study lies in using the cloud servers provided by different service providers. This study is given in Chapter 5.

1.1 The User Behavior of the Video Sharing Sites

1.1.1 Examples of the Links in Video Sharing Sites

The User Generated Content (UGC) sites gain world-wide popularity these years. In these sites, people are not only the information consumers, but they can also actively upload contents of their own. Different UGC sites have different emphasis. For example, Facebook is built as a general online community, Flickr is best known as a photo sharing site, and twitter is unique in its short message distributions. Among the UGC sites, this thesis will focus

Figure 1.2. An example of an external link of YouTube.

more specifically on video sharing sites, which are best represented by YouTube [106] and Youku [107].

The video sharing sites provide numerous functionalities to expedite video distribution. There are the related video links (See Fig. 1.3) which arrange videos by similar topics. There are mechanisms inside video sharing sites to organize users and videos together.

To further popularize the video distribution, video sharing sites introduce external links. An example of the external link is shown in Fig.1.1.1. We can see that for each video in YouTube, an embedded link is provided. The user can copy and paste this embedded link into anywhere such as their personal webpages, blogs, or even forums. When people watch the videos outside the video sharing sites, traffic and click counts go through YouTube. Clearly, the external links allow YouTube videos to be embedded in non-YouTube sites to attract views. This can further accelerate video distribution.

1.1.2 The Internal Interactions and External Interactions

A common belief of the success of the UGC sites is that the information generated by users can be distributed much faster through the UGC sites. Undoubtedly, in video sharing sites,

Figure 1.3. An example of an internal link, the related video link (R-link).

the information distribution is accelerated by external links and related video linked. From the previous subsection, we see that these external links are very different from those functions and features that arrange the internal contents, such as the videos and users. To be more generally, we can divide the factors that accelerate the distribution of videos into two types of interactions as follows:

- Internal interactions: we define the internal interaction as user-to-user, user-to-video, and video-to-video relationship inside the video sharing sites. Clearly, the related video links organize internal interactions.
- External interactions: We define the external interactions as the referencing of the videos outside the video sharing sites, such as hyperlinks to the videos. We can also see that external links facilitate external interactions.

However, to the best our knowledge, the current studies on the video sharing sites generally focuses on internal interactions. That is, the relationship of user-to-user, video-to-video and user-to-video relationship inside the video sharing site. We list the current studies on the internal interactions as follows:

- User-to-video relationship: one example of this relationship is the behavior of users viewing videos in the video sharing sites, which is represented by the popularity of videos. The studies on video popularity are such as [20] and [23]. Here, the conclusion that video total views follow the power distribution is made. Besides that, possible ways to improve the content delivery efficiency for video sharing sites, such as proxy caching, and peer-to-peer techniques are discussed in these three studies.

- User-to-user relationship: one typical example of the user-to-user relationship is the friend relationship in YouTube. This type of relationship in the video sharing sites is also studied in [70]. The observation is made that in the friend network in YouTube has a strong connected core, which is made up by users with large number of friends. It implicates that the content distribution, as well as the virus, can be accelerated by these strong connected cores.

- Video-to-video relationship: the video-to-video relationship can be represented by the related video links, which is shown in Fig. 1.3. The related video links are also studied in [100], and it is observed that the related video links can contribute to 30% of the video total views.

1.1.3 The Necessity of Studying External Interactions

Although the internal interactions, which are illustrated in the previous subsection, have been widely studied, to the best of our knowledge, there is still a lack of studies on the external interactions. Generally, the internal interactions affect the videos and the users inside the video sharing sites. Improving the internal interactions, such as the user relationship, or the related video links, can only directly affect the content popularity inside the video sharing site. Nevertheless, the external interactions are able to affect the users outside the video sharing sites. As it is shown in Fig. 1.4, we study the external interactions, as well as the relationship between the external and internal environment, so as to have the complete knowledge of the environment both inside and outside the video sharing sites, so as to further accelerate the content distribution.

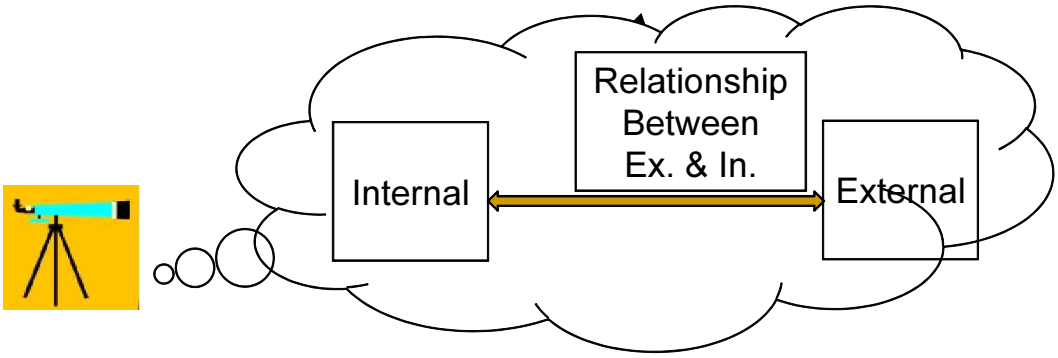


Figure 1.4. External and Internal interactions

Therefore, we study the external links of the video sharing sites in our thesis, as an example of external environment of the UGC sites. Compared with past studies on the interaction between users and videos within the video sharing sites, we are the first to concentrate on external links to videos of these sites. We aim to reveal the impact of these external links on the video sharing sites, as well as its correlation with the internal interactions. The detail studies on external links is given in Chapter 3.

The difficulties of the studying on the external links of video sharing site lies in the data collection. In detail, from YouTube and Youku, we can only have the information of "top" external links, which are calculated based on the number of views contributed to the videos since the videos were uploaded. Since we cannot have the information of all external links for one video, we admit that the lack of total information of all external links affects the accuracy of our studies. In the study of Chapter 3, we use the following techniques to overcome this difficulty.

- We fucus on the studies that are less affected, e.g. comparison of the total views from these external links in different video age groups.

- We may use curve fitting techniques to estimate the total views from unknown external links.

1.2 The Stability of the Resource Allocation of the Internet Applications

1.2.1 The General Internet Application Architecture

Just as other Internet applications, the cloud applications should also provide services for multiple users. While facing the great heterogeneity of the end user requests, resource allocation and workload management can be challenging tasks for the system service provider.

To cope with this difficulty, the service providers use service differentiation to achieve categorized monetization. The service providers use the Service Level Agreement (SLA) for the cloud customers. For example, SLA can specify several service classes with different response time targets and the importance or priority to meet these response time targets. Such kind of SLA indicates the following:

- Each user of the web applications belongs to one service class with one specific response time target;
- While the response time target is violated, penalty (such as the loss of money or the loss of user number) can be caused to the service providers.
- Therefore, for the service providers with such kind of SLA, minimizing the loss of money with a limited resource can be an important target.

To fulfill the minimization of the loss of money of service providers, a general system (adopted by IBM) is shown as Fig. 1.5, which uses three hierarchies: the front-end HTTP servers, the proxies, and the application servers. Their details are as follows:

- Generally, the HTTP servers are to filter out invalid requests.
- The proxy categorizes the request to its corresponding service class and assigned a *service rate*, and then dispatches the requests to application servers for execution.
- The application servers also estimate their maximum workload and provide feedbacks to the proxies. Such mechanism prevents the proxies to dispatch more requests to a server than that it can handle.

Therefore, we can see that the proxy is the key player in queuing the requests, workload management and SLA maintenance.

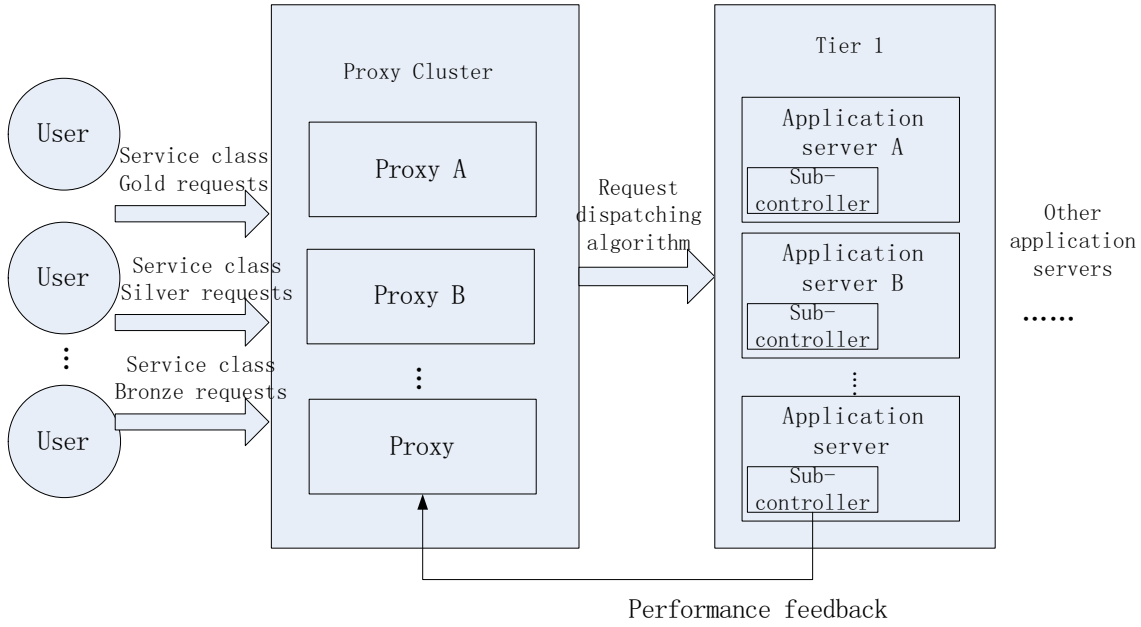


Figure 1.5. A General Architecture for the Cloud Environment

1.2.2 Methodologies for Resource Allocation

As the resources in the application servers are limited, allocating the service rate for different service classes becomes a challenge.

- Static resource allocation cannot satisfy our SLA system. This is because the workload is changing from time to time, static resource allocation would inevitably cause the resource over-allocation or under-allocation. Over-allocation for one service class results in the resource under-utilization. Under-allocation causes violation in *request response time* (the time between the request made by the user and the response received).

- For online resource allocation algorithms, which solve the resource allocation according to the arrival requests piece by piece, cannot have the global knowledge of the

request arrival rate. Therefore, the online algorithms are generally sub-optimal.

To achieve better resource allocation, we can use prediction-based mechanism to dynamically allocate the resources. We solve the resource allocation problems as follows.

- Firstly, we perform estimation for the request arrival rates of all service classes.
- Secondly, we perform optimization for the resource allocation, with the objective to achieve the optimal response time that minimizes the overall loss from the target response time violation.

Obviously, if the request arrival rates can be accurately predicted, the workload management can be optimal, and the minimization of the loss of money can be achieved. However, since there are no prediction algorithms assuring 100% prediction accuracy, the system still has underlying perils of workload management failure.

1.2.3 The Stability Problem in the Prediction-based Workload Management Scheme

In our thesis, we focus on the cloud applications with sequential jobs. As it is studied in [28], the request arrival rate of a single user is determined by the *inter-arrival time*, which is defined as the duration between the two continuous requests. As it is shown in Fig. 1.6, the inter-arrival time has two parts. The first part is the response time of the previous request, and the second part is the think time, which is the duration for the user to perform some actions after receiving the response and before initiating next request.

Therefore, we should notice that there is a close loop between the predicted request arrival rate and the response time (the time between the request initiated and the response received). Firstly, the predicted request arrival rate affects the response time. This is because that the assigned service rate increases as a larger request arrival rate is predicted, and vice versa. Therefore, the response time is affected. Secondly, the response time has impacts on the predicted request arrival rate in return. This is because the request arrival rate of each session is determined by the request inter-arrival time, which is comprised by the think time

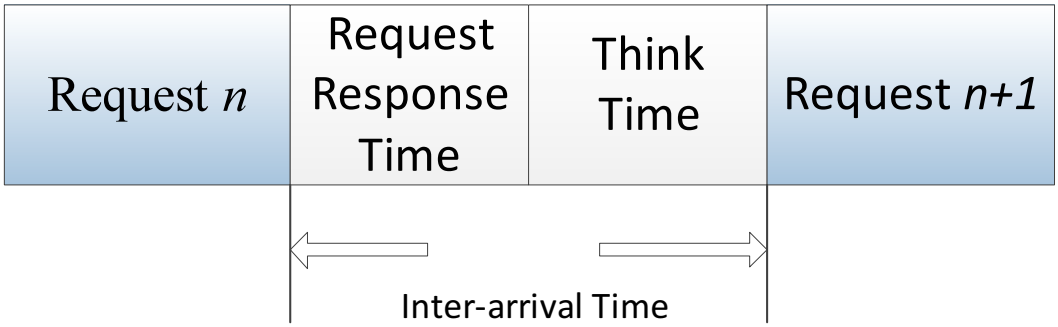


Figure 1.6. The Illustration of the Inter-arrival time

and the response time. Since the response time is affected, the actual request arrival rate is also affected. As the predicted request arrival comes from the past actual request arrival rates, the prediction is hence impacted.

Since the close loop exists, a question arises that whether the response time prediction error can reduce to zero if an error happens in request arrival rate prediction. To solve this problem, we have to face the following difficulties. Firstly, we have to model the instability, from the close loop between the predicted request arrival rate and the request response time. This model aims to illustrate in what circumstance the system becoming instable. Secondly, we have to find the solution for the instability problem, and we explore the solutions to control methods such as PID controller, Kalman filter. We also explore different types of Kalman filters for our problem. Thirdly, we should also balance between the prediction accuracy and the system stability.

In our thesis, we define this problem as the stability problem, and we focus on and then solve this problem in Chapter 4.

1.3 Multi-provider Selection Scheme For the Cloud Deployment

Cloud computing has become an increasingly popular paradigm for Internet resource accessing for these years. Generally, cloud computing service providers deliver three categories of

services over the Internet: software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). In this thesis, we focus on IaaS, which delivers the customized hardware resource, such as CPU cycles, memory as well as storage capacity as services over the Internet.

The key benefit of IaaS services lies in that they free the enterprise customers from focusing on the trivial matters, such as hardware and platform configuration, electricity power management and even the infrastructure cooling, and it just needs to simply run and manage the applications that enterprises require. However, so far as the IaaS services have developed today, we can witness there have been a large number of IaaS service providers available in market, such as Amazon EC2, GoGrid, and Rackspace.

Each of these service providers has a set of *service instances*, their "products", with different hardware configurations and also prices. The flourish of IaaS providers in the market brings various available service instances for the users. Nevertheless, the users may not always find a charming service instance from a single vendor as customer applications can vary more significantly than the service instances provided.

Recently, there are solutions and platforms developed that can manage different service instances from different IaaS providers at the same time, e.g., to monitor resource usages, to spotlight problems, to develop and to deploy an application on different service instances. For example, Zenoss [122], Nagios [115] and Splunk [119] can provide a uniform interface to monitor and spotlight cloud server problems, and Aneka [25] is designed for application development and deployment on different cloud service providers.

We believe in the foreseeable future, the number of cloud service providers and the number of independently developed tools that can manage different service providers will increase. It is therefore interesting for a user to think to host his applications using service instances from multiple providers, in order to achieve better cost-efficiency.

The benefit of the multiple provider strategy can be explained with an example. Suppose three IaaS vendors, A, B, and C provide a total of six service instances (see Table 1.1). Suppose the user application needs at least 13 CPU cores, 12.5 RAMs and 1060 storage

Vendor	Index	Service instance resource	Price
A	1	1 core CPU, 1.7GB RAM, 160 GB Storage	$8.5
A	2	4 core CPU, 7.5GB RAM, 850 GB Storage	$34.0
B	3	4 core CPU, 1GB RAM, 50 GB Storage	$50.0
B	4	10 core CPU, 4GB RAM, 200 GB Storage	$200.0
C	5	5 core CPU, 4GB RAM, 160 GB Storage	$24.0
C	6	5 core CPU, 8GB RAM, 320 GB Storage	$48.0

Table 1.1. Vendors and service instances

space. If we can only select servers from one vendor, the optimal solution is purchasing from vendor A, one service instance 1 and three service instance 2, with a cost of $8.5 + $34 $\times$ 3 = $158.5. If we can select from multiple vendors, we can choose one service instance 2 from vendor A, one service instance 1 from vendor B and one service instance 1 from vendor C, with a total cost of $108. This contrived example is not special. In this thesis, we will generalize this example into a multiple provider selection problem and conduct a systematic study.

While our idea is simple, there are many practical difficulties to solve.

- On the user side, user application requirements are diverse and the resource to be consumed by the applications is not as simple as the aforementioned example. Sometimes there are also applications that require high throughput communication between each other, which make the problem more complicated.
- On the service provider side, there are also large variants. For example, GoGrid provides special discount for heavy buyers, and this makes the price model from the service provide side even more complex.

Therefore, other than providing the model for common case, we also formulate the models

for the cases that the high communication throughput is required among different servers and also the wholesale is provided in Chapter 5.2.2. These models are solved as a subroutine of the common model in Chapter 5.3.4.

As a result, in the Chapter 5 of our thesis, we focus on selecting the service instances from multiple cloud service providers, with the objective of minimizing the costs of constructing the servers for the Internet applications.

1.4 Contributions

In general, the contribution of our thesis lays in the studies on the whole process of cloud migration, which includes the measurement of the user behaviors, the stable workload management scheme, and the innovative deployment methods using the service instances from different IaaS vendors. In detail, we can also specify our contributions of each part of the studies in this subsection as follows.

For the studies on the external links, we have the following contributions:

- We proposed to study the external links of the video sharing sites and we tried to quantify its impact. We believe this adds to the knowledge base, and could be useful for future comparison;
- We showed that the impact from external links is non-trivial and we also found substantial differences on the impact of the external links on YouTube and Youku;
- We conducted measurements on both external links and some important internal links and we studied their correlations.

For the studies on the stable resource management, our work is as follows:

- We analyze the root cause for one type of the cloud system being instable (i.e., service levels cannot be stably differentiated) through both analytical techniques and experiment traces. We notice there is a close loop between the response time and the request

arrival rate. We prove that if the request arrival rate prediction error is larger than a specific threshold, the response time prediction error will never converge to zero.

- We develop a Kalman Filter-based algorithm that maintain the system stable. It becomes the core of our SLA maintenance module. To the best of our knowledge, we are the first to study stable SLA workload management in clouds.

- To evaluate the performance of our design, we first conduct real experiments where we implement our design with three proxies and three application servers. We observe that our algorithm can successfully provide stable SLA service as compared to conventional proxy and a state-of-the-art proxy. Our response time is also small when the workload of the system is high.

At last, we also study the deployment of the Internet applications on the cloud servers from multiple service providers. We also have the following contributions:

- We model the Multiple Provider Selection problems, which is for the deployment of the Internet applications on multiple provider cloud servers.

- We prove that MPS problem is NP-complete, and we developed the following algorithms for it: the exact algorithm for a special case, the $(1 + \epsilon)$ fast approximation algorithms, and the greedy algorithm.

- We simulate the MPS problem, and we find in what circumstances the multiple provider strategy profits, and the applicable situations for different algorithms.

- We experiment the MPS problems and deploy a web game in three different IaaS service providers, and we find that the multiple provider strategy reduces the costs by about 30%.

1.5 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 gives the related work; Chapter 3 presents the study on the external links for two different video sharing sites: YouTube and Youku. The stable workload management scheme of the cloud servers is illustrated in chapter 4; Chapter 5 presents the details of multiple provider strategy for deploying the Internet application on the cloud servers. The future research work is proposed in Chapter 6. Finally, Chapter 7 concludes the thesis.

CHAPTER 2

RELATED WORK

In this study, we work on the whole process of migrating an Internet application to the cloud servers. Our work is from the user traffic perspective, which is affected by user behavior. We firstly study the video sharing sites as an example of the Internet applications, and measure how the user traffic is affected by a special feature, *the external links*. After that, we design a stable workload management scheme for the cloud servers in the fluctuating user traffic. At last, we propose a cost-efficient deployment scheme. Therefore, in the related work, we focus on three aspects: the previous studies on the user behaviors of Internet applications, the workload management of the cloud servers, as well as the cloud deployment methodologies.

2.1 The Studies on User Behaviors of Internet Applications

Nowadays, there have been a wide range of studies on the user behaviors of Internet applications. There are two types of Internet applications enjoying world-wide popularity currently, namely the online social networks and the video sharing site.

Both the online social networks and the video sharing sites belong to the User Generated Content sites (UGC sites). In such a type of sites, the users not only consume the contents in the sites, but also freely create the contents their own. Due to large success of the UGC sites, the characteristics of them are widely studied. In this section, we introduce the previous studies on online social networks and video sharing sites.

2.1.1 The Studies on the Online Social Networks

There are different types of online social networks. Early studies focused on general views of Facebook, a comprehensive online social network site [59] [91], Flickr, and a photo sharing site Flickr [84] [90]. Typically, a video sharing site, YouTube, is studied in [20]. This study analyzes the video popularity distribution in YouTube, especially the tail of video popularity distribution. Many sites with partially online social behaviors are also studied, for example, Slashdot Zoo [56], Orkut [22], MSN [61], online E-commerce [68] and a more general network [45].

Studies nowadays focus on specific aspects of the online social networks. Two important aspects are the user behaviors and the information contents.

A group of studies focus on the user behaviors. For example, the study in [95] develops a model to measure the continuous value relationship strength between the users, from weak to strong, between the users, instead of a binary friendship relationship. The experiment results shows continuous value relationship strength gives the higher autocorrelation and better classification. It is noticed in [62] that there are negative links (indicating the opposition relationship) in some online social network sites (such as Epinions, Slashdot and Wikipedia), and it finds that the signs of the links (either positive or negative) can be predicted in the online social network sites. An algorithm is developed to predict the attribution of these links, which can be positive or negative. The studied in [37] suggests methods to measure two important characteristics of online social networks social influence and homophily. Here social influence means users are likely to change their attributes to conform to their neighbor values, and homiphily means that users are likely to link other individuals with similar attributes. Other related studies on the user behaviors include [52] [72].

The contents of the online social network sites are also widely studied. For example, in [60], a method is presented for prediction of the future popularity of the news. To improve the content delivery, the geography information of users are extracted and exploited in [75] and a new mechanism is proposed. This content delivery mechanism is evaluated by the YouTube links on the social network site Twitter, and the results indicate that this

mechanism can improve the video cache hit rates. The study in [19] presents an automatic method for assessing the credibility of the message on Twitter, and the precision of this new method is about from 70% to 80%, compared with the human assessment. While facing the misinformation propagations,the study in [15] takes the notion of competing campaign in the social networks and tries to minimize the number of people adopting the misinformation.

There are also a flourish of interests in understanding other specific features of the online social networks. For example, in [38], the authors suggest a keywords generation scheme exploiting the "wisdom of the crowd", which suggest keywords by the associations between queries and URLs as they are captured by the clicks of users. Besides that, in the study of [76], a tag recommendation mechanism, which leverages the information in the search engine click logs, is recommended, with the objective to populate the targeted contents as much as possible.

Methodologies are developed to study the online social networks. For example, a sampling method based on random walk called frontier sampling is presented in [74] to lower the estimation error. A framework [9] is developed for bucket testing of social networks. An algorithm called HyperANF [11] is developed to efficiently estimate the neighborhood function of the online social networks.

2.1.2 The Studies on Video Sharing Sites

In this thesis, we study video sharing sites. The successful examples of this type of UGC site are YouTube, which enjoys world-wide popularity and Youku, the biggest video sharing site in China [103].

We are not the first to study the video sharing sites. Among past studies, measurement methodologies have been proposed. For example, in [12] guidelines are provided to do sampling in the video sharing sites and a framework is also proposed to study the popularity dynamics of user-generated videos. An important conclusion is that using key-word search videos as the seeds, videos would be biased towards the popular videos, while using recently uploaded videos, there would have no bias. The study in [100] provides a method to count the

total number of YouTube videos by leveraging the characteristics of video ids. This method is proved to be unbiased and the bounds of variance and confidence interval are offered, and it is estimated that there are about 500 million videos in YouTube by May, 2011. A globally distributed active measurement platform is established in [2] for YouTube video delivery system.

The analysis on the video sharing sites widely spans to user-to-user, user-to-video, and video-to-video relationship.

For example, the video popularity distribution of YouTube is analyzed in [20], where the long tail of video popularity distribution is analyzed. More aspects of YouTube are analyzed in [24], such as video life cycles, user viewing behaviors, and the small world phenomenon. In [42], the traffic of YouTube in campus is characterized. It shows that there is strong correlation between videos viewed (watched) on consecutive days. This work also demonstrates that caching can improve user experiences, reduce bandwidth consumption and lower the burden of YouTube. The recommendation system in YouTube is studied in [101]. It shows 30% of the video total views come from related video links.

The user behavior in the video sharing sites is also widely studied. For example, the study in [33] focuses on the YouTube video uploaders, and demonstrates positive reinforcement between online social behavior and uploading behavior. In [36], it is shown that user access patterns are similar in different locations and their access devices (either PCs or mobile phones): they usually use default video resolution and player configuration. The work in [14] shows an interesting result that online video consumption appears geographic locality of interest.

The video-to-video relationship is analyzed in [96], and it shows that more than half of the YouTube videos contain re-mixed video segments, and some particularly popular videos are correlated with virus.

There are suggestions to improve YouTube. NetTube is developed using a peer-to-peer structure for YouTube [24]. An algorithm using geographic information to improve multimedia content delivery in YouTube is suggested in [75]. The study in [98] suggests an

improved semi-supervised training method for classifying YouTube videos by using video labels and co-watch relationship (videos watched in one session).

There are also studies that compare YouTube with other UGC sites. For example, the video popularity distribution of four different video sharing sites is characterized in [71]. It shows that the life time video popularity have relevance with caching size. A comparison of different UGC sites (including YouTube) can be found in [70] and observations of the free scale, small world and strong connected cores, are drawn. The study in [43] compares the user sessions between the video sharing sites and the traditional web sites. It concludes that the YouTube users have larger data traffic and longer think time.

We can see that existing studies on video sharing sites all focus on internal interactions; that is, the content-to-content, user-to-content, and user-to-user relationship inside the UGC sites. In this thesis, we focus on understanding the characteristics of the external links. We have some measurement and analysis of the related videos of YouTube and Youku; though the emphasis is on comparison with the results of external links. We have two preliminary works [57] [58]. In [57], we present a short study on YouTube only. This thesis substantially extends [58] to the domains such as the correlation between external links and internal links in all video age groups.

2.2 The Workload Management for the Cloud servers

2.2.1 Overview of Studies on Cloud Computing

The general surveys of the cloud computing are such as [7] and [97], where the obstacles and the state-of-the-art researches on clouds are illustrated. One important conclusion is drawn in these two papers that cloud computing is potential to realize the dream of using computing as a utility.

To provide the cloud computing as a utility, the cloud service providers focus on the performance of cloud servers, the pricing mechanisms, and the energy saving schemes. Compared with the service provider, from the consumer perspective, the privacy and a cost-

effective deployment scheme are the key issues the consumers concentrate on. As a summary, some examples of these aspects are listed as follows.

- The performance of the cloud servers. These studies focus on the performance such as the Service Level Agreement (SLA), the deadline of the user tasks, and the response time of the user requests.
- Energy consumption of the clouds. Large scale cloud servers have brought the concerns of wasting energies, and the studies on this aspect engage in reducing the energy costs.
- The economy of the cloud service. The pricing of the cloud service as well as the cost minimization for constructing cloud service for applications will be discussed in this aspect.
- Enterprise application migration problems. These studies focus on the privacy of the enterprise user data.

The performance of the cloud servers has been widely studied. For example, The study in [48] argues that it is not thoughtful that the current SLA mechanism only takes computation resource and storage resource into consideration. It proposes SecondNet, which further considers the bandwidth resource, as well as computation resource and the storage space in SLA. The study in [13] notices that different types of servers may play different rolls in applications, and have different response time. In this case, it presents AriA, which schedules different types of servers, to meet the application request deadline. Study [85] presents Nefeli, a virtual infrastructure gateway that is capable of effectively handling diverse workloads in cloud environments. At last, study [88] presents an algorithm named DONAR, which provides an optimal replica-client selection for the users to minimize the user request response time.

Compared with the performance, there are also many studies focusing on the energy consumption of the clouds. Study [73] presents a cost, energy consumption and performance

comparisons between different types of cloud frameworks, such as Fat-tree, de Bruijn, and BCube. Compared with this, the study in [53] notices that there is a large energy consumption as well as performance reduction while a VM migration occurs. Based on the migration costs, an algorithm, namely Mistral, which optimizes VM migration, is provided.

Besides the performance and the energy consumption problems, there are also many studies concentrating on other aspects of cloud computing. For example, to analyze the performance of different clouds, the study [10] proposes a benchmark model for cloud computing; to lower the costs of cloud upgrade of heterogeneous equipment, the study in [29] presents an upgrade strategies called LEGUP, which flexibly upgrades the system and remains the cloud system a high performance; at last, noticing that some enterprise application modules may be confidential, the study in [50] provides an enterprise application migration scheme, which help enterprise application migrate to cloud, without exposing the confidential component to the Internet.

2.2.2 The Framework of the Cloud Servers

The framework of the cloud servers is complex. According to the study in [73], the cloud server framework can be divided into three categories in the view of high level, which is as follows:

- The switch-only architecture: the packet forwarding task is only done by the switches. A typical example of switch-only architecture is the three-tiered design recommended by Cisco [26]. The further study on this architecture is such as [44] and [5].
- Server-only architecture: in this architecture, the servers both execute applications and forward requests to other servers. An example of this is suggested in study [1].
- Hybrid frameworks: examples are such as BCube [46] and DCell [47]. This architecture use both servers and proxies to forward requests. Note that although our SLA maintenance module is developed for a real industry cloud, it is designed to be framework dependent and can be widely applied.

2.2.3 The Workload Management of the Cloud Computing

These years, the prosperity of the cloud computing has brought the necessity of devising an efficient workload management mechanism for servers with the Quality of Service (QoS) assurance. The key characteristic of the cloud computing is that it does not consume resource on the user side (hardware, software, and technical maintenance). The service providers have the responsibility to provide service, and also handle multiple users and requests, and server workload management.

As it is studied in the previous section, the user behaviors of the Internet applications, which is represented by the online social network sites and the video sharing sites, are distinct from one to another. Besides that, the user traffic also vary from time to time and it is hard to be predicted. To migrate the Internet applications with the fluctuating user traffic to cloud servers, devising a workload management can be a critically important issue to maintain the system in a stable performance.

Facing the increasing traffic of the Internet, the cloud service providers need to improve the service capacity by replicating minor-servers. To achieve the overall efficiency of the replicated servers, one of the important tasks is the do load-balancing among the different replicated servers.

According to the study in [18], the load-balancing of the servers can be divided into the following four types: the client-based approach, the DNS-based approach, the dispatcher-based approach, and the server-based approach. Their details are as follows:

- The client based approaches: the routing of the web servers is carried out by the web browser, the applet applications in the web clients, or by the client-side proxy servers. The advantage of this approach is that this approach does not consume the resource on the server side, no matter the software resource or the hardware resource. However, the drawback is that it requires the deployment on the client side and the applicability is limited.

- The DNS-based approaches: as it is shown in Fig. 2.1 the routing of the web servers is carried out by the authoritative DNS servers. The DNS servers dispatch clients to different servers according to the different client states (such as the client physical locations or the client IP addresses). The client can be also dispatched to different servers by the specific time of sending requests. The advantage of DNS-based approach is that it does not generate any extra bottleneck for the web server system. However, as the DNS server cannot be aware of the server load condition, it can only provide coarse-grain control for the load-balancing.

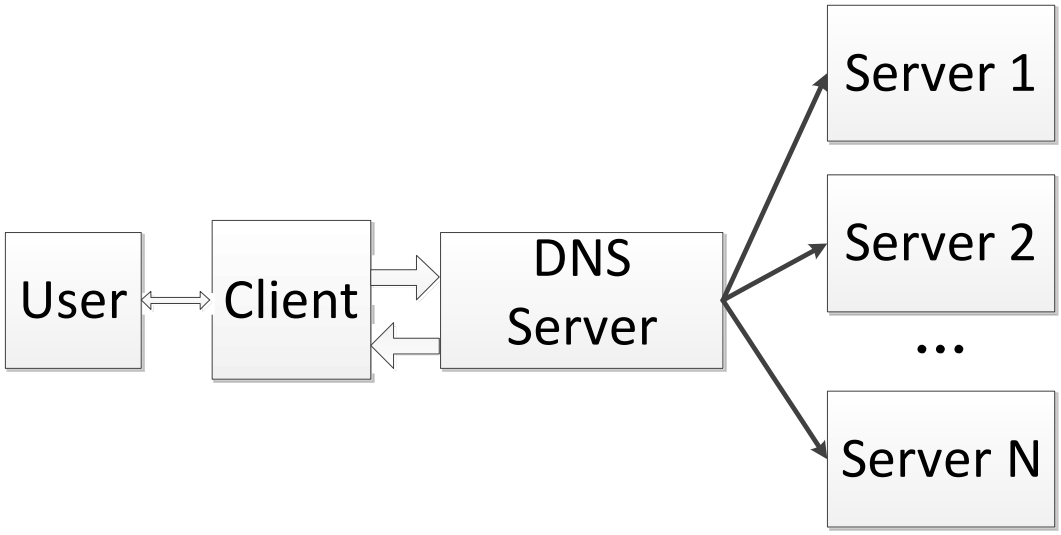


Figure 2.1. The DNS-based approach for the load-balancing control

- Dispatcher-based approach: to achieve a fine-grained control of the routing of the user request, the web server administrators may deploy dispatchers for the web server systems. These dispatchers have the load condition of the servers, and it can achieve the fine-grained load balancing of the request routing. The detail is shown in Fig. 2.2. However, the disadvantage of dispatcher-based approach is that it requires the deployment of the dispatch servers and it also introduce extra bottleneck.

- Server-based approach: to achieve load-balancing, in this approach, a server reassigns a received user request to another server by using the HTTP redirection. The details

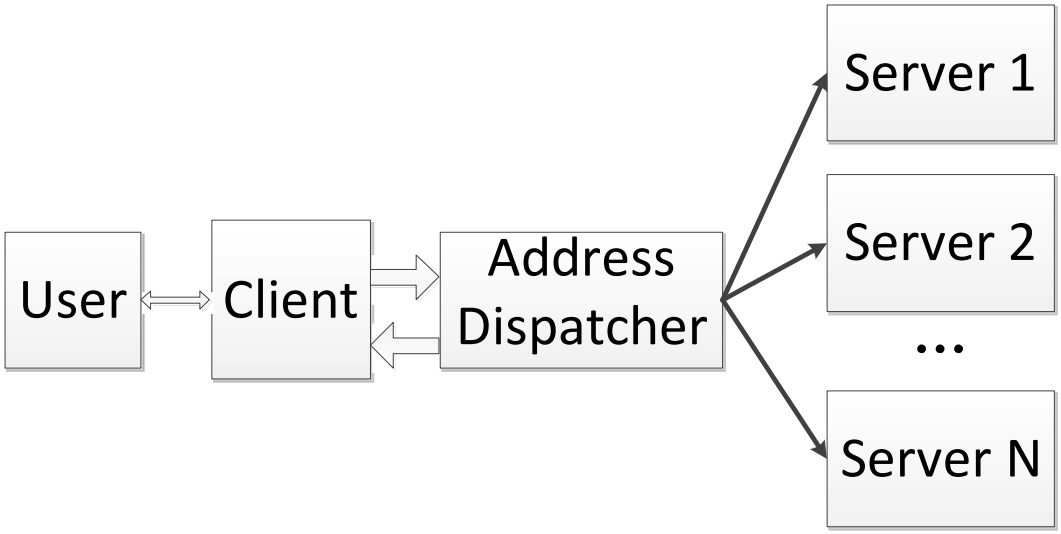


Figure 2.2. The Dispatcher-based approach for the load-balancing control

is shown in Fig. 2.3. The client firstly visits Server 1 according to the DNS result. Secondly, server 1 sends HTTP redirection packets to the client according to the load conditions of all servers, and hence the client sends the request to the redirection server. This approach can also achieve fine-grain control but it also introduces extra latency because of the redirection of the user request.

In our thesis, we focus on the dispatcher-based method, which fully controls the user requests according to the knowledge of the server condition. More than just doing the load-balancing, our thesis also aims to achieve the minimized loss of money from the violation of the SLA, by controlling direction rate of requests from the dispatchers (called proxy in our thesis) to the servers.

2.2.4 The Quality of Services and the Feedback Control on Cloud Severs

There are studies on the Quality of Services (QoS) on Internet applications using Service Level Agreement. For example, in [63], a method for maximizing profits in a general class of e-commerce environments is proposed. This method formulates the profit optimization problem, which is presented based on queuing model and the service differentiations are

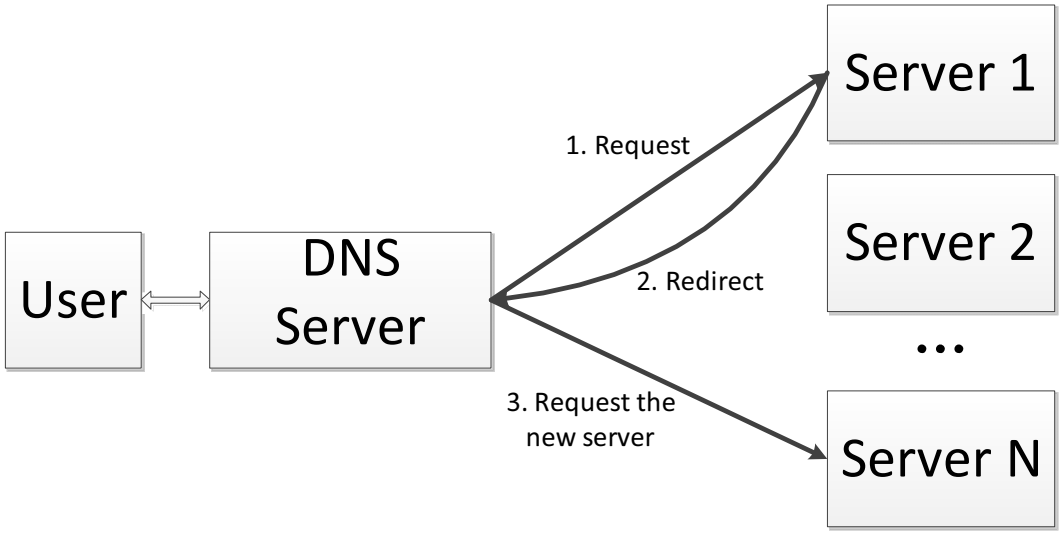


Figure 2.3. The Server-based approach for the load-balancing control

taken into consideration. A dynamic resource provisioning algorithm is proposed in [31], with an objective of satisfying the pre-defined end-to-end response service level agreement. At last, a surrogate model to limit violations of the SLA in the enterprise applications is suggested in [39]. This study suggests using Virtualized Data Centers (VDCs), which are able to dynamically change the execution environment to react to unexpected changes of the environment, such as the sudden workload burst.

For the stability of the web server performance, there are also many studies leveraging the feedback control methods. However, these studies generally focus on the guarantees of the end-to-end delay of SLAs. For example, the study in [77] uses a feedback control approach and the queueing model to keep the web server performance (in the basis of the request response time) close to the service level specification. However, in this study, the mutual impact between different service classes is not considered. Compared with this, another study in [66] notices the relative delay in different service classes, and the feedback control with the queueing model approach is also adopted.

Our work furthers the studies of workload management and monetization based on SLA. Our work aims to minimize the loss of money from violation of the SLA. Compared

with the two previous work, our work not only aims to assure the delay, or the relative delay between the different service classes, but we aims to minimize the money loss from the SLA violation. To the best of our knowledge, the closest work to ours is the [99]. Compared with that, we further notice that there is a close loop between the predicted request arrival rate and the request response time in the sequential job scenario. Here, we target to maintain the minimization of the loss of money under this feedback, which is neglected in the previous studies. Our problem comes from the pre-research of a large scale industry application by IBM (we do not disclose the application name due to anonymity requirement). We believe, however, the problem of a stable SLA is universal for cloud computing and can be applied to other applications.

2.3 The Cloud Migration Methodologies

Currently witnessed a wide range studies on cloud computing. The service of cloud computing, which is divided into Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), are warmly welcomed by individual users and enterprise users. This is because cloud services are hosted by cloud service providers, and the users are not required to actually deploy and maintain the software, operating system, or the hardware of the computer system. They only need to have a basic computer with Internet access to acquire the service. Therefore, in our study, we consider migrate the Internet applications to the cloud servers, with the objective to minimize the deployment cost.

As cloud computing opens the door of providing the computing resource as a utility, the monetization of the cloud becomes a key problem both for cloud service providers and customers.

From the centric view of service providers, a good pricing strategy and a cloud service model keep the cloud products highly competitive in the market, and the existed researches are as follows: one example of the cloud service model is the study in [17]. This study proposes a cloud computing service model focusing on the customer business requirement, with the objective to build a cloud ecosystem that contributes to the sustainable

development of the cloud. The study in [49] notices the unfairness of the current pricing scheme and then proposes a pay-as-you-consume pricing scheme based on machine learning prediction model to achieve pricing fairness. The study in [86] tries to bridge between distributed systems and economics by a pricing scheme that decouples users from cloud providers. Compared with that, the study in [6] tries to find the generalized Nash equilibrium in price between SaaS service provider and IaaS service provider, for the situation of the SaaS service providers hosting their applications in IaaS services. Finally, study in [79] proposes an adaptive pricing scheme allowing resource reservation, which encourages users to use resource more careful.

On the cloud consumer side, selecting cloud products with reasonable price can greatly lower the expenditures. Therefore, researchers consider both from performance requirements and the budget. The study in [67] dynamically allocates or deallocates VMs and schedules tasks on the most cost-efficient instances, to ensure deadline requirement with minimum financial cost. To minimize the economic cost and meet the deadlines, the work in [55] studies the problem of resource allocation for real-time tasks.

To the best of our knowledge, there are no studies focusing on lowering costs of constructing cloud service by selecting service instances from multiple service providers yet. Nonetheless, the closest study to us is the study in [78], which minimizes the rental cost for provisioning resource, as well as the transition cost for reconfiguring resource for different application requirement in different time. Compared with that, our study makes full use the different performance price ratio of service instances offered by different service providers, and then lowers the service construction costs according to this characteristic. Our study result shows that selecting service instances from multiple service providers is able to large reduce the service construction costs.

CHAPTER 3

UNDERSTANDING THE EXTERNAL LINKS OF VIDEO SHARING SITES: MEASUREMENT AND ANALYSIS

3.1 Overview

This chapter presents the measurement study on the external links for the video sharing sites, with the objective to understand how the external links affect the user traffic. In detail, we study the external links on the following aspects.

- We try as much as we can to quantify how much the external links contribute to the popularity of the videos on the two video sharing site: YouTube and Youku.
- We analyze the relationship between the number of external links and some of the internal factors, such as the video popularity and the number of related video links.
- We also present the relationship between the number of external links and internal factors of the videos with different video age groups.

The remainder of this chapter is organized as follows. Section 3.2 discusses the background and measurement methodology. The impact of the external links on the videos is given in Section 3.3. In Section 3.4 we study the correlation between internal links and external links. In Section 3.5, we focus on the evolution of the external links by studying the external links of the videos of different uploading time. Finally, we conclude this chapter in Section 3.6.

3.2 Background and Measurement Methodology

3.2.1 Background and Motivation

Currently, there is rapid growth of the popularity of user generated content web sites. One key feature of these sites is that the users are not only the information consumers, but also actively upload contents of their own. One notable class of UGC sites is for video sharing, represented by YouTube and Youku. These video sharing sites have attracted a great number of studies in the recent years. These studies, however, focus on user-to-user, user-to-video or video-to-video relationship within these video sites. To distribute the content videos more widely and to attract more users, these video sites provide external links for videos. Users can easily obtain an embedded link of a video and paste the link to any web pages in other web sites, such as forums, or their blogs. In this thesis, we define the *internal links* as those maintaining a relationship within the web sites. These links include the user-to-video, user-to-user, video-to-video relationship. We define the *external links* as the links to the videos that are embedded in other web sites.

These external links are important for improving the popularity of the videos; however, there is no rigid study to quantify the effectiveness of these external links. Therefore, we would like to know as follows.

- The impact of the external links on videos, e.g., how many views are contributed by external links;
- The relationship of external links and internal links; their differences, interactions and correlations.

Such curiosities motivate the studies in this chapter.

3.2.2 Measurement Methodology

Our experimental data sets come from two video sharing sites, YouTube [106] and Youku [107]. YouTube is one of the largest video sharing sites in the world and in 2009, it accepts

1.886 billion views [103] every day. Youku is the most popular video site in China [103], with views of 40.9 million per day [103].

The necessary data for our study are 1) a large number of randomly collected videos, and 2) the external links of these videos.

We first built a crawler to sample a large base of videos. In principle, we start our data sampling from seed videos and follow their related videos in the crawling. We follow [35] [12], where it is shown that if those recently-uploaded videos do not link popular videos as their only related videos, the data collection will not be biased by choosing the seed videos from recently-uploaded videos and using breadth first crawling.

In detail, this crawler initiated $\mathcal{N}$ threads to retrieve the video information from the video sharing sites. In our practice, we choose $\mathcal{N} = 10$ to assure the best crawling performance. If $\mathcal{N} > 10$, the servers regarded our crawling as attacks and rejected all our connections. Each thread works as Algorithm 3.2.1 and recursively follows the routine of 1) getting an un-crawled video from a queue (we name it *the task queue* afterward); 2) retrieving the information of the external links of the video; 3) retrieving potential un-crawled videos from the related videos; 4) discarding crawled videos, and queuing the crawled videos at the tail of the task queue. More specifically, each time the crawler gets the video from the head of the task queue for crawling and queues the new videos at the tail of the task queue to assure breadth-first crawling.

More specifically, for YouTube, we started our sampling on Mar. 24th, 2009 using the "recent" videos as seeds. We recursively crawled all the related videos for seven days until Mar. 31st, 2009. In total, we have collected 1.24×10^6 videos from YouTube. For Youku, unlike YouTube, there is no category of "recent" videos. Therefore, we used all videos in the main pages, which were uploaded within one day, on July 8th, 2009 as seeds, and recursively collected the related videos for five days. In total, we have collected 1.43×10^6 videos. We admit that for these videos, they might have more views.

To collect the external links, we used a universal Java Script engine provided by Google [108]. This engine can parse the Java Script codes from video pages, so as to track

Algorithm 3.2.1 VideoCrawling()

Input: The task queue

Output: The information of external link in the data base

```
1: while true do
2:     Retrieve video v from the task queue;
3:     Retrieve the information external links and views from external links
4:     and store in the database;
5:     Retrieve a related video set RV of video v;
6:     for rv ∈ RV do
7:         if rv is not crawled then
8:             Store rv at the end of the task queue;
9:         end if
10:    end for
11: end while
```

the external link information maintained by YouTube and Youku internally. With this engine, we could get the URLs of the external links as well as the number of views of each external link. However, from YouTube and Youku, we can only have the information of "top" external links, which are calculated based on the number of views contributed to the videos since the videos were uploaded. YouTube maintains the top-5 external links of each video. We have not found a method that can collect the information of all the external links of videos. Intrinsically, if YouTube does not provide an interface to release such information, unless one can explore the entire Web, it is unlikely that all external links can be collected. In our study, we use the top-5 external links for YouTube videos. For Youku, we used similar method. Youku provides more information, and we obtained the total number of external links for each video, the URLs and the total number of views of the top-20 external links.

We admit that collecting information only from top external links affects the accuracy of the study. Our argument is that, on one hand, part of our studies, e.g., comparison of the total views from these external links in different video age groups, is less affected by the total

external links. On the other hand, for the Youku trace, we have the total number of external links, and the number of views of each top-20 external link. Thus, we have a strong basis to analyze the remaining part of the views of external links. We did find that the average views from external links as a function of the rank of external links in a large sampling space fit the power law function with an r-square over 99.8% (see Section 3.3.2). Therefore, we are more confident that our observations of this study are close to reality. In the remaining part of our thesis, we simply say external links of YouTube and external links of Youku, which should be understood that we denote the top-5 external links for YouTube and the top-20 external links for Youku, given that there is no ambiguity.

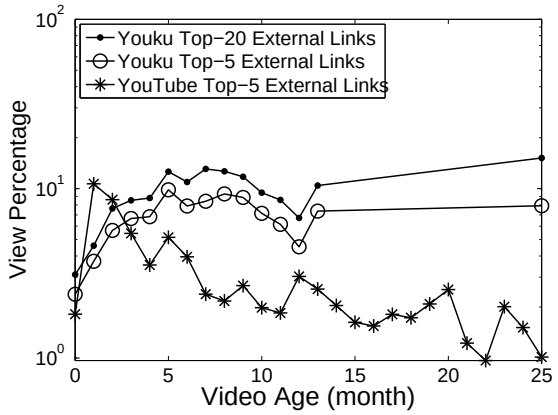
Besides the external links, we also collected the information of internal links (we focus on the related video links) for a comparison study. For such data collection we adopt the same strategies as [20]. We also started on Mar. 24th, 2009 and July 8th, 2009 and carried out a data collection of 7 days and 5 days for YouTube and Youku respectively. The data collection is a breadth first search, following the related video links.

3.3 The Contribution of the External Links

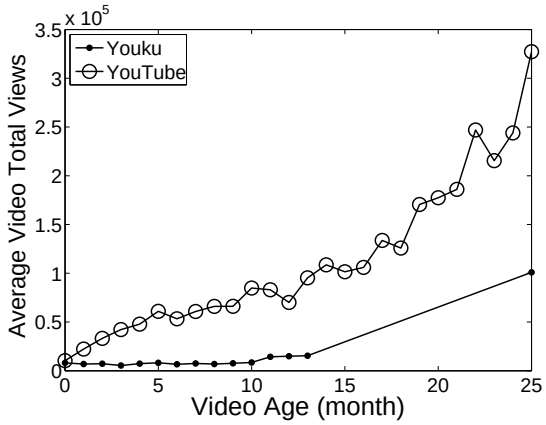
3.3.1 Overall Contribution of External Links

We first show the impact of the external views on the videos in Fig. 3.3.1. We classify the videos according to their *ages*, i.e., the total duration since they have been uploaded to the video sharing sites. Note that YouTube provides the upload date for each video, whereas Youku provides a rougher estimation of how many days or months or years a video has been uploaded. For example, the videos uploaded 13 months or 14 months ago in Youku will both be labeled as 'uploaded one year ago'. As such, the points of 13-month and 25-month in our figures for Youku stand for the videos uploaded one year and two years ago. Note that our results are not affected as the points in our figures are the average (not accumulative) number of views.

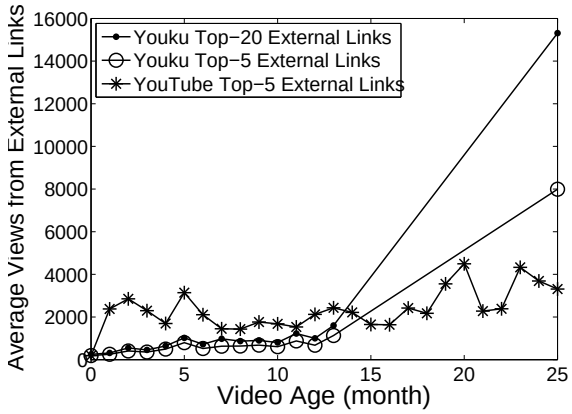
In Fig. 3.3.1 (a), we show the percentage of the views that come from the top external



(a) External views vs. video age



(b) Total views vs. video age



(c) External views vs. video age

Figure 3.1. The contribution of external links on the popularity of videos on videos sharing sites.

links. We see that for the videos in YouTube with an age of two months, 10% of the views come from the top-5 external links. For videos with an older age, the percentage of the views from external links gradually drops to around 2%. For Youku, the impact of external links is much higher. For most of the videos, more than 8% of views come outside the video sharing site. For videos with an age of 24 months, views from external links can contribute as many as 15%. Even considering the top-5 external links of Youku, they contribute about 6% - 9% of total views, which is still more significant than YouTube.

To explain the situation more clearly, we show the specific number of the video total views and views from the top external links as a function of video ages. In Fig.3.3.1 (b) we show the total views, averaged per video, for different video age groups (this includes both views from internal links and external links). The total views increase steadily for both YouTube and Youku as video ages increase. It is also clear that YouTube attracts much more views than Youku. This is not surprising as YouTube is more popular world-wide. In Fig.3.3.1 (c), we show the total views from external links (averaged per video). We see that for YouTube, the total external views are comparatively stable among all video age groups whereas for Youku, the total external view increases. In addition, though the external views of YouTube are still greater than that of Youku, the differences are not as big as the total views. This explains why the percentage of external views of Youku is more significant than that of YouTube in Fig. 3.3.1 (a).

Such differences of the impact of the external links on YouTube and Youku do not conform to our expectation. We consider a possible explanation can be as follows. YouTube is a video sharing site of world-wide popularity. As such, the external links are widely spread to external websites all over the world. These external websites may not be world-wide popular, however. Thus, the external links cannot obtain world-wide popularity and have less impact (in terms of percentage). Youku, on the contrary, has popularity within China only. The external links are also on the Chinese-based websites and can have China-wide popularity. As such, the comparative impact of external links on Youku is much higher than on YouTube. Based on our current data, we are unable to verify this. As a first work on external links, we confine ourselves to the fundamental problems such as the correctness

of the data collection, and the understanding of the basic characteristics of external links, as will be presented in the remaining part of this chapter. We will leave such questions to our future work.

3.3.2 The Number of External Links

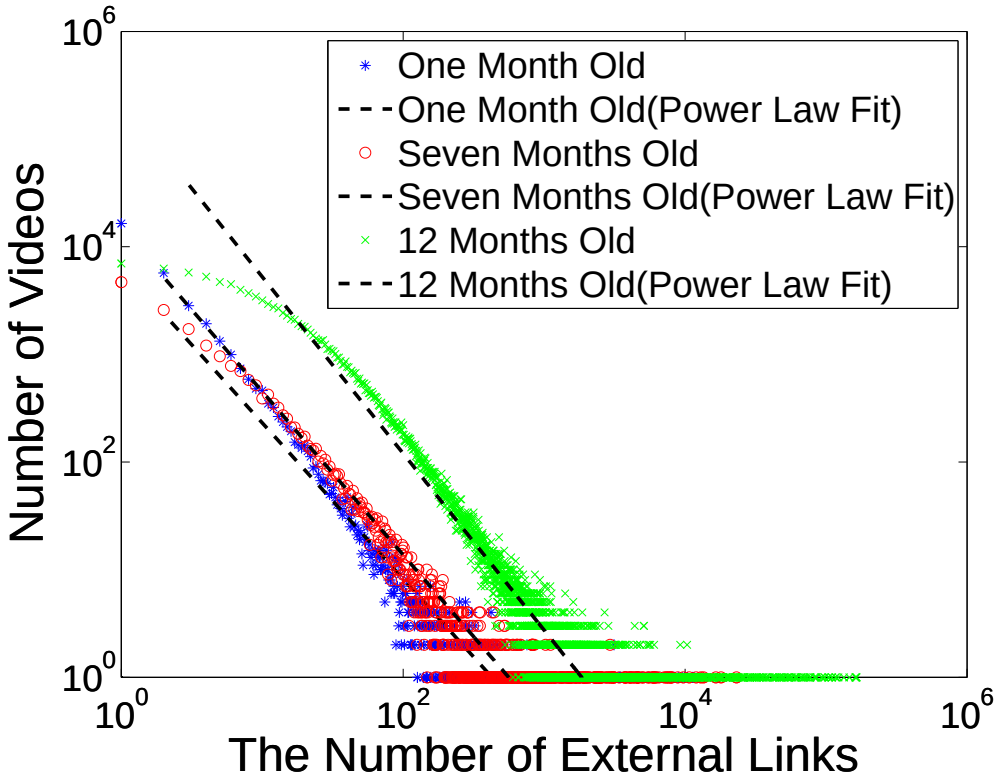


Figure 3.2. The distribution of the number of external link of videos in different age groups.

Fig. 3.2 plots the number of videos as a function of the number of external links in a log-log scale for different age groups. Since YouTube cannot provide the total number of the external links for each video, we only study Youku in this figure. We can see clearly that a small portion of videos enjoy the majority of the external links. With the methodology introduced in [27], we therefore use Maximum Likelihood Estimation (MLE) to fit power law distribution $f(x) = \alpha \times (\frac{x}{x_{min}})^{-s}$ (Table 3.1 shows the specific parameters α, s and x_{min} for each age group) to the sampled data. We find that in Youku the power law distribution fairly matches the external link distribution. This is especially true for the videos with a

t (month)	$\alpha(t)$	*s(t)*	x_{min}	KS statistic
1	6.508×10^3	1.450	20	0.018
7	1.490×10^4	1.519	80	0.0052
12	2.293×10^5	1.806	120	0.0016

Table 3.1. The Parameters of Power Law Fits for Fig. 3.2

large number of external links, i.e., the tails of the distribution. We find that the 'tails' follow power law with Kolmogorov-Smirnov statistics (KS statistics) respectively 1.8%, 0.5% and 0.2%. This is not entirely surprising. Notice that the power law behavior has been observed in various properties of the video sharing sites. For example, it is shown that the distribution of the number of video views in YouTube also fits the power law [20] [23].

We classify the videos into three age groups (one month, seven months, and 12 months). We study the percentage of videos with more than ten external links. We see that different age groups show clear trend: for the videos in age group of one month, 13.4% of videos have more than ten external links; for the videos in age group of seven months and 12 months, 33.0% and 64.8% of videos have more than ten external links. This shows that in Youku older videos, on average, may get a larger number of external links.

3.3.3 The Views from External Links

In this subsection, we study the views contributed by external links. We still use the Youku data set for analysis. Fig. 3.3 shows the views of the external links as a function of the ranks of the external links in a log-log scale. Note that the *rank* is from one to 20 as we have the views of the top-20 external links. We plot the data for the videos with an age of one month, seven months and 12 months in Fig. 3.3. We found the data approximately follow power law function where there are deviations at the tails. The power law means that most of the external views come from the high rank external links.

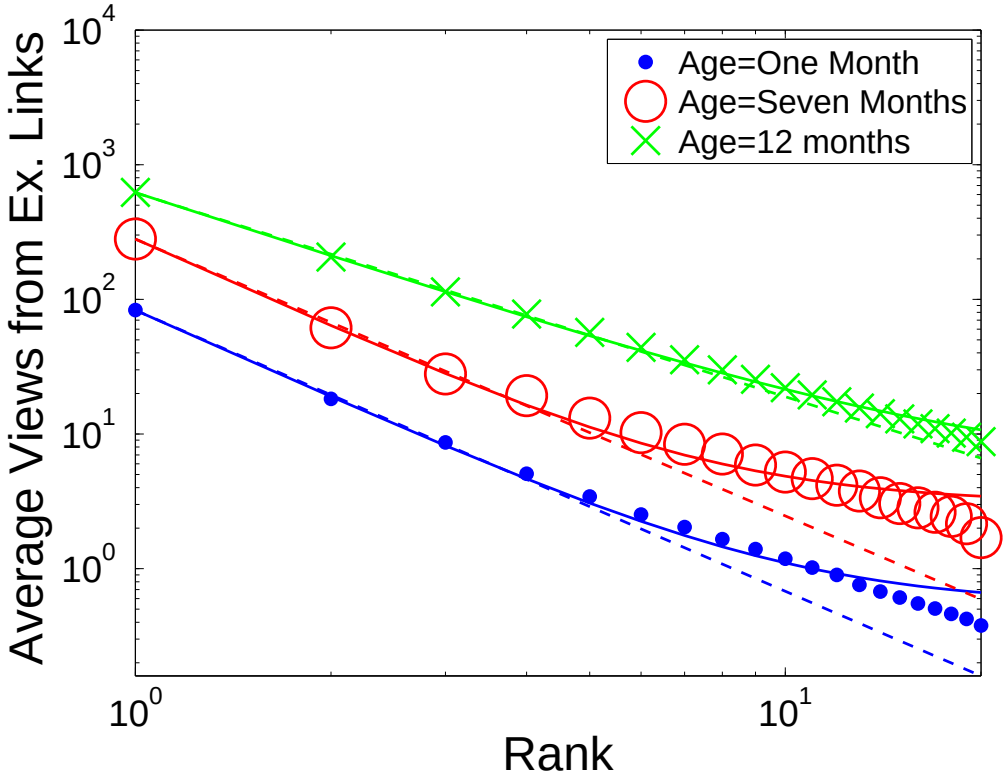


Figure 3.3. External views as a function of rank (all videos). The dashed lines shows the power law fit lines $\mathcal{V}_1(t) = a_1(t) \times r^{-p_1(t)}$, and the solid line shows the fits $\mathcal{V}_2(t) = a_2(t) \times r^{-p_2(t)} + \epsilon(t)$

To further analyze how the views of external links decay as the rank of external links increases, in Fig. 3.3, we use the least square method to fit the original data with power law function $\mathcal{V}_1(t) = a_1(t) \times r_1^{-p_1(t)}$ where t is the age of the video group, r_1 is the rank of external links, $p_1(t)$ is an exponential factor and $a_1(t)$ is an adjustment factor (Table 3.2 shows $p_1(t), a_1(t)$). Since the tails have deviation, we also plot a power law function with a deviation term ϵ, $\mathcal{V}_2(t) = a_2(t) \times r_2^{-p_2(t)} + \epsilon(t)$ (Table 3.3 shows $p_2(t), a_2(t)$, and $\epsilon(t)$). We can see that 1) the power law functions fit the original data well (as it is shown in Table 3.2 and Table 3.3, all the $\mathcal{V}_1$ and $\mathcal{V}_2$ fits are with an r-square greater than 0.99); 2) the views from high rank (larger than the rank of ten) of external links are slightly larger than the power law fit $\mathcal{V}_1$ but slightly smaller than the fit $\mathcal{V}_2$.

Note that we only have the number of views for the top-20 external links. Since

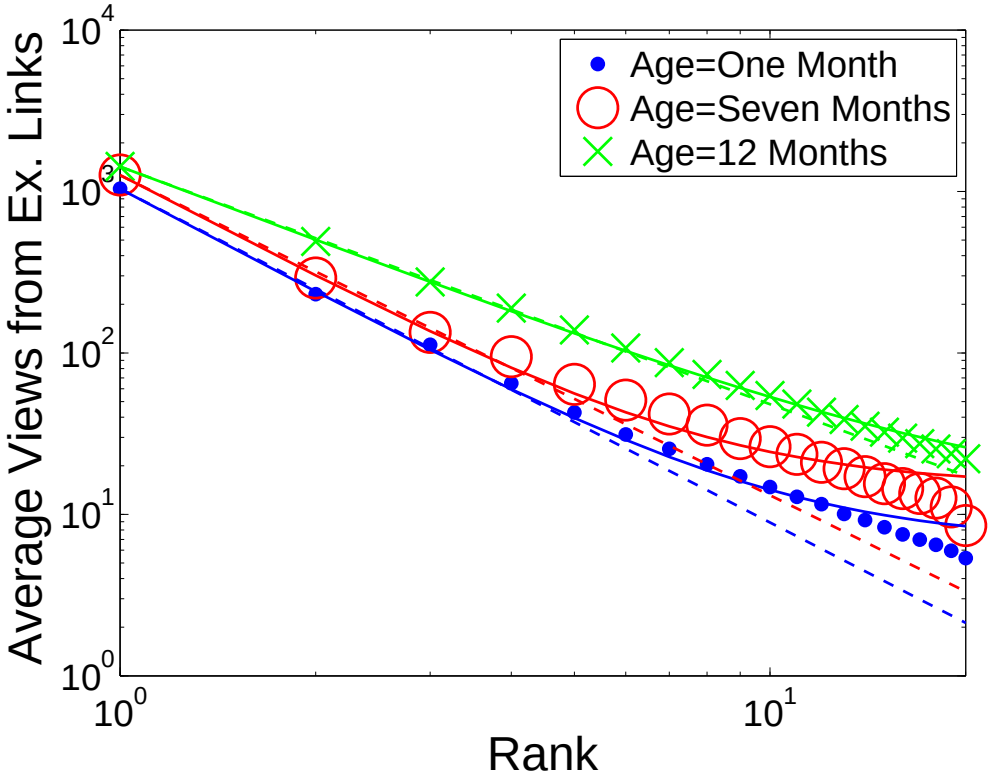


Figure 3.4. External views as a function of rank (number of links ≥ 20). The dashed lines shows the power law fit lines $\mathcal{V}_1(t) = a_1(t) \times r^{-p_1(t)}$, and the solid line shows the fits $\mathcal{V}_2(t) = a_2(t) \times r^{-p_2(t)} + \epsilon(t)$

we have seen a power law fit, we conjecture that the views of top-20 external links are representative for the views of all external links. As a result, we estimate the total views from all external links. This will make the conclusions of this study more grounded.

To estimate the total views from all external links, we divide the videos into two groups.

1. The videos with more than 20 external links.

2. the videos with less than 20 external links.

For Group 2, Youku provides the total views from the top-20 links of each video, or the views of the top-k links if the video only has k external links and $k \leq 20$. Let the

t (month)	$a_1(t)$	$p_1(t)$	r-square
1	82.98	2.087	0.9998
7	279.9	2.055	0.9999
12	619.3	1.515	0.9999

Table 3.2. The Parameters of Power Law Fits for Fig. 3.3

t (month)	$a_2(t)$	$p_2(t)$	$\epsilon(t)$	$r - square$
1	82.55	2.165	0.54	0.9998
7	277.4	2.089	3.06	0.9999
12	615.9	1.580	5.31	0.9999

Table 3.3. The Parameters of Power Law Fits with a Constant Deviation Terms for Fig. 3.3

total views from external links of Group 2 be V_2^*. For Group 1, we can further divide it into two parts: a) the total views from the top-20 external links and b) the total views from other (non-top-20) external links. Youku provides a). Let this be V_{20}. Thus, our primary target is to estimate part b) of Group 1.

In Fig. 3.4 we plot the total views of the external links as a function of the ranks of the external links in a log-log scale for the videos with more than 20 external links. We plot the data for the videos with an age of one month, seven months and 12 months. We see that the number of views also is close to the power law function. Again, to fit the power law function, we also let $\mathcal{V}_1(t)$ and $\mathcal{V}_2(t)$ be the total number of views for videos with age t months for the fits. We plot power law function $\mathcal{V}_1(t) = a_1(t) \times r_1^{-p_1(t)}$ and the power law function with deviation $\mathcal{V}_2(t) = a_2(t) \times r_2^{-p_2(t)} + \epsilon(t)$ for different age groups in Fig. 3.4 (Table 3.4 and Table 3.5 show $p_1(t), a_1(t), p_2(t), a_2(t)$, and $\epsilon(t)$, as well as the r-squares of the fits). We also see that the number of views from higher rank of external links is slightly

t (month)	$a_1(t)$	$p_1(t)$	r-square
1	1037	2.369	0.9978
7	1259	1.981	0.9985
12	1424	1.471	1.0

Table 3.4. The Parameters of Power Law Fits for Fig. 3.4

t (month)	$a_2(t)$	$p_2(t)$	$\epsilon(t)$	$r-square$
1	1031.6	2.1429	6.77	0.9998
7	1248	1.9203	14.95	0.9998
12	1416	1.529	11.49	1.0

Table 3.5. The Parameters of Power Law Fits with a Constant Deviation Terms for Fig. 3.4

larger than $\mathcal{V}_1$ but smaller than the fit $\mathcal{V}_2$.

Let $V_1^*(t)$ be the total views for part b) of Group 1, and let $\mathcal{N}(t)$ be the average number of external links of the videos aged t months, then

$$\sum_{i=21}^{\mathcal{N}(t)}(a_1(t) \times i^{-p_1(t)}) \leq V_1^*(t) \leq \sum_{i=21}^{\mathcal{N}(t)}(a_2(t) \times i^{-p_2(t)} + \epsilon) \tag{3.1}$$

Therefore, the estimated external link total views are $V_{20} + V_1^*(t) + V_2^*$. We plot both lower bound and upper bound of percentage of the external link total views contributing to the video total views in Fig. 3.5. We also plot the percentage of views from top-20 external links. Clearly, both the lower bound and the upper bound of the external link total views are greater than the views of the top-20 external links (which is only a section of the video total views). However, we can see that the views of the top-20 external links are very close to the estimated external link total views. Most of the time, the views of the top-20 external links

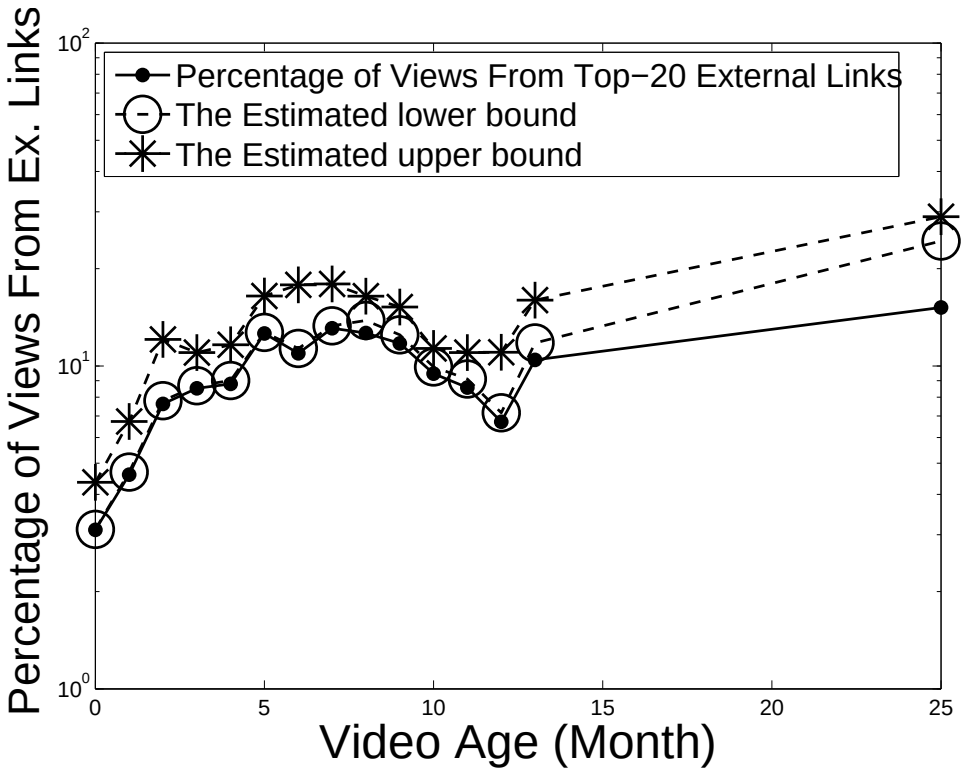
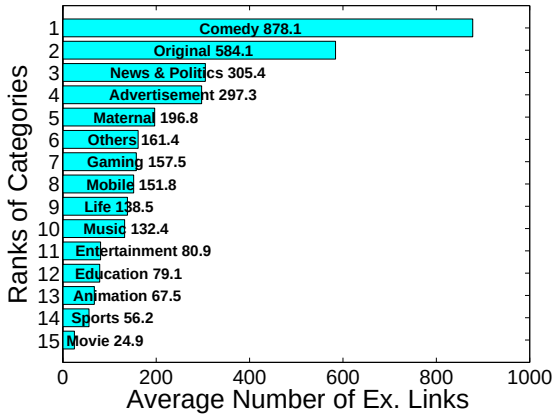


Figure 3.5. The estimated views from external links of the Youku videos.

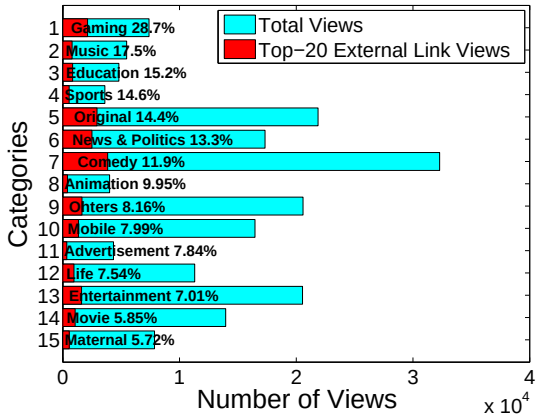
contribute to 90% of the total views from all external links. As such, we conclude that the total views from the top-20 external links are representative for our study.

3.3.4 The External Links from Different Video Categories

We next study from the point of view of the videos. We plot the number of external links of different video categories in Youku in Fig. 3.6 (a). In this figure, the categories are ranked by the average external links on each video. We see that the number of external links of videos is substantial. For example, for an average comedy video, there can be as many as 878.1 external links. Looking into the details of our log file, we see that many comedy videos are linked by a substantial number of different users in their blogs, usually by copying and referring of others' blogs. Some videos are linked in a great many pages in web forums. This actually suggests that external links can greatly increase the popularity of the videos.

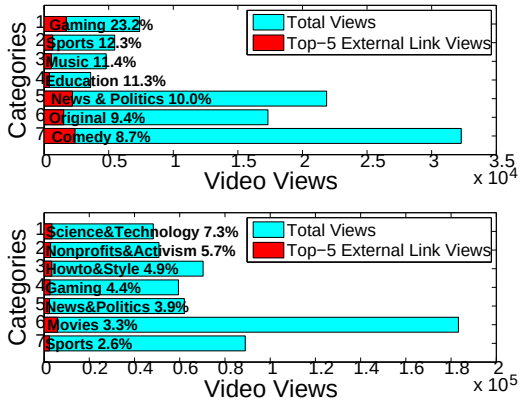


(a) The average number of external links per video



(b) Average views from top-20 external links per video

(Youku)



(c) Average views from top-5 external links per video

Figure 3.6. The contribution of external links on different categories of videos from both YouTube and Youku

In Fig. 3.6 (b), we select 15 categories in Youku, which have the highest percentage of external views. We plot the average total views of each category (in green) and the average views of the top-20 external links (in red). In the figure, “Comedy” attracts the largest number of external views, on average 3847.5 per video. This is not surprising as “Comedy” also attracts the largest number views (32286.2), representing the popularity of comedy videos in general. We also see that “Gaming” attracts 7391.4 views in total and 2125.0 views from external links, where the external views share the highest percentage. This suggests that as compared with other video categories in Youku, more percentage of views in the “gaming” category come outside Youku.

To compare Youku and YouTube, we show the views of different categories based on their respective views of the top-5 external links (see Fig. 3.6 (c)). In general, YouTube attracts an order of magnitude more views than Youku, but the percentage of external views is much smaller. This conforms to the observation in Fig. 3.3.1. Another observation is that the categories most viewed by external links are substantially different in YouTube and Youku. For example, they share in common only 3 out of 7 categories, namely “Gaming”, “Sports”, and “News & Politics”. This might show different tastes of the users throughout individual regions. In addition, the more success in Youku in extending its impact of the external links could suggest that there may be also potential for YouTube to increase its external views.

3.3.5 Summary

We summarize our major observations in this section as follows:

- the sheer number of external views and the external links are substantial for both YouTube and Youku. The external views/links have contributed greatly to Youku while it still remains small to YouTube;
- Most of the external links are linked to a small number of videos, i.e., the number of external links conforms to power law distribution; it fits especially well for the videos with large number of external links;

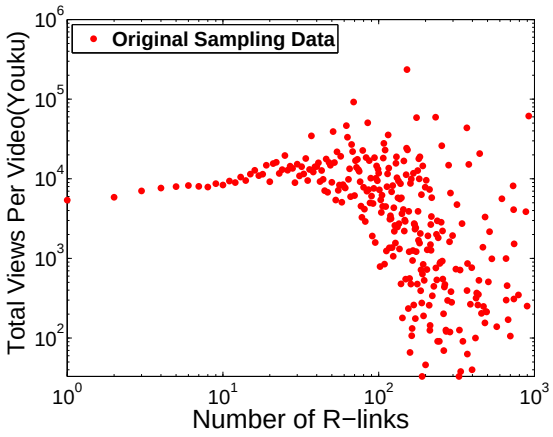
- The number of external views also conforms to power law; it fits better for old-age videos. Though we cannot obtain the total views from all external links, with the observation of power law, we can deduct that the views from top-20 external links are representative enough for videos; 4) Different video categories attract different percentages of external views. In some categories, e.g., “Gaming” in Youku, almost 30% of views are contributed by external links.

3.4 External Links vs. Internal Links

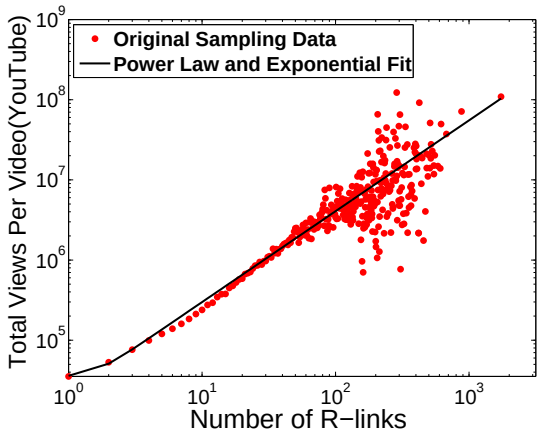
Since external links contribute to the video popularity, we analyze the factors that affect the number of external links. We study the relationship between internal links and external links which respectively represent the internal interactions and the external interactions. According to Alexa [103], a user spends an average of 22 minutes on YouTube and 6.7 minutes on Youku every day. As a result, we infer that many users would watch multiple videos in the video sharing sites. For these users, there are many ways to view multiple videos in the video sharing sites, and one of them is to follow the related video list (See Fig. 1.3). We call a video the *parent video* for the videos in its related video link list. We specifically focus on the relationship of the external links, the related video links (We call them the *R-links* thereafter) and the total views from parent videos (We call them *parent views* thereafter). In addition, we also study the relationship between the external links and other factor, such as the total views of videos.

3.4.1 Internal Parameters and External Links

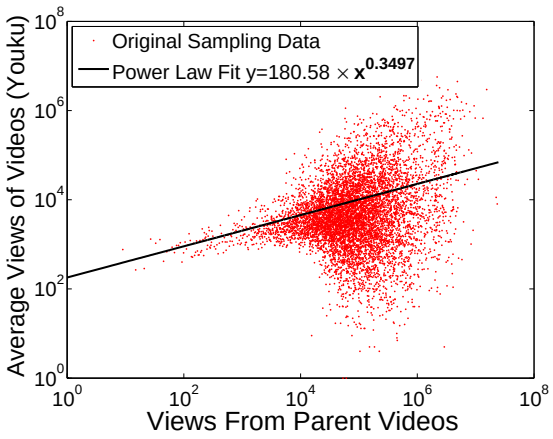
Fig. 3.7 presents the relationship of total views of a video with two internal factors, namely the number of R-links and parent views, in both YouTube and Youku. We plot in Fig. 3.7 (a) and (b) the number of total views as a function of the number of R-links of Youku and YouTube respectively. Here, we hardly see any impact of the R-links on the improvement of the total views in Youku, but we see that clear correlation exists for YouTube. This shows



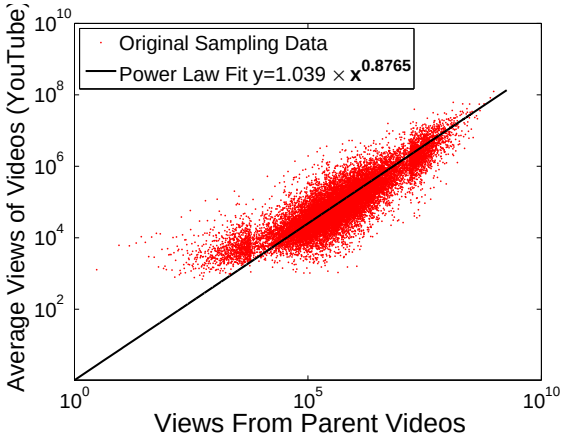
(a) The number of R-links as a function of total views (Youku)



(b) The number of R-links as a function of total views (YouTube)



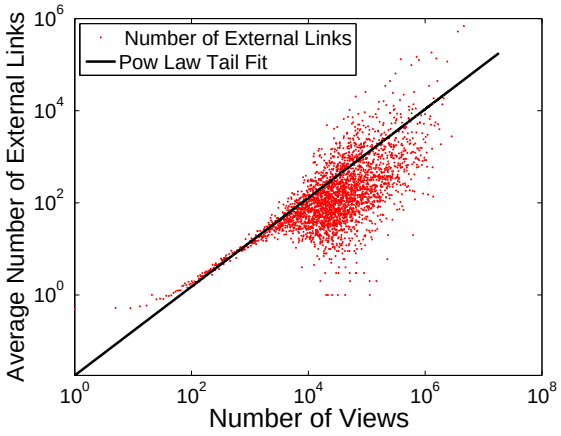
(c) Views of parent videos as a function of total views (Youku)



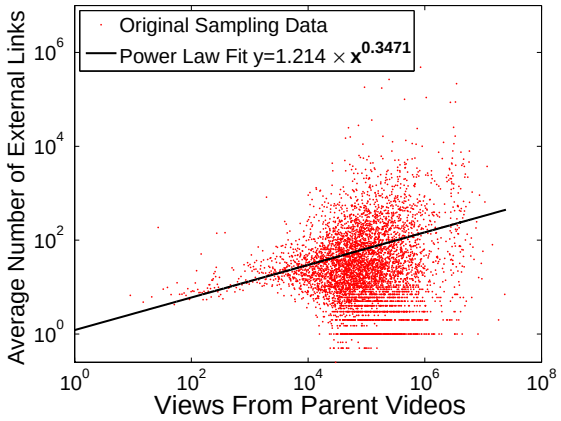
(d) Views of parent videos as a function of total views (YouTube)

Figure 3.7. The relationship between the video views, the number of external links, the views from parent videos, and the number of R-links

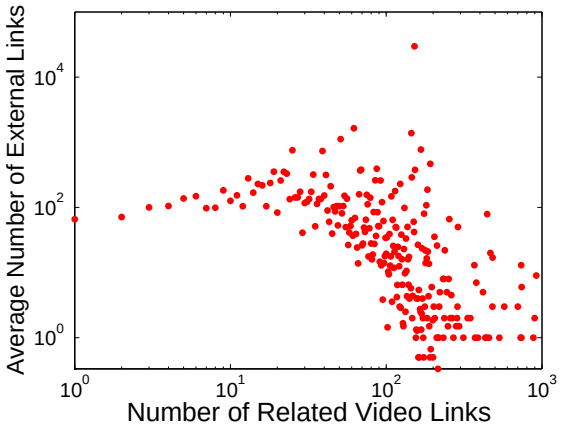
that more R-links lead to an increase of the views of YouTube. We plot in Fig. 3.7 (c) and (d) the average views of a video as a function of the views from the parent videos for Youku and YouTube respectively. We see that in general, the parent views have a positive impact on the video views for both Youku and YouTube, and the parent views of YouTube show an even stronger impact.



(a) Video views as a function of the number of ex. links



(b) Views of parent videos as a function of number of external links



(c) The number of R-links as a function of the number of ex. links

Figure 3.8. The relationship between the number of external links and various internal factors

Fig. 3.8 depicts the relationship between the number of external links and the three internal factors, namely the total views, the views from parent videos and the number of R-links. All the results in Fig. 3.8 are from Youku data set. Fig. 3.8 (a) shows a clear relationship between the total views and the number of external links. Especially when the number of views is over 100, we can see this relationship is almost linear. Fig. 3.8 (b) shows the views from the parent videos has weaker correlation with the number of external links. The number of external links scatters as the views of parent videos grows. In Fig. 3.8 (c), we see there is even weaker relationship between the number of R-links and the number of external links.

3.4.2 Analysis of the Correlation

$Corr$	views	Ex. links	R-links	parent views
views	1	0.506	-0.018	0.22
Ex. links	0.506	1	-0.029	0.20
R-links	-0.018	-0.029	1	0.23
parent views	0.22	0.20	0.23	1

Table 3.6. The correlation coefficient of parameters in Youku

$Corr$	views	R-links	parent views
views	1	0.49	0.77
R-links	0.49	1	0.60
parent views	0.77	0.60	1

Table 3.7. The correlation coefficient of parameters in YouTube

We next conduct analysis on the correlation coefficients[1] between the number of external links and internal factors for both Youku and YouTube (see Table 3.6 and Table 3.7). In Table 3.6 we find that the total views and the number of external links of Youku are most correlated with a correlation coefficient of 0.506. The number of R-links hardly affects the number of external links and the total views. The views from parent videos weakly correlated with all the factors. This conforms to the general intuition that the popularity of the video itself will directly impact on the number of external links (and vice versa). Also, we see that in Youku, the views of the parent videos have a more moderate impact and a larger related video links can hardly have an impact on the external links.

Compared with Youku, the connections of internal links in YouTube are much tighter. In details, we can see that the relationship among the number of R-links, the parent views and the total views are high. The correlations are 0.77 between video total views and parent views, 0.60 between the number of R-links and parent views, and 0.49 between the number of R-links and video total views respectively.

A possible explanation of the tighter correlation in YouTube is the stability of the related video list; that is, if the related videos change more frequently, it will affect the correlation between R-links and other factors. To verify this, we randomly chose 20,000 videos from the crawled video data set of both Youku and YouTube. We crawled/searched these videos from Youku and YouTube again on Nov. 28th 2011. We see that as compared to the 20,000 videos collected on Jul. 8th, 2009, 5376 Youku videos have been deleted and as compared to the 20,000 videos crawled on May 24th, 2009, 387 YouTube videos have been deleted. As the videos in YouTube are significantly more stable than Youku, we infer that there is less change in the related videos of YouTube too. Therefore, YouTube has a tighter correlation of internal links than Youku.

[1]Here, a correlation coefficient of 1 indicates that the two parameters are linearly correlated, i.e., one parameter will increase (or decrease) linearly with the other parameter. A correlation coefficient of -1 indicates that one parameter will increase linearly as the other parameter decreases.

3.4.3 Summary

We summarize our major observations in this subsection as follows:

- The number of external links is mostly and more directly affected by the total views of the videos;
- The number of external links can be affected indirectly by such internal factors, such as parent views and the number of R-links, since these factors can increase the views of videos;
- The internal factors in YouTube have a stronger correlation than that of Youku. This may be because the Youku videos are less stable as there is a higher deletion rate of the videos.

3.5 External Links on Videos in Different Age Groups

To further understand the impact of external links, as well as the correlation of external links with various internal factors, in this section, we investigate the characteristics of external links in different video age groups. Since we did not trace specific external links or trace the external links for specific videos, we group the videos according to different ages. Our study is then on the characteristics of external links on younger videos and older videos. We believe this provides a macro view of the evolution of the external links on videos.

3.5.1 The External Links on Videos of Different Age Group

We first compare the percentage of videos received external links in YouTube with that in Youku. We only focus on the videos that have five or more external links.

From Fig. 3.9 we can see that videos in YouTube are more linked by external links than videos in Youku. For example, it is observed that for YouTube, 90% of the five-month-old videos have at least five external links. Looking from another angle, we can say that

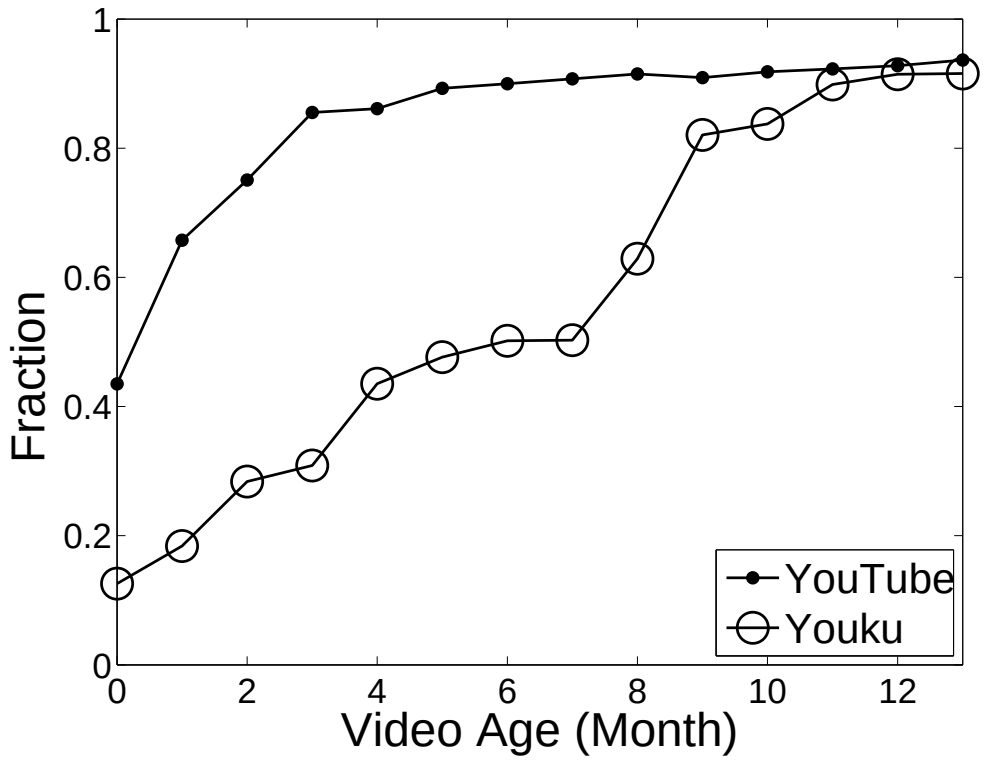


Figure 3.9. The percentage of the videos with more than five external links.

93.7% of the videos are with at least five external links after 15 months. On the contrary, in Youku, only about 50% of the five months old videos have at least five external links. Nevertheless, the percentage rises eventually and from the data we collected, we can see that after a video that is one year old, more than 90% of the chance (both for YouTube and Youku) it will have more than five external links. The differences between YouTube and Youku may be accounted as follows. First, compared with Youku, YouTube is a world-wide video sharing site and the videos in YouTube have a larger audience base. Therefore, the YouTube videos gain popularity more quickly. This has been proved in Fig. 3.3.1 (b) where we see the YouTube videos have a larger number of average views in each age group. Second, as we have shown in Section 3.4.2, the number of external links for each video has a positive relationship with the video total views. Therefore, as the YouTube videos gain popularity (being viewed) more quickly, the YouTube videos also get external links faster than Youku videos.

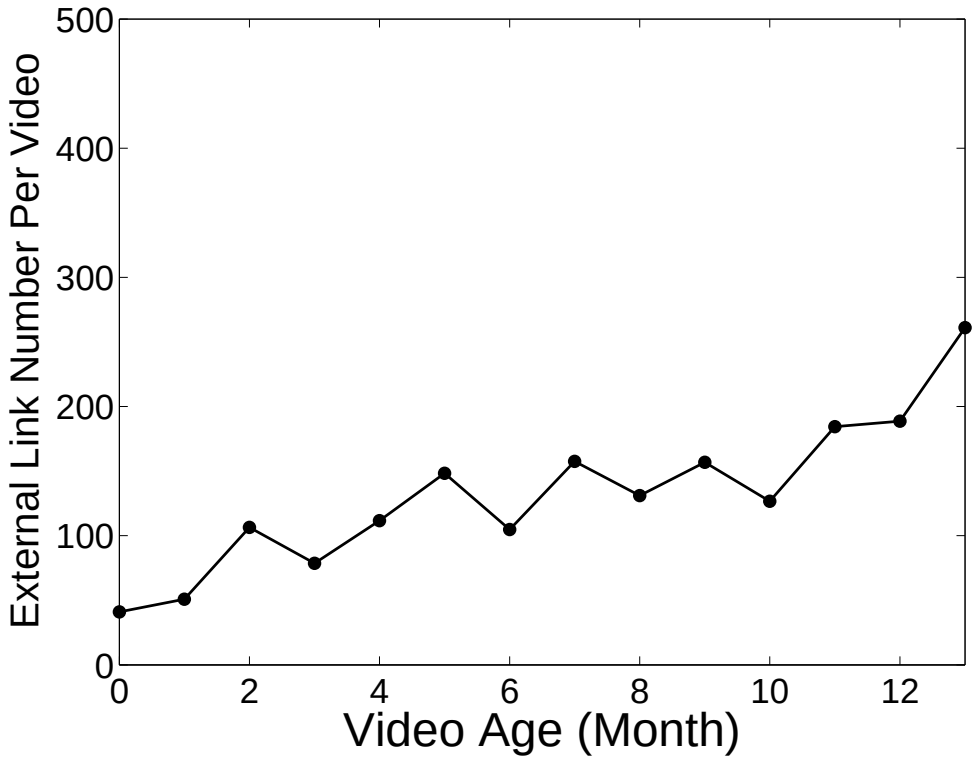


Figure 3.10. The average number of external links for each video of different video ages, Youku.

We then study the average number of external links for each video of different video ages. As YouTube can only provide the top-5 external links for each video, we thus focus on Youku data. The results are in Fig. 3.10. We can see that the average number of external links is increasing. This is not surprising that the total views of videos are increasing with video ages (as it is shown in Fig. 3.3.1 (b)), and there is a positive relationship between the video total views and the number of external links (as it is shown in Section 3.4.2). As well, the average views from external links for each video of different video ages are also increasing, as shown in Fig. 3.11.

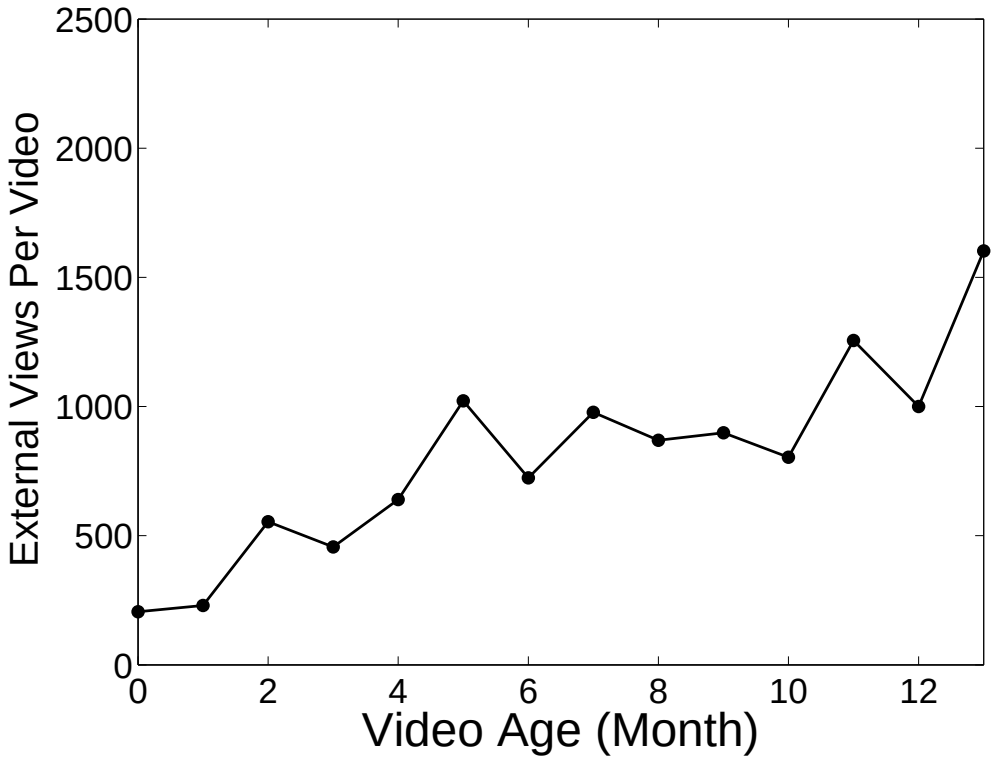


Figure 3.11. The average views from external links for each video of different video ages, Youku.

3.5.2 The Correlation between External links and Video Total Views in Different Video Age Groups

We study the correlation of the number of external links and the video total views according to different age groups. From Section 3.4, we see that there is a positive correlation between the number of external links and the total views (including both external views and internal views).

From Fig. 3.12, we see that the correlation becomes stronger for the videos in older age groups. The correlation coefficient rises from 0.42 for the video group of one month old to about 0.85 for the video group of one year old. This indicates that the relationship between the number of external links and the video total views is strengthened with time. We conjecture that there is positive impact from both sides, i.e., 1) the number of external links and external views increase, contributing to the video total views, and 2) the video total

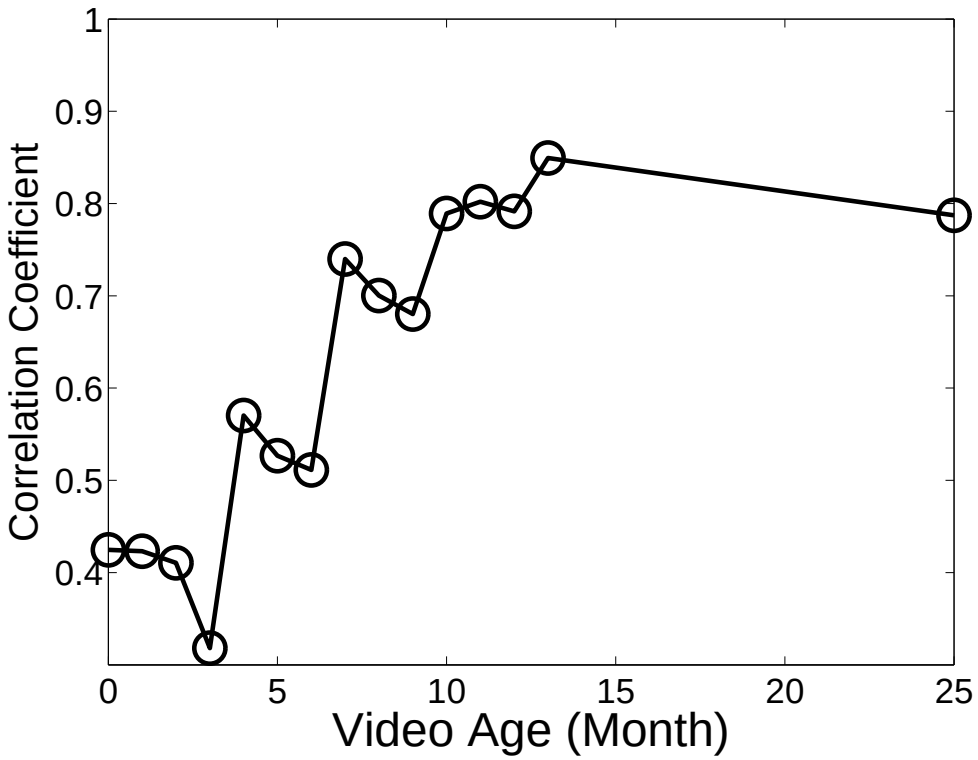


Figure 3.12. The correlation of external link number and video total views with the video ages

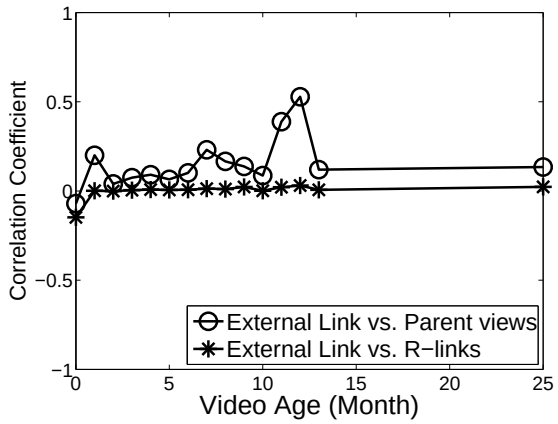
views (and thus the popularity) increase, contributing to the increase of the external links.

3.5.3 The Correlation Coefficients in Different Video Age Groups

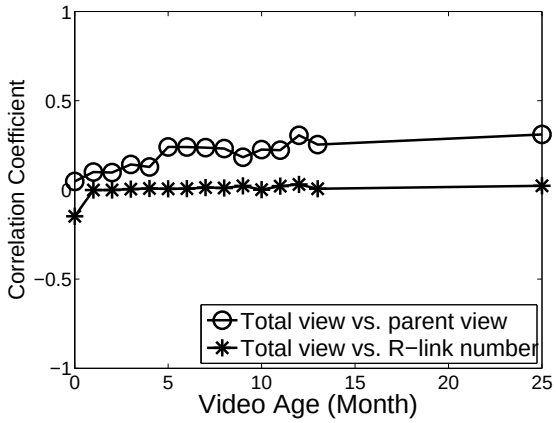
In Fig. 3.13, we show the correlation coefficient among external links, video total views, parent views, and the number of R-links in YouTube and Youku.

Fig. 3.13 (a) shows the correlation coefficient of the number of external links with the parent views and the number of R-links in Youku. We can see that the correlation coefficient between the number of external links and the parent views fluctuates sometimes but remains above 22% after the first month. However, the correlation coefficient between the number of external links and R-links remains zero. This conforms to the observations in Section 3.4.2.

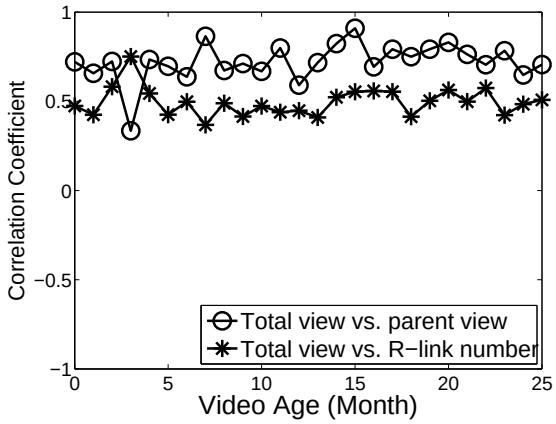
Fig. 3.13 (b) shows the correlation of video total views with parent views and the



(a) The corr. of external links with other factors (Youku)



(b) The corr. of video total views with other factors (Youku)



(c) The corr. of video total views with other factors (YouTube)

Figure 3.13. The evolution of correlation of external links and other internal factors with time

number of R-links in Youku in different video age groups. Not surprisingly, we can see that in Youku the number of R-links do not contribute to video total views in any video age group. However, similar with the relationship between the number of external links and the video total views, we can see the correlation between the video total views and parent views is also strengthened as videos get older, from 0% to about 40%. As the correlation between the number of external links and video total views are strengthened with the video age, and the relationship between video total views and parent views is also positive, we infer the following: for Youku, if a video can gain more parent views, the total views and the number of external links of the video may be increased after a period of time.

Fig. 3.13 (c) plots the correlation of video total views with parent views and the number of R-links in YouTube in different video age groups as a comparison. Here, we can see that these two correlation coefficients maintain stable and high. As compared to Youku, we can still see that the correlation of internal links in YouTube is significantly larger. This also conforms to the results in Section 3.4.2.

3.5.4 Summary

As a summary, as the videos get older, the number of external links of each video is increasing as well as the number of external views of each video. This indicates the increase of the number of external links is not restricted in certain ages of videos but for all videos. The correlation between the number of external links and the video total views is strengthened with the video ages. As we can also see a positive correlation coefficient between video total views and parent views, we conjecture that if a video can obtain more parent views, the number of external links may be increased.

3.6 Conclusion

In this chapter, we studied in detail an important aspect of video sharing sites: the external links. The external links provide a unique way for the video sharing sites to accelerate the

distribution of the videos. We observed that the external links can play a non-trivial role both in terms of the number of external links on a video, and the number of views contributed to the video. We also observed that the external links have quite different impacts on YouTube and Youku. We studied the external links for different video categories. We also discussed the correlations of the external links and the internal related video links. We showed that the number of internal related video links have less impact on the external links than the total views of the video. We also study the characteristics of external links in different video age groups. We see that videos are possible to get external links and external views in all age groups. We believe that our work can provide the foundation for the video sharing sites to make more targeted advertisement, customized user development, etc.

In this work, we study external links as an example to show how we estimate specific functions or features increase the application popularity. This provides important accordance for the deployment if we are going to migrate this video sharing site to the cloud.

As a first work on the external interactions of video sharing sites, we concentrate on some fundamental problems, such as how the data of external links can be collected, whether the data collection on top external links can provide a good approximation for the overall picture, and some basic aspects of the external links. There are problems yet to be answered. Especially, we are interested in more detailed analysis of the different impacts of external links on Youku and YouTube.

CHAPTER 4

KALMAN FILTER-BASED SCHEDULING FOR STABLE DIFFERENTIATED SERVICE IN CLOUD SERVERS

4.1 Overview

In this chapter, we propose a Kalman filter-based workload management scheme for the cloud servers to maintain a stable service differentiation under the fluctuating user traffic. The previous chapter shows that the user traffic of the Internet applications can be highly dynamic since it is affected by various factors. For example, the user traffic of the video sharing sites is raised by external links and related video links. Therefore, for the Internet applications in the highly dynamic user traffic, devising a stable workload mange scheme can be a key issue to maintain the cloud system in a high performance.

In detail, we concentrate on the Service Level Agreement (SLA) in the cloud servers and we aim to minimize the cost from the violation of SLA. We study the predicted-based workload management scheme, which is widely adopted. We firstly scrutinize the root cause of the predicted-based scheme being instable, and then propose a Kalman-filter based workload management scheme as a solution. The actually deployed experiment shows that the proposed scheme is able to maintain the system stable in the fluctuating user traffic.

The organization of this chapter is as follows. Section 4.2 shows the root cause of the system being instable, from both the experimental and mathematically modeling approaches. Section 4.3 suggests a Kalman filter-based scheme as a solution. Section 4.4 presents the experiment and the performance evaluation. We finally conclude this chapter in Section 4.5.

4.2 System Architecture and Service Level Agreement Stability

4.2.1 The Objective of the SLA system

The major objective of our SLA system is to minimize the SLA violation loss. Here we let this loss be a utility as a function of the response time. Clearly, the response time is a function of service rate and request arrival rate. Let D_k be the utility function of the service class k, T_k be the response time for service class k, and μ_k be the service rate for service class k, where $k \in \{gold, silver, bronze\}$. We use abbreviation $\{g, s, b\}$ for $\{gold, silver, bronze\}$ respectively. Suppose that the total available service rate is μ, our system goal is to minimize the utility, which is shown as Eq. 4.1.

$$\begin{array}{l} min_{\mu_k} \sum_{k \in S} D_k(T_k(\mu_k)) \\ s.t. \\ \sum_{k \in S} \mu_k \leq \mu \end{array} \tag{4.1}$$

Without the loss of generality, we let $S = \{gold,\ silver,\ bronze\}$ during the discussion. We use abbreviations $\{g, s, b\}$ to refer these different service classes respectively. Intuitively, the violation loss function faces higher loss for higher response time, and higher loss for gold than that of silver than that of bronze. Formally, we have

- For $t \leq d_k$, $D_g(t) = D_s(t) = D_b(t)$ and they are equal a small constant. Since the response time control target is not violated, and there is small;
- For a given response time $d_k < t < \infty$, $D_g(t) > D_s(t) > D_b(t)$. This refers to that prioritized service classes, such as the gold, require better service quality;
- $D_k(T_k)$ is monotonically increasing with higher response time T_k; and 4) $D_k(\mu_k)$ is a convex function. This is because nonconvex increasing functions, e.g., logarithmic, cannot well-capture the real-world relationships between the violation loss and response time [32]. Therefore, we have the result as Eq. 4.2.

$$\begin{array}{l} \frac{\delta D_k}{\delta \mu_k} < 0 \\ \frac{\delta^2 D_k}{\delta \mu_k^2} > 0 \end{array} \tag{4.2}$$

4.2.2 System Architecture

The users generate tasks to be executed in the servers. We call the independent tasks *sessions*. In each session, there are correlated *requests* to achieve a specific task. The requests are correlated in the sense that the current request will be initiated only after the previous request is served. We call it the *inter-arrival time* for the duration between two consecutive requests. As it is studied in [28], the inter-arrival time has two parts. One is the response time of the previous request, and the other is *think time*, which is the duration for performing some actions after receiving the response and before initiating next request. A detailed system architecture is shown in Fig. 4.1.

The requests are queued in the proxy according to their SLAs. A dispatch algorithm collect requests from the queues and forward them to application servers according to the assigned service rates achieved from the SLA maintenance module. Normally, the assigned service rates from the SLA maintenance module assures the response time of different service classes to minimize the SLA violation costs.

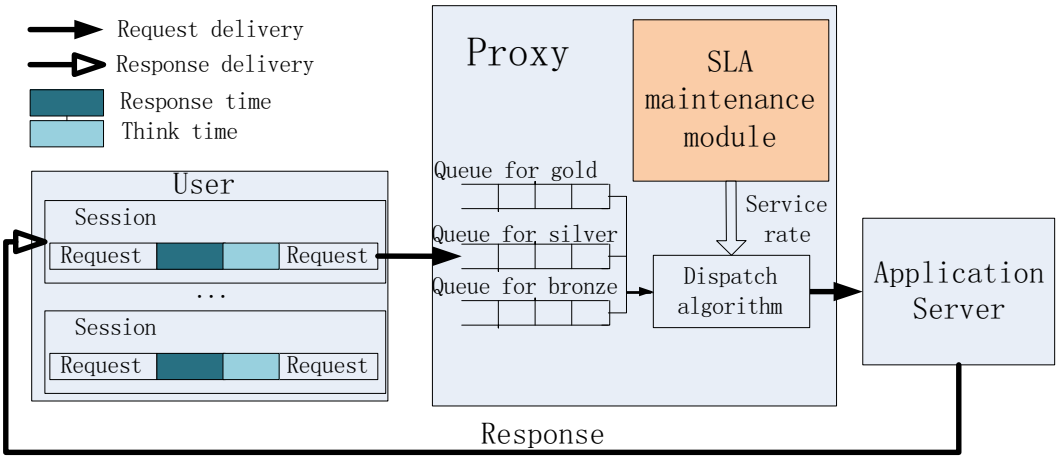


Figure 4.1. A detailed architecture for cloud server environment

To design the SLA maintenance module, however, having a differentiated service rate for each service level does not necessarily mean the SLA can work correctly. We see a simple example in Fig. 4.2. Here, the service rate for gold request is 200ms and the service

rate for silver request is 300ms. Nevertheless, if gold requests come every 100ms, and the silver requests come every 300ms, we see that there is no queueing delay for silver requests and there are additional delay (an average of 200ms if five requests in roll) for gold requests.

From this example, we can see that the request arrival rate can be an important factor for the services. The request arrival rate, however, is not solely determined by the system. It is thus necessary to conduct prediction of the request arrival rate.

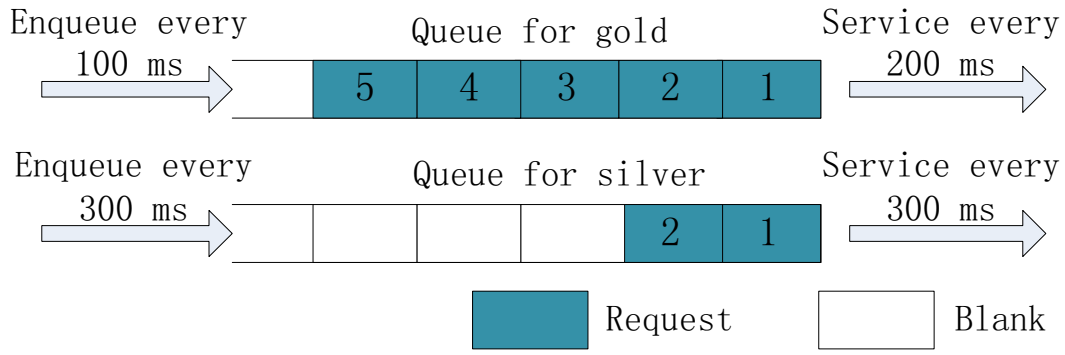


Figure 4.2. An example of service differentiation

The SLA maintenance module runs periodically, and we define this period as a *control cycle*. In the end of each control cycle, SLA maintenance module generates service rates according to Eq. 4.1 for all service classes in the next control cycle. At the same time, the proxy estimates the *estimated response time* according to the assigned service rates. Given that the request arrival rate in the next control cycle is predicted and the service rate has been assigned, its response time can be estimated by queueing model, such as MM1 or the Generalized Processor Sharing (GPS) model [21]. The estimated response time is an important criterion to assess whether the service rate allocation is successful. If the estimated response time is close to the actual response time, it means that the request arrival rate is accurately predicted and the service rate allocation successfully achieves the minimization of the SLA violation loss.

We will develop an SLA maintenance module. It is designed to output one parameter, the service rate; as such it minimally affects other modules. Before presenting our

SLA maintenance module, we first systematically study the intrinsic reasons of the system instability.

4.2.3 System Instability: an Experimental Approach

From the general architecture of the servers, we can see that the minimization of the utility relies on the accuracy of request arrival rate prediction. An accurate request arrival rate prediction assures the successful control of request response time, and hence to minimization of the utility.

However, we should also notice the **close loop** between the request arrival rate and the response on the system stability. Firstly, the predicted request arrival rate affects the response time. This is because that the assigned service rate increases as a larger request arrival rate is predicted, and vice versa. Secondly, the response time has impacts on the predicted request arrival rate in return. This is because the request arrival rate of each session is determined by the request inter-arrival time, which is comprised by the think time and the response time. Since the response time is affected, the actual request arrival rate is also affected. As the predicted request arrival comes from the past actual request arrival rates, the prediction is hence impacted.

Since the close loop exists, a question arises that whether the response time prediction error can reduce to zero if an error happens in request arrival rate prediction. This situation would become more complex in the multi service class scenario.

We conduct an experiment on IBM's cloud SLA system. We use one proxy and one physical server with Intel Xeon E5405 CPUs, 4GB RAM and 7200 RPM hard disk. We deploy the IBM framework as discussed in Section II. We use the IBM benchmark user request generator [99] to generate user requests. For illustration simplicity, we only use two service classes: *gold*, and *bronze*. We apply the state-of-the-art prediction technique which, however, assumes the request arrival rates in two continuous control cycles are identical (i.e., they overlook that the user requests are correlated) [99]. We have tried other prediction techniques (e.g., AR(1) [21]) and observe the same problem. Intrinsically, these prediction

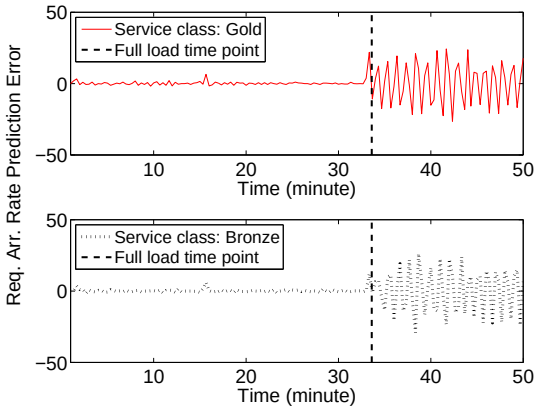
techniques focus on the prediction accuracy, not addressing the closed-loop problem which we will show shortly.

Duration	Average session number	Average think time
about 15 min.	1	500 ms.
about 10 min.	1	400 ms.
about 10 min.	1	100 ms.
about 15 min.	5	100 ms.

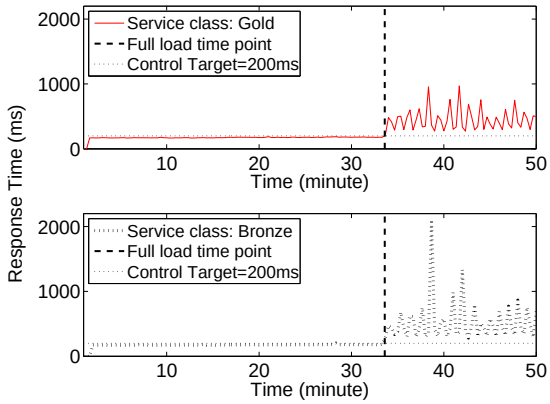
We set the control target of the response time to 200ms and run the experiment for 50 minutes and the results are shown in Fig. 4.3. In particular, we divide this 50-minute experiment into three stages. In each stage, we increase the request arrival rate by adding more sessions. The details are as follows: in stage #1 (0 to 15 minutes), the request arrival rate is 5 requests per second for all service classes; in stage #2 (15 to 33 minutes) the request arrival rate is 12 requests per second. Note that the system is underload in these two stages. State #3 (33 to 50 minutes) is a full loaded stage where the request arrival rate is 40 requests per second.

As shown in Fig. 4.3(a), we can see prediction errors while we increase the workloads in the system. In the first two stages (before 33 minutes), these errors will not change the request arrival rate in Fig. 4.3(b). This is because the response time is still under our control target in Fig. 4.3(c). However, when the response time exceeds the control target in stage 3 (after 33 minutes), the request arrival rate in Fig. 4.3 will be largely fluctuated by the prediction error. This biased arrival rate unavoidably introduce more prediction errors and further affects the request response time. It is also worth mention that the system stuck in this instable stage and fail to recover even after 50 minutes.

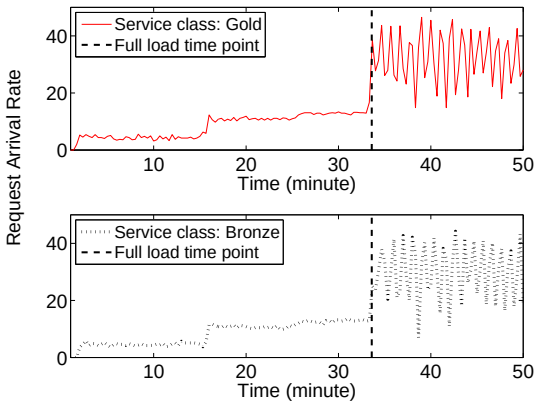
As such, we give a formal definition on the system stability. We let the *estimated response time* of the ith control cycle be t_i', and the *actual response time* be of the ith control cycle t_i. Also, we let Δ_i be the *response time estimation error* and $\Delta_i = |t_i - t_i'|$. Therefore,



(a) Request arrival rate prediction error



(b) Response time



(c) Request arrival rate

Figure 4.3. Instability problem in SLA systems

we define the stability of the response time estimation error according to Lyapunov stability and it is as Definition 1.

Definition 4.2.1. *We define our SLA system is* Lyapunov stable*, if it fulfil the following requirement: if for every* $\Gamma > 0$ *, there exists a* $\delta = \delta(\Gamma) > 0$ *such that, if* $|\Delta_0 - 0| < \delta$ *, then* $|\Delta_i - 0| < \Gamma$*, for every* $i > 0$*.*

In Definition 4.2.1, we stress on the stability of the response time estimation error Δ_i. This definition aims to restrict the response time estimation error within a scope, and intuitively this scope should be as small as possible. As definition 1 focus on the long-term stability, we also define the *bounded error* to depict the response time prediction error in a specific control cycle.

Definition 4.2.2. *We call a service has* ϵ-bounded error *if the following holds: if* $\frac{|t_i - t_i'|}{t_i} = \epsilon$*, we say that the* i*th control cycle has an* ϵ-bounded error.

Hereafter, we call the problem as the *instability problem*, if the response time prediction error Δ_i cannot reduce to zero.

This definition means the *estimated response time* should be as close to the actual ones as possible. Such estimation has its limitations: the estimation stability depends on the accuracy of request arrival rate prediction. For example, the request arrival rate λ_{k+1} of the upcoming control cycle is unknown. Therefore, they have only be predicted in advance. Here, we let λ_{k_i} be the actual request arrival rate for service class k in the ith control cycle, and $\tilde{\lambda}_{k\_i+1}$ be the predicted request arrival rate for service class k in the ith control cycle. The current existed solutions for the prediction are such as $\tilde{\lambda}_{k\_i+1} = \lambda_{k\_i}$ [30], or AR(1) model [21]. Nonetheless, these studies only focus on the accuracy of the prediction, but neglect the close loop between the request arrival rate and the service time. As a result, the instability problem may happen, which we are going to model in the next subsection.

Notation	Definition
K	The set of service class. In this chapter, $K = \{Gold, Silver, Bronze\}$
λ_{k_i}	The actual request arrival rate for service class k in the ith control cycle.
$\tilde{\lambda}_{k\_i}$	The predicted request arrival rate for service class k in the ith control cycle.
L_k	The actual request arrival rate in next control cycle for service class k
ψ_k and ϕ_k	The request arrival rate prediction error in two continuous control cycles for service class k
l_k and L_k	The actual request arrival rate in two continuous control cycles for service class k
μ_k and U_k	The service rate in two continuous control cycles for service class k
D_k	The utility value for service class k, $k \in K$.

Table 4.1. The notation table for the improved proxy algorithm

4.2.4 Modeling the System Instability

To model the system instability, we mathematically examine the close loop between the request arrival rate prediction error and the response time. We use K to denote the set of the service classes and $k \in K$. We define λ_{k_i} as the actual request arrival rate for the

service class k in the ith control cycle, and $\tilde{\lambda}_{k\_i}$ as the predicted request arrival rate for the service class k in the ith control cycle. Also we define the prediction error ϕ as Eq. 4.3. The prediction error ϕ can be positive or negative. $\phi > 0$ indicates that the request arrival rate is overestimated while $\phi < 0$ means it is underestimated. Also, we define ϕ_k and ψ_k as the request arrival rate prediction bias of service class k in the two continuous control cycle, and ψ_k is after ϕ_k. We also define l_k and L_k are the actual request arrival rate for the corresponding continuous control cycle of service class k, and L_k happens after l_k. We also define U_k and μ_k are the service rate for the corresponding continuous control cycle of service class k, and U_k happens after μ_k. The utility value of the service class k is denoted as D_k.

$$\phi = \text{predicted request arr. rate} - \text{actual request arr. rate} \tag{4.3}$$

In order to model the system instability, we study the relationship between the request arrival rate prediction errors of two consecutive control cycles. Assuming our SLA system has n service classes, let the request arrival rate prediction error vector of the current control cycle be $\phi = (\phi_1, \phi_2, ..., \phi_n)$, the prediction error vector in the next cycle be $\psi = (\psi_1, \psi_2, ..., \psi_n)$, the actual request arrival rate in current control cycle be $l = (l_1, l_2, ...l_n)$, that in the next control cycle be $L = (L_1, L_2, ...L_n)$. We have the following theorem.

Theorem 4.2.1. *If user behaviors stay constant (the session number and the user think time do not change), there is an error threshold ϵ existed for request arrival rate prediction, and if the request arrival rate is underestimated and $|\phi| > \epsilon$, the prediction error will not converge to zero.*

Proof. We formulate the prediction errors of two consecutive cycles as $\psi = A\phi$, where we use A to denote the transition matrix of the two continuous control cycles. For the simplification of the analysis, we assume that prediction algorithm is $\tilde{\lambda}_{k\_i+1} = \lambda_{k\_i}$. As a result, we have $A = \frac{\delta\psi}{\delta\phi} = \frac{\delta(L-l)}{\delta\phi} = \frac{\delta L}{\delta\phi} - \frac{\delta l}{\delta\phi} = \frac{\delta L}{\delta l}\frac{\delta l}{\delta\phi} - \frac{\delta l}{\delta\phi} = \frac{\delta l}{\delta\phi}(\frac{\delta L}{\delta l} - 1)$.

According to Lyapunov stability theory, the sufficient and necessary condition of stability is that none of the eigen values of matrix A have positive real parts. Let E_i be the ith eigen value of matrix A, and a_{ii} be the diagonal element of A. According to the property of matrix, $\sum_{i=1}^{n} E_i = \sum_{i=1}^{n} a_{ii}$. To ensure all the eigen values have negative real parts, one necessary condition is that $\sum_{i=1}^{n} a_{ii} < 0$. As a result, the instability prerequisite is $\sum_{i=1}^{n} \frac{\delta l_k}{\delta \phi_k}(\frac{\delta L_k}{\delta l_k} - 1) > 0$. After analysis, we have:

- $\frac{\delta l_k}{\delta \phi_k} > 0$ and $\frac{\delta L_k}{\delta \psi_k} > 0$: according to the **three criteria** in Section 4.2.1, we can easily prove that the more the request arrival rate is predicted, the more the service rate is allocated for this service class. Therefore, the corresponded service class has a shorter response time and a shorter request inter-arrival time. This service class hence has a larger request arrival rate.

- $\frac{\delta^2 L_k}{\delta l_k \delta \phi_k} < 0$: $\frac{\delta^2 L_k}{\delta l_k^2} = \frac{\delta^2 U_k}{\delta l_k^2}\frac{\delta^2 T_k}{\delta U_k^2}\frac{\delta^2 L_k}{\delta T_k^2}$. According to the MM1 queueing model or the GPS model from [21], we can easily have $\frac{\delta^2 T_k}{\delta U_k^2} < 0$ and $\frac{\delta^2 U_k}{\delta l_k^2} < 0$ with the water filling theory. As the next cycle request arrival rate L_k is opposite proportional to the request response time T_k, we also have $\frac{\delta^2 L_k}{\delta T_k^2} < 0$. Therefore, we have $\frac{\delta^2 L_k}{\delta l_k^2} < 0$. As $\frac{\delta l_k}{\delta \phi_k} > 0$, we have $\frac{\delta^2 L_k}{\delta l_k \delta \phi_k} < 0$.

As a result, the $\frac{\delta l_k}{\delta \phi_k} > 0$, the instability prerequisite is reduced to $\sum_{i=1}^{n}(\frac{\delta L_k}{\delta l_k} - 1) > 0$ or $\sum_{i=1}^{n} \frac{\delta L_k}{\delta l_k} > n$.

Therefore, if user behavior is active enough (this means the average session number is large or the think time is small enough) to assure the upper bound of $\sum_{i=1}^{n} \frac{\delta l_k}{\delta \phi_k} > n$, there is an underestimation threshold ϵ existed. If the request arrival rates are underestimated and $|\phi| > \epsilon$, the instability occurs. □

From the close loop between the predicted request arrival rate and the response time, this theorem proves the condition that the response time estimation error do not converge to zero, or in other word, the condition of the system instability. It shows that while the request arrival rates of all service classes are underestimated to one specific extends, the close loop between the request arrival rate and the response time will never bring the response time prediction error to zero. A typical instability case may be as follows: for example, a piece of explosive news happens, and the increase of the request arrival rates is larger than the threshold ϵ, the instability occurs. This phenomenon is common in our daily life, and drives us to improve a stability-aware proxy.

4.2.5 Solution Space

Since the system instability comes from the close loop between the predicted request arrival rate and the request response time, we use the controllers as a solution to keep the system stable.

We consider the solution space of the Proportional Integral Derivative (PID) [92] controllers and the Kalman filter. However, we choose the Kalman filter as the solution for our problem for the consideration of simplifying the design. In our problem, the system model is the optimization problem in Eq. 4.1. However, the PID controller requires linearization of the system model, the controller gains, poles and zeros to do the control, and they are complicated in our problem. Compared with PID controllers, Kalman filter is a kind of predictors that provide the estimation (the predicted request arrival rate in our case) as parts of the strategy to control the system. It takes the system model into consideration, and this would greatly lower the design complexity. Therefore, we use Kalman filter as our solution.

There are many types of Kalman filters for us to choose from, e.g. Kalman filter, extended Kalman Filter (EKF), Iterative Extended Kalman Filter (IEKF), and Unscented Kalman Filters (UKF). However, as we know the traditional Kalman filter is for linear systems. As it is shown in Section 4.3.4, the system function for our problem is not linear, traditional Kalman filter is not the suitable one for one problem. The other three Kalman

filters, EKF, IEKF, and UKF are all for non-linear systems but perform distinct in different scenarios. As it is studied in [3], the EKF does not outperforms IEKF and UKF because EKF leverages Jacobian matrix to linearly estimate the non-linear transformation between the state variables and the measurement variables, and transformation errors are introduced. IEKF outperforms EKF because it runs the transformation iteratively to reduces the errors. UKF uses unscented transformation, which is a non-linear transformation, to increase the accuracy. Nevertheless, according to the simulation results in [64] [65], UKF is not the best filter algorithm while the process and measurement noises are not Gaussian. As in our scenario, the noises come from the close loop, and these noises are obviously not Gaussian. As a result, we choose IEKF as the solution for our instability problem.

4.3 The Stability-aware Proxy Model

To provide a stable proxy, it might be possible to reconstruct the SLA system architecture entirely. This is beyond the scope of the study in this chapter. We choose to develop an SLA maintenance module, where we output a single parameter, the service rate. The dispatch algorithm, the control point for application server load balancing, uses this service rate to the servers. Such design is easy to get integrated and has a minimal impact on system modification. We detail our design in this section.

4.3.1 The Design Objective

In order to improve the stability of the workload management system, we are going to provide a stability-aware proxy design. Basically, the stability-aware proxy design should fulfill the follows.

- **Optimization.** The workload management in the proxy module should minimize the total utility value most of the time.
- **Stability.** The proxy should be able to control the close loop between predicted request

arrival rate, and keep the response time the response time estimation error as small as possible.

- **Scalability.** The system should not only satisfy three service class scenarios, but also can be easily applied in the systems with any number of service classes.

4.3.2 Kalman Filter in SLA Scenario

As it is discussed in III.E, we use IEKF to solve the instable problem. Note that, the instability problems comes from request arrival rate prediction error and the close loop between the request arrival rate and the response time. IEKF solve the instability problem by offering the request arrival rate prediction as a strategy to keep the system stable.

A general Kalman filter is as Eq. 4.4. Here, we use $X^{(i)}$ as state variable vectors in the ith control cycle, which contains the parameters to be predicted, and $Z^{(i)}$ as measurement variable vector in the ith control cycle, which encapsulates parameters that are measured and help calibrating the state variables. Let $f(\cdot)$ be the function between the current state variables and the previous state variables, and $h(\cdot)$ denotes the function between measurement variables and the state variables. At last, ω and v are respectively the process and observation noises which are both assumed to be zero mean multivariate with covariance Q and R.

$$\begin{cases} X^{(i)} = f(X^{(i-1)}) + \omega & \omega \sim N(0, Q) \\ Z^{(i)} = h(X^{(i)}) + v & v \sim N(0, R) \end{cases} \tag{4.4}$$

Here, we use $\hat{X}^{(i)}$ and $\hat{Z}^{(i)}$ to denote the predicted state variables and measurement variables in the ith control cycle. According to the transition function $f(\cdot)$, the system predicts the current state variable $\hat{X}^{(i)}$ from the previous state $\hat{X}^{(i-1)}$. From predicted current state $\hat{X}^{(i)}$ and the system function $h(\cdot)$, we can also have the predicted measurement variable $\hat{Z}^{(i)}$. At last, the difference $Z^{(i)} - \hat{Z}^{(i)}$ helps the system further improve the state variable prediction. Here, we can see that the core of Kalman filter lies in feedback adjustment between the state variables and the measurement variables.

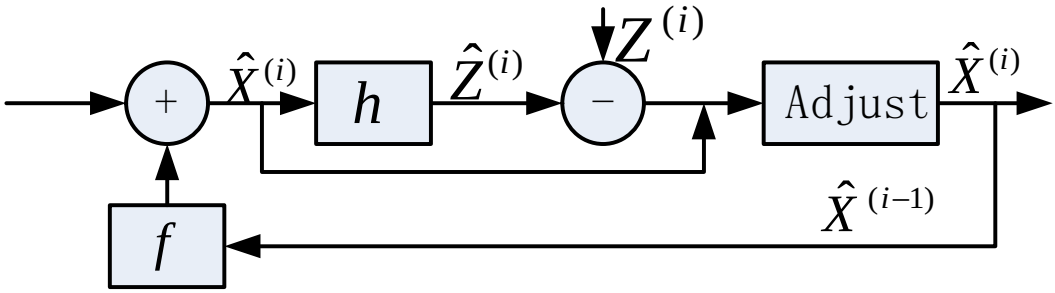


Figure 4.4. The architecture of Kalman filter

4.3.3 The Stability-aware Proxy Design

We detail our SLA maintenance module in Fig. 4.5. Our SLA maintenance module has only one output, the dynamic service rate. In the SLA maintenance module, there is a measurement interface, collecting the request arrival rate and response time of different service levels in the previous control cycle. This information becomes the input for our prediction algorithm (the Kalman filter in Fig. 4.5), whose output is the predicted request arrival rates. After that, the service rate scheduling module in the SLA maintenance system transforms the predicted request arrival rate into the service rate, with the objective to minimize the loss of the service provider.

Here, we use Kalman filters for the request arrival rate prediction, so as to maintain the system stable. We use separate Kalman filters for every service class instead of a single Kalman filter for all service classes.

Despite an overall Kalman filter for all service classes bringing more accurate control for the whole system, the reason of using separate Kalman filters is as follows: 1) the difficulty of constructing of system function for the overall Kalman filter. Kalman filters require the solution of the optimization problem in Eq. 4.1 as the system function. The current solutions for convex optimizations use search methods like the Newton method, which is hardly to be a system function for Kalman filters; 2) the overall Kalman filter lowers the scalability,

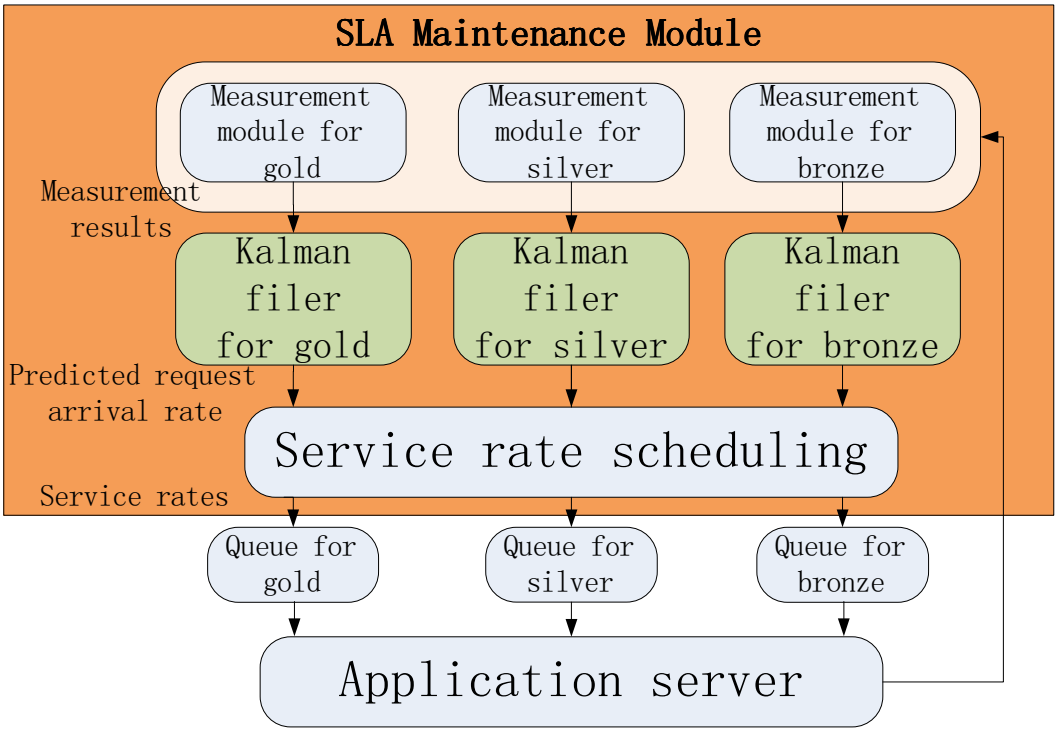


Figure 4.5. The architecture of the stability-aware proxy

since adding or dropping one service class from the system, the system function should be reconstructed. Therefore, we adopt separate Kalman filters for service classes, for this can greatly lower the complexity of system construction as well as to improve the scalability with an acceptable cost of performance.

The algorithm in the SLA maintenance module is shown as Algorithm 1, and it works as follows: firstly, the Kalman filter for each service class predicts the state variables X_k of the next cycle; secondly, according to X_k, and the allowed maximal service rate u, the proxy performs optimization to divide the total service rate u to each service class. At last, the queue for each service class receives the requests from clients, and the forwards these requests to application servers according to the respective assigned service rate.

Here, the unknown functions are *Initiate(·)*, *KalmanFilter(·)*, and *Optimize(·)*. For *Initiate(·)*, we may intuitively choose small values for state variables. This is because systems are normally vacant just after boot-up (for example, the request arrival rate and the non-

empty-time is low in the beginning); the function *Optimize(·)*, which optimizes the problem of Eq. 4.1, can be sovled by Newton search methods; at last, the function *KalmanFilter(*$\cdot$) is the core part of our stability-aware proxy. To realize the function *KalmanFilter(*$\cdot$) in Algorithm 1, we have to take the following aspects into consideration, and we will introduce the detail design of the function $KalmanFilter(\cdot)$ in the next subsection:

- The definition of the state variable vector X_k and the process variable vector Z_k;
- The detail configuration of IEKF.

4.3.4 The Design of Kalman Filter Algorithm

In this subsection, we introduce the key function of the stability-aware proxy, *KalmanFilter(*$\cdot$).

Firstly, as the response time estimation is an important part of our system, we define our response time estimation algorithm as Eq. 4.5 which extends the GPS model in [30]. The reason we employ the GPS model, but not the queueing models like the MM1 model, or the GG1 model, is that these traditional queueing models (either MM1 model or GG1 model) are only applicable for the systems in the steady state, while the GPS can also capture the transient behaviors from various application workload [30]. However, as it is studied in Section III.C, the GPS model in [30] do not take request execution time into consideration. Thus we extend the GPS model with an extra part T_{e_k}, which denotes the request execution time in the application server for service class k. Here, we also use T_{q_k} to denote the queueing time for k, and T_{s_k} to denote the service time. We let q_k be the initial queue length of the service class k, w_k be the non-empty-time, which is the duration the queue for service class k is not empty in one control cycle, and λ_k and μ_k are respectively the current request arrival rate and the service rate for service class k. w is the duration of one control cycle. In our system $w = 20$ *seconds*.

$$\begin{aligned} T_k(\lambda_k, \mu_k, w_k) &= T_{q\_k} + T_{s\_k} + T_{e\_k} \\ &= \frac{w_k}{w \times \mu_k}(q_k + \frac{w_k}{2}(\lambda_k - \mu_k)) + \frac{1}{\mu_k} + T_{e\_k} \end{aligned} \tag{4.5}$$

Algorithm 4.3.1 StabilityAwareProxy()

Input: The total available service rate μ, the measurement variable vector $Z_k, k \in \{Gold, Silver, Bronze\}$, and the previous control cycle state variable X'_k.

Output: The allocated service rate for different service classes μ_k.

1: **while** $k \in \{Gold, Silver, Bronze\}$ **do**

2: $Initiate(X'_k)$;

3: **end while**

4: **while** The proxy is running **do**

5: **while** $k \in \{Gold, Silver, Bronze\}$ **do**

6: $X_k = KalmanFilter(Z_k, X'_k)$;

7: **end while**

8: Perform optimization for Eq. 4.1.

9: $(\mu_g, \mu_s, \mu_b) = Optimize(\mu, (X_g, X_s, X_b))$;

10: **while** $k \in \{Gold, Silver, Bronze\}$ **do**

11: Update $X'_k = X_k$;

12: **end while**

13: Update μ from application servers;

14: **end while**

Secondly, as λ_k, w_k, and T_{e_k} in the next control cycle can only be predicted, while the others can be measured. Therefore the state variable vector is $X_k = (\lambda_k, w_k, T_{e\_k})$.

Thirdly, the values of T_k, λ_k, and w_k of the current control cycle can be observed. The measurement variable vector is $Z_k = (T_k, \lambda_k, w_k)$.

Fourthly, we need to define the transition function $f(\cdot)$ and $h(\cdot)$. We use $X^{(i)}$ to denote the state variable vector in the ith control cycle. We assume that the state variables

do not change during two continuous control cycle, that is $X^{(i+1)} = X^{(i)}$. This is because our objective is to stabilize the state variable prediction for the unchanging user behaviors, but not the prediction accuracy of the request arrival rate. Users may employ any prediction method, such as Weiner filter [93], or Auto Regression Moving Average (ARMA) here. However, as a work that focuses only on the stability of the system, we simply assume $X^{(i+1)} = X^{(i)}$. As a result, the function $f(\cdot)$ is as Eq. 4.6. Also, and the function $h(\cdot)$ is as Eq. 4.7. Here, we let $\lambda = \{\lambda_g, \lambda_s, \lambda_b\}$, $w = \{w_g, w_s, w_b\}$, $T_e = \{T_{e\_g}, T_{e\_s}, T_{e\_b}\}$ and $T(X) = \{T_g(X_g), T_s(X_g), T_b(X_g)\}$ ($X_k = (\lambda_k, w_k, T_{e\_k})$).

$$f(X^{(i+1)}) = (\lambda^{(i)} \quad w^{(i)} \quad T_e^{(i)})^T \tag{4.6}$$

$$h(X) = (T(X) \quad \lambda \quad w)^T \tag{4.7}$$

According to the solution of IEKF, we have the algorithm for function $Kalmanfilter(\cdot)$ in Algorithm 4.3.2. The Objective of the Algorithm 4.3.2 is to present an estimation for state variables as a strategy to curb the instability. Here, we use ζ to denote the iterative time of IEKF (in our experiment we have $\zeta = 4$), $\hat{X}_k$ to denote the priori state variable for service class k, and $\tilde{z}$ to denote the measurement residual. Also, we use S_k to denote the innovation covariance for service class k, K_k to denote Kalman gain for service class k, let P_k be estimate covariance for service class k, let H be the Jacobian matrix $\frac{\delta h}{\delta X}|_{X=\hat{X}_k}$, and let F be the Jacobian matrix $\frac{\delta f}{\delta X}|_{X=\hat{X}_k}$.

4.3.5 The Noise Covariance Configuration

In Kalman filter, the prediction state variable adjusting rate and stability contradict with each other in noise configuration. As it is shown in Eq. 4.4, Kalman filter defines the noise covariance in matrix Q and R, which are respectively the covariance of process noise and measurement noise. According to the properties of Kalman filter, firstly, the process noise ω determines the adjustment step length. The larger the adjustment step length is, the more

Notation	Definition
ζ	The iterative time of IEKF
$\hat{X}_k$	The priori state variable for service class k.
$\tilde{z}$	The innovation, or measurement residual.
S_k	The innovation covariance (the output of estimation error) for service class k.
K_k	Kalman gain for service class k
P_k	estimate covariance for service class k
H	The Jacobian matrix $\frac{\delta h}{\delta X}\|_{X=\hat{X}_s}$
F	The Jacobian matrix $\frac{\delta f}{\delta X}\|_{X=\hat{X}_s}$

Table 4.2. The notation table for the IEKF

quickly the state variables adjust, but at the cost of lowering the stability, and vice versa. Secondly, v defines the level of noise in measurement variables. A larger observation noise configuration makes the filter adjust more slowly, and consequently assures a better stability, and vice versa.

For the reliable consideration, we use the classic noise covariance configuration method. As it is introduced by [87], the noise covariance configuration R of the Kalman filter is generally pre-computed by offline simulation. After that, we have $R = (0.005, 0.40, 0.0250)$. Nevertheless, the configuration of the noise covariance of Q is more complex because of the dynamically changing process noise. To assure the system stability from close loop between the predicted request arrival rates and the response time, we can assume the original system has a very small process noise. Here, we also use the offline simulation to have $Q = (0.0500^2, 0.0010^2, 0.0010^2)$ to assure both stability and adjusting rate in our system.

Algorithm 4.3.2 KalmanFilter()

Input: Previous state variable X'_k, measurement variable Z_k.

Output: The state variable X_k.

1: **for** $i = 1$ to ζ **do**

2: Initialize priori state variable $\hat{X}_s = f(X'_k)$;

3: Initialize priori estimate covariance $\hat{P}_k = F_k P_k F_k^T + \omega$;

4: Get innovation $\tilde{z} = Z_k - H_k(\hat{X}_k)$;

5: Get innovation covariance $S_k = H_s \hat{P}_k H_k^T + v$;

6: Get Kalman gain $K = \hat{P}_k H_k^T S_k^{-1}$;

7: Update $X_k = \hat{X}_k + K\tilde{z}$;

8: Update estimate covariance $P_k = (I - KH_k)\hat{P}_k$;

9: Update $X'_k = X_k$;

10: **end for**

The noise covariance configuration is differed with the different system capacities. Generally, the larger system capacity may require a larger process noise covariance because the request arrival rate may change more quickly. Similarly, a system with a smaller capacity may require a smaller process noise covariance, as a smaller capacity may have a smaller request arrival rate change rate and requires a higher stability.

4.4 Performance Evaluation

4.4.1 Experimental Setup

We use six IBM servers to conduct experiment for the performance evaluation. Three of them are are configured as proxies, and others are application servers. All these servers have two Intel Xeon E5405 CPUs, one 4GB RAM and a 7200 RPM hard disk. The operating system is Fedora 12 with the Linux 2.6.31; three application servers run Apache Tomcat 6.0.29

to provide HTTP service. We implement the sub-controller which provides performance feedback on each application server with Perl 5.0 to control the CPU utilization below 90%, as it is used in [99]. To emulate the time consumption of additional overheads in the servers, the HTTP responses for each requests will be performed after a sleep (in our experiment, the sleep time is set as 90 ms). Our SLA maintenance module is implemented on Tinyproxy [126], a widely used proxy for performance evaluation.

We use the Eq. 4.8 as the SLA violation loss function in our system. This function is also widely applied in many other studies, such as [99] and [21]. We configure different service classes by setting the different importance value I_k in the Eq. 4.8. The importance values (I_k in Eq. 4.8) for each service class (I_g, I_s and I_b) are set to 4, 3, and 2 respectively. For all service classes, we set the response time control target as 200ms for all service classes. This is the base-line response time when our system is fully loaded[1]. We use Eq. 4.8 as the SLA violation function for the following reasons:

- Eq. 4.8 satisfies the SLA violation loss function criteria in Section 4.2.1;
- This SLA violation loss function cares about the response time control target. Therefore it is also used in the works such as [21] and [30].
- we use Eq. 4.8 as an example of the SLA violation loss function, and it can also be replaced by other functions satisfying the SLA violation loss function criteria in Section 4.2.1 of this chapter.

$$D_k(T_k) = \frac{I_k}{2}(T_k - d_k) + \sqrt{(T_k - d_k)^2 + 0.5} \tag{4.8}$$

In our experiment, the workloads (user requests) are generated by IBMs private synthetic workload generator [30]. The HTTP responses for each request will be generated after

[1]This configuration (control target equal to 200ms) indicates that the SLA system will start the optimization when the system is overloaded. It gives us an observable start point for comparison and will not bias our conclusion.

Round	Duration	Avg. concurrent session number	Avg. think time
1	around 5 min.	5	500 ms.
2	around 4 min.	5	200 ms.
3	around 4 min.	8	200 ms.
4	around 5 min.	11	200 ms.
5	around 7 min.	14	200 ms.
6	around 7 min.	17	200 ms.
7	around 11 min.	24	200 ms.

a sleep. This sleep time is set to 90ms based on the default settings of the existing IBM SLA system. Based on the basic patterns of Internet user behavior [4], the users think time follows a truncated negative exponential distribution in our experiment. We run our experiment for 33minutes as it is shown in Table 4.4.1 for the stability aware proxy. The experiment is divided into 5stages and each round costs 5 to 10minutes.

The detailed configuration can be found in the following table where we increase the total number of sessions to introduce more workloads to the system. Note that we also change the total number of sections in each stage following a normal distribution with standard deviation of one. This is to emulate the user dynamics, such as the random failure, in real-world systems. While the for the Zhangs proxy [99] and the AR(1) proxy [21], we only complete the 20 minutes experiment in the first two stage, since these two proxies crash while the system is overloaded (e.g. the response time of these two system can be large than 10 seconds).

We compare the performance of proxies as follows:

- Offline optimal solution: The actual request arrival rate in the next control cycle is used instead of the request arrival rate prediction algorithm. The result of the offline optimal solution is used as a calibration to compare the result of different schemes.

- Stability-aware proxy: the proxy with SLA maintenance module;
- Conventional [30]: This is currently adopted by IBM, where we set (i.e., predict) the request arrival rate of the next control cycle to that of the current control cycle;
- AR(1) [21]: This is a prediction algorithm which minimizes the squared prediction error between next and current control cycle.

We use response time as a main measurement metric. Note that our primary concern is that the response time must follow the service level agreement.

4.4.2 Experiment Results

Fig. 4.6 presents the SLA violation loss of the three service classes in the stability-aware proxy. It shows the SLA violation loss of the stability-aware proxy is quite close to offline optimal solution all the time. Specifically, in the first 20 minutes, while the workload is under the system capacity, the stability-aware proxy achieves the same performance as the offline optimal algorithm. After the 27th minute and the total workload approaches the system capacity, the SLA violation loss of the stability-aware proxy is slightly higher than the offline optimal results. Nevertheless, the differences are not larger than 10%.

To demonstrate the improvement from the current existing proxy algorithms, we further compare our proxy algorithm with conventional proxy and AR(1) in Fig. 5.4 in the first 20 minutes. We do not present the simulation results of the traditional proxies because the instability becomes even severe after 20 minutes and the system occasionally crashes because some service classes starves. To provide a uniform comparison, we only compares the performance of the first 20 minutes. We can see that while the workload is small, all the algorithms have the same performance. However, while after about 13 minutes, we can see the SLA violation losses of current existing algorithms (conventional and AR(1) algorithms) are about 50% larger than the offline optimal algorithms on average, while the stability-aware algorithm can also maintain an optimal level. According to our previous study, this is because the current existing algorithms neglect of the close loop between the predicted request

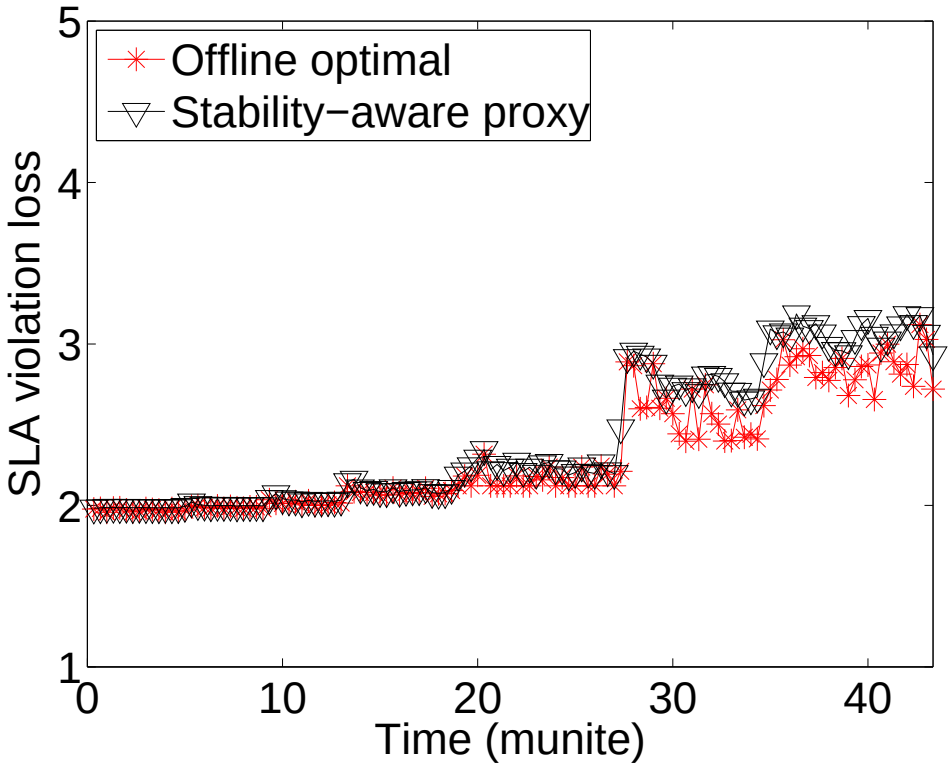


Figure 4.6. The SLA violation loss of the stability-aware proxy

arrival rate and the response time. To further confirm this reason, we provide the request arrival rates of the different algorithms and as well as the request response time.

To demonstrate the actual performance of the three proxies, we firstly shows the request response time in each type of proxies in Fig. 4.8. Fig. 4.8 (a) presents the response time of the three service classes of the stability-aware proxy. We see that the response time normally increases with the request arrival rate. In the first 25 minutes, there is no apparent service differentiation since the system is underload. We can observe an obvious service differentiation after the last two rounds while the system is overloaded. To demonstrate the improvement of the stability-aware proxy, we compare its response time with the two traditional proxies from Fig. 4.8 (b) to Fig. 4.8 (d). We can see a stable response time in the stability-aware proxy, but in the other two algorithms a large fluctuation can be observed after about 14 minutes. In detail, we can see a larger fluctuation in response time in the con-

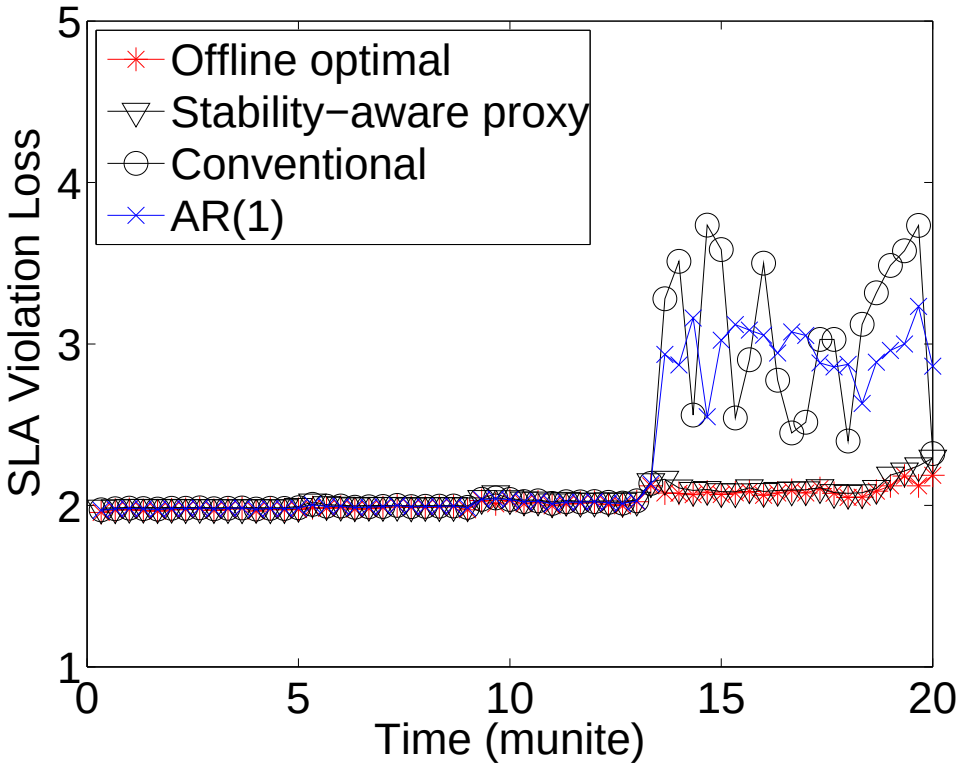


Figure 4.7. The SLA violation loss comparison of all algorithms

ventional proxy than that in the AR(1) proxy. This may be because of the AR(1) algorithm is able to filter a specific level of noises. However, as AR(1) algorithm can only filter the Gaussian noises but not control the close loop, the instability remains in this algorithm.

The response time prediction error is an importance criterion for stability. Fig. 4.9 (a) presents the response time prediction error of the stability-aware proxy. We can see that in the first five rounds of the simulation, the differences between the predicted and actual response time are generally zero. However, the slight errors are also observed at the beginning of each simulation round, since the slow state variable adjustment of the IEKF to assure the system stability. After 30 minutes, we can see that the average response time is more than 200 ms and the bias is below also 10%. While in the 35 minutes, the average response time is more than 300 ms and the error is less than 30 ms. In general, we can see that the stability-aware has 10% bounded error. Nevertheless, Fig. 4.9 (b), Fig. 4.9 (c) and Fig. 4.9 (d) compare

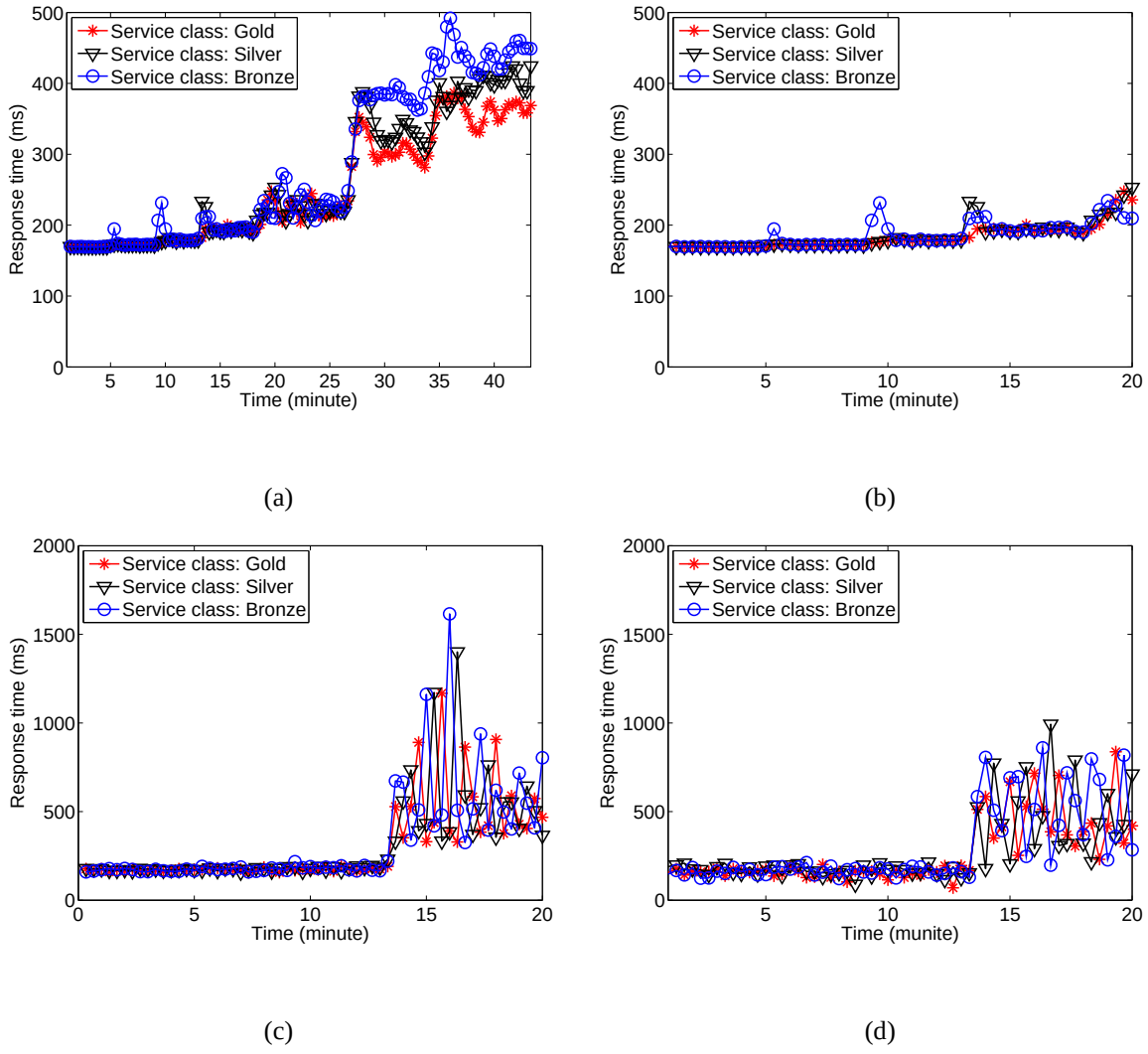


Figure 4.8. The response time as a function of running time, (a)(b) stability-aware proxy (c) conventional proxy, (d) AR(1) proxy

the prediction errors of the three proxies. Compared with the stability-aware proxy, the the prediction errors in conventional proxy can be larger than one second and that in the AR(1) proxy is nearly one second. From this, we can see a large improvement in stability of the stability-aware proxy.

To make a comparison, Fig. 4.10 shows the request arrival rates of the proxy algorithms. Fig. 4.10 (a) shows the request arrival rate to the stability-aware proxy. It is observed that, in the stability-aware proxy, from the beginning of the simulation to 20 minutes, the request arrival rate keeps stable. Nevertheless, request arrival rate fluctuates sightly after 25 minutes. Also, we compare its request arrival rates to the traditional proxies from Fig. 4.10

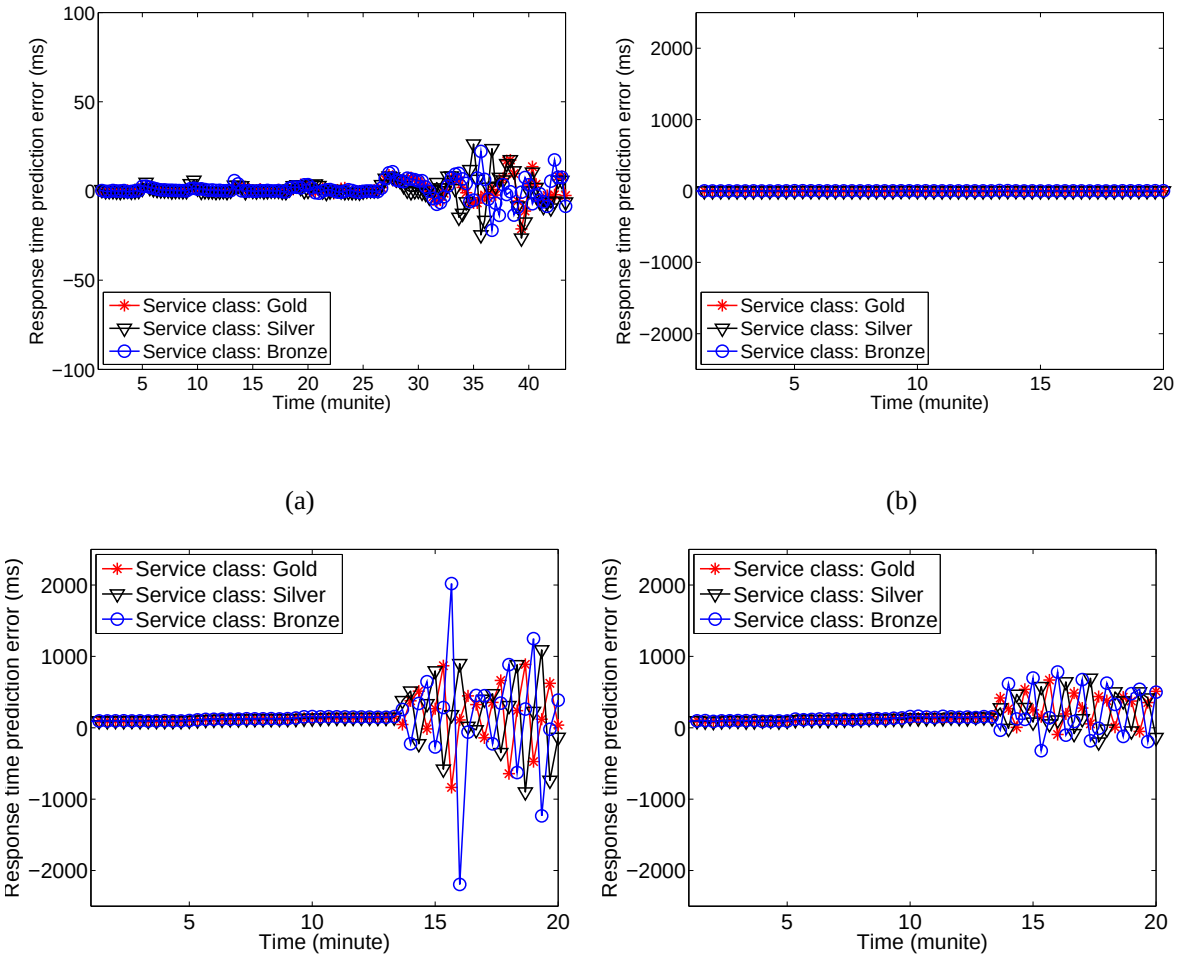


Figure 4.9. The response time prediction error, (a)(b) stability-aware proxy (c) conventional proxy, (d) AR(1) proxy

(a) to Fig. 4.10 (d). Comparatively, between roughly the seventh minute to the 20th minutes, the conventional proxy and the AR(1) proxy have a more vibrating request arrival rate since these two types of proxies are unable to control the instability caused by the close loop. To further demonstrate how the IEKF controls the system stability, we present the comparison of predicted and actual request arrival rates in Fig. 4.11. Fig. 4.11 (a) (b) and (c) respectively show the comparison from the service class gold, silver and bronze. In general, we can see that in the first four rounds, the actual request arrival rates remain stable and the predicted request arrival rates quickly converge to actual ones and maintain stable. Besides that, in the last three rounds, the actual request arrival rates fluctuate but gradually converge to the pre-

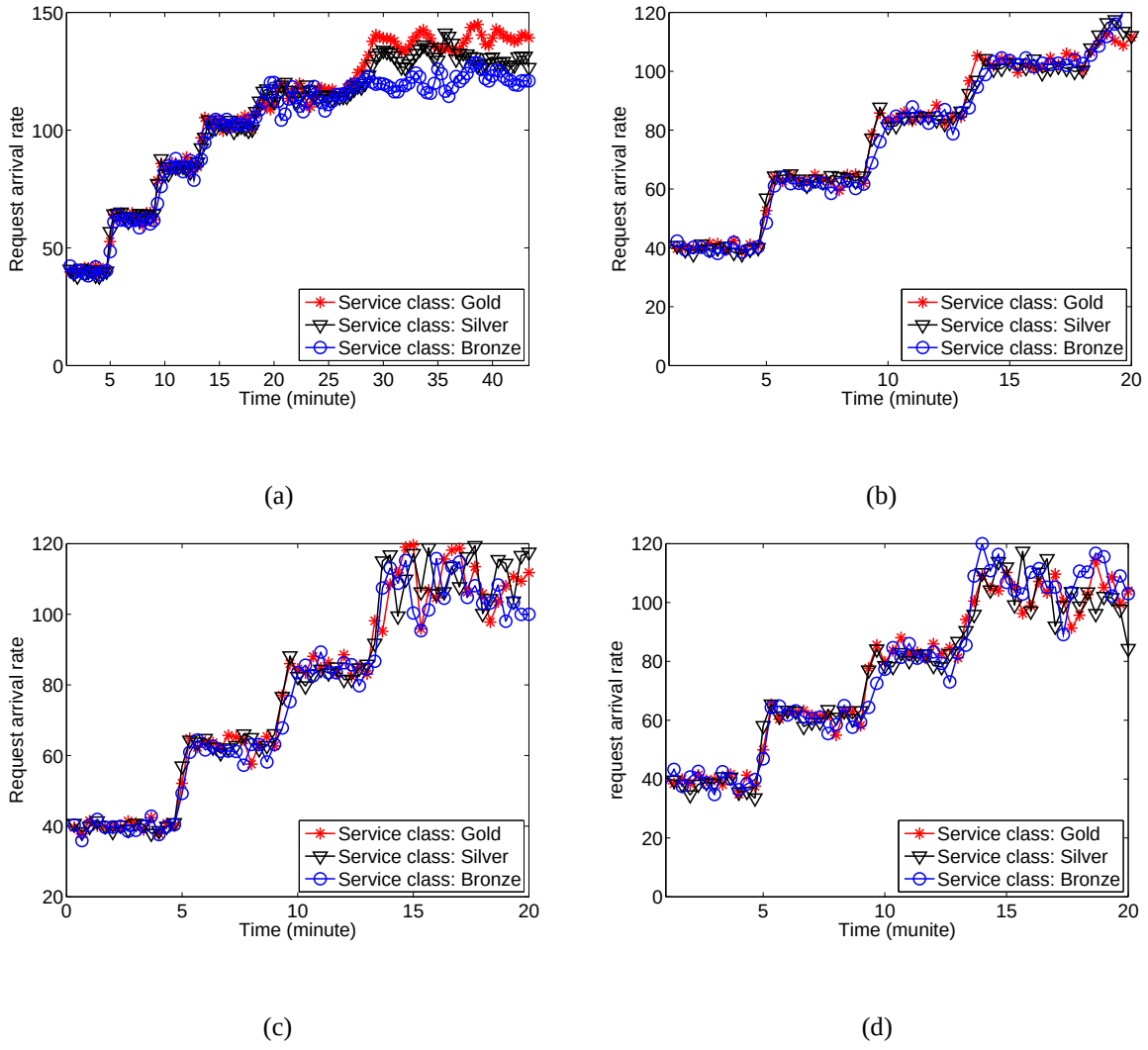
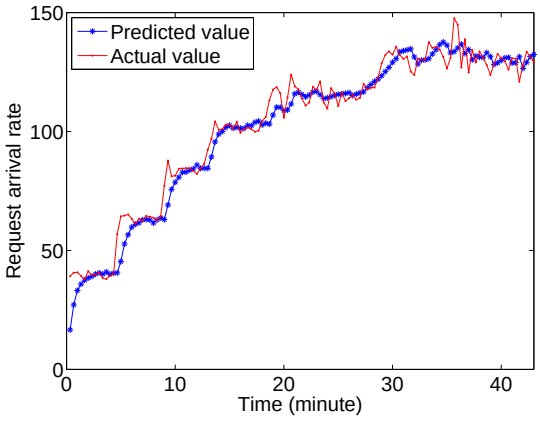


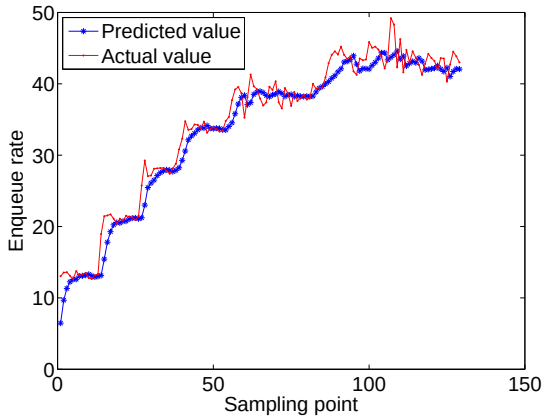
Figure 4.10. The request arrival rate comparison (a)(b) stability-aware proxy (c) conventional proxy, (d) AR(1) proxy

dicted request arrival rate, since the predicted request arrival rates act as a strategy to control the system stable.

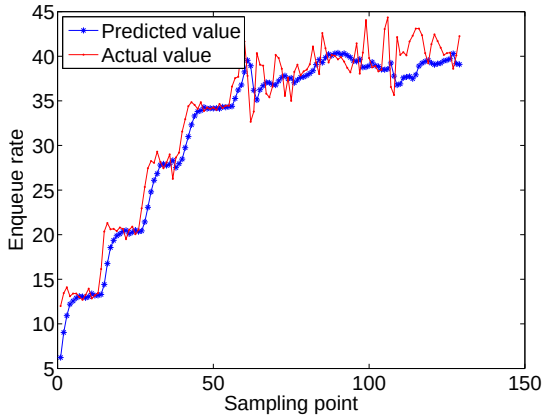
We further show the overhead of the application servers. Fig. 4.12 presents the CPU utilization of the simulations. Similarly, we can see that the stability-aware proxy is able to keep the CPU utilization stable in the first five rounds. In the last two rounds, the application servers reach the control target at 90%, and the CPU utilization fluctuates slightly around the control target. However, compared with the the stability-aware proxy, the conventional proxy and the AR(1) proxy can only remain CPU utilization stable in the first three rounds and in the fourth round, we can see an obvious CPU utilization fluctuation around 77%, this



(a) The request arrival rate for gold

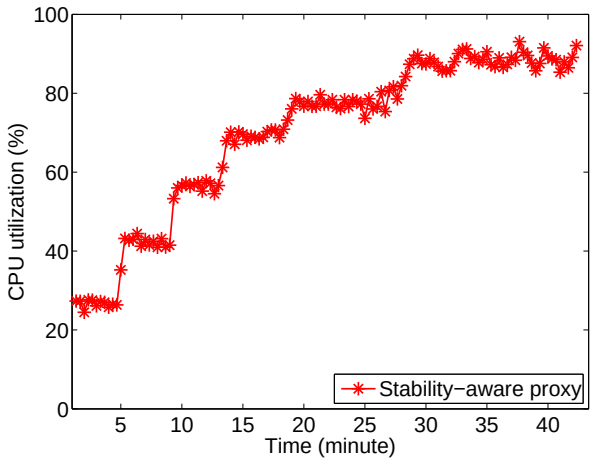


(b) The request arrival rate for silver

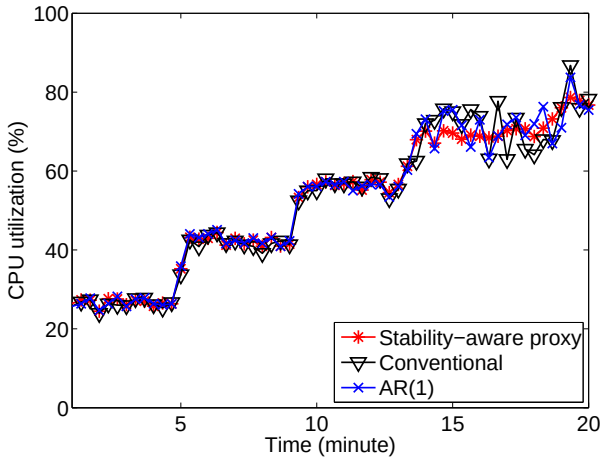


(c) The request arrival rate for bronze

Figure 4.11. The comparison of predicted and actual request arrival rates



(a)



(b)

Figure 4.12. The CPU utilization of the application servers

indicates the instability in the request arrival rate in these two types of proxies. Also, from this we can see that the instability hampers the full utilization of the servers.

4.5 Conclusion

In this chapter, we noticed the stability problem in the sequential job scenario, and aimed at minimizing the total SLA violation loss under the constrained system resources. We first

found that the system instability comes from the close loop impact between the predicted request arrival rate and the request response time. We evaluated such instability in experiment and model the root cause of the instability. We proposed a stability-aware proxy based on IEKF to minimize the SLA violation loss as well as to stabilize the system under the SLA constrain. Our design only needs to plug an SLA module to other systems, and minimally affects other parts of the systems. In experiment, we found that our scheme can maintain the system stable and achieve the minimization of the SLA violation loss.

CHAPTER 5

MINIMIZING THE BILLS OF CLOUD SERVICES BY MULTI-PROVIDER SELECTION

5.1 Overview

This chapter study on the deployment of the Internet applications on cloud servers according to different scales of the user traffic. In detail, we present a cost-efficient scheme to deploy Internet applications on the cloud servers. We choose the cloud servers from different service providers. The essence of this idea is to leverage the different rates of allocating resources in different service providers, to cover the different resource requirement of different modules in one application.

The organization of this chapter is as follows: Section 5.2 presents the definition of the problems as well as their complexity analysis for this chapter; Section 5.3 proposes the algorithms for the problems; In Section 5.4, we show the simulation evaluation for our algorithms; Section 5.5, we presents a real experiment, where we deploy a real MMORPG game in the cloud server for performance evaluation; finally section 5.6 concludes this chapter.

5.2 The Problems and Complexity Analysis

5.2.1 The General Problem

In this paper, our focus is only restricted on the deployment of large scale server clusters, since the deployment of server clusters is more complex and the payment is not subtle. Besides that, migrating parts of application server clusters to the cloud takes also quite an

important part of IaaS business, such as [110] [111]. Therefore we have these assumptions for the study in this chapter.

1. We neglect price of the software that perform uniform monitoring or deployment among different service providers. There are different tools with varied prices, and there are also free and open source ones. Besides that, the software is also charged quite differently from cloud services. Generally, once they are bought, they can be used forever, as compared to the cloud servers are charged monthly.

2. Therefore we assume that each server runs one application, since this is common in the server clusters for the consideration of caching and for bettering the maintenance. Typical examples are such as the web servers and game servers. In detail, it is common that the web server clusters separate the distribution of different contents (such as pictures, texts, and videos), and the delivery of different application services into different servers [116]. Game server clusters separate their different functions, such as login, game logic, billing, and database services, into different servers [94].

Assume that there are $\mathcal{L}$ service providers. From the cloud service provider point of view, a *service instance* τ_j is a virtual machine provided by a service provider $l, l \in [0, \mathcal{L}-1]$ with a set of resource and a price c_j. Assume there are o types of resource such as CPU, RAM, storage, etc. τ_j can be illustrated by a vector of resource, and $\tau_j = \{r_{0j}, r_{1j}, \cdots, r_{o-1j}\}$, where r_{ij} denotes the amount of resource type i for instance j. We assume that there is a special service instance $\tau_0 = \{0, 0, \cdots, 0\}$ with cost 0. This indicates that the service provider does not provide any service. Assume each service provider has N_l service instances and $\mathcal{N} = \sum_{l=0}^{\mathcal{L}-1} N_l$.

From the user point of view, the users usually care how many concurrent users are needed for their applications. Typical examples are [110] [111], where enterprise users construct IaaS solutions according to the number of concurrent online clients. Assume an enterprise needs to accommodate $\mathcal{K}$ concurrent users and there are M applications $\mathcal{B} = \{b_0, b_1, \ldots, b_{M-1}\}$. Without loss of generality, assume each user activates one application

at a time. The cloud maintains one *session* for each online user. In [80], it is demonstrated that the sessions of the same application consume roughly equal resource and the resource consumption on one server is as Eq. 5.1. Let Θ_{ij} be the resource consumption for application b_i under service instance j. More specifically, $\Theta_{ij} = \{\theta_{ij}^{(1)}, \theta_{ij}^{(2)}, \cdots, \theta_{ij}^{(o)}\}$, where $\theta_{ij}^{(k)}$ denotes the resource consumption of resource k using service instance j for each session of application i. According to [80], certain basic resource is needed for each service instance to run software supports and the application, even if there is no session running. For example, the software supports for the web applications may be the Linux and Apache Tomcat, and both these softwares consume CPU cycles, RAM, and storage capacities. We define A_{ij} as the basic resource consumption of application i under service instance j and we have $A_{ij} = \{a_{ij}^{(1)}, a_{ij}^{(2)}, ..., a_{ij}^{(o)}\}$. Also, we let $\mathcal{G}(i,j)$ be the maximum number of sessions of application i that can be concurrently handled in service instance j. We have Eq. 5.1. Here $a_{ij}^{(k)}$ and $\theta_{ij}^{(k)}$ follow distributions. In real deployment, users may choose the average value of $a_{ij}^{(k)}$ and $\theta_{ij}^{(k)}$ to cover the average requirement. However, in our paper, in order to assure the system resource can cover the consumption in all cases, we choose $a_{ij}^{(k)}$ and $\theta_{ij}^{(k)}$ respectively to be the maximal values of the distributions.

$$A_{ij} + \mathcal{G}(i,j) \times \Theta_{ij} \leq \tau_j \tag{5.1}$$

A user can select multiple service instances from a service provider l. For each service instance, the user can select multiple copies. Let s_j denote the number of copies of instance j. A *Multi-Provider Selection* is an assignment $\mathcal{S} = (s_1, s_2, \ldots, s_{\mathcal{N}})$.

Definition 5.2.1. *The Multi-Provider Selection Problem (MPS): Given a requirement of $\mathcal{K}$ users which will run M applications, find a Multi-Provider Selection, such that the cost is minimized and Eq. 5.1 holds.*

5.2.2 Two Variants of MPS

We design the MPS problem as general as possible. There can be additional constraints. We discuss two MPS variants, one in the service provider side and one in the user side.

1) Special price strategies from the service providers

For promotion, cloud service providers have package sales. The general principle of such plans is that with the same price, user can receive more resource than he purchases service instances individually by buying a wholesale (usually big). An example of package sales is from GoGrid [114], with sale plans of "Professional Cloud", "Business Cloud", "Corporate Cloud", and "Enterprise Cloud".

More specifically, let $\mathcal{C}_j^w$ denote the jth wholesale price of a service provider. The amount of resource for this wholesale is $\tau_j^w = \{r_{0j}^w, r_{1j}^w, \cdots, r_{o-1j}^w\}$. Then for any combination of individual service instances of this service provider, with a combined cost of $\mathcal{C}$ and the combination resource τ, we have $\mathcal{C} \leq \mathcal{C}_j^w \Rightarrow \tau \preceq \tau_j^w$. Here $\preceq$ indicates that for each type of resource r_i in τ, $r_i \leq r_{ij}^w$. This means that to get a certain amount of combined resource, purchasing from a wholesale has the minimum cost. We call the MPS problem that must satisfy such constraint as MPS$_S$.

2)Special user requirement

The users also have extra constrains. For example, some applications require that the service instances come from the same IaaS vendors. A typical example is that suppose an online game company is considering employing the IaaS to host services for ten thousand online players. Among several types of different game servers, there are two applications, the *game gates* and *game servers*, requiring low latency and high throughput in communication between them. This is because the game gates act as the proxy of game servers. High communication latency and low throughput will cause poor user experience. To minimize the communication latency and to maximize the throughput of the two applications,

it is suggested to choose service instances of the two types of applications from one service provider.

More specifically, assume there are ω groups of applications $m_{ij_0}, m_{ij_1}, \ldots,$ where $i \in [0, \omega - 1]$. Each group of applications must use service instances from the same service provider. We call the MPS problem that must satisfy such constraint as MPS$_C$.

5.2.3 The Complexity of MPS

We will show in Section 5.3.1, that if c_j and $\mathcal{G}(i, j)$, i.e., the price and the maximum number of sessions, are correlated, we can develop an optimal solution.

The general MPS problems are NP-hard as follows.

Theorem 5.2.1. *MPS is NP-complete.*

Proof. We reduce MPS to Minimum Set Cover problem which is proven to be NP-hard [41]. The Minimum Set Cover problem is as follows. There is a finite set S and a collection C of subsets, and we need to find a set cover for S by a collection of $C' \subseteq C$ that every member of S belong to at least one subset of C' and the cardinality of $|C'|$ is minimized.

For each instance of the Minimum Set Cover problem, we construct an instance of MPS as follows. For each element s_i of the finite set S, we create a resource type i. For each subset $u_j \in U$ where $u_j = [s_{j0}, s_{j1}, \ldots]$, we create a service instance $\tau_j = [r_{0j}, r_{1j} \ldots]$ where $r_{kj} = 1$ if resource type k is created or $r_{kj} = 0$ if resource type k is not created. The cost of τ_j is equal to the number of resource type that are created for this service instance. In each case, $A_{ij} = \{a_{ij}^{(1)}, a_{ij}^{(2)}, ..., a_{ij}^{(o)}\}$ and for any value of i, j, and o, $a_{ij}^{(o)} = 0$. For $\Theta_{ij} = \{\theta_{ij}^{(1)}, \theta_{ij}^{(2)}, \cdots, \theta_{ij}^{(o)}\}$, for any value of i, j, and o, $\theta_{ij}^{(o)} = \tau_{ij}^{(o)}$.

We can see that the instance constructed is a subset of MPS problem and if MPS can

be solved in polynomial time, we can directly get the optimal cardinality of the Minimum Set Cover problem. Therefore, MPS problem is NP-hard. □

We can also easily prove that MPS$_S$ and MPS$_C$ are NP-Complete as the MPS problem is a special case of MPS$_S$ and MPS$_C$ problems

5.3 Algorithms

5.3.1 An Exact Algorithm for the Special Case

We first develop an exact algorithm for the MPS problem in a special case where c_j and $\mathcal{G}(i,j)$ are correlated. The correlation is defined as 1) for each $i = 1, 2, ..., M$ and $j = 1, 2, ..., N-1$ $\mathcal{G}(i,j) < \mathcal{G}(i,j+1)$; 2) $c_j < c_{j+1}$ and 3) $\frac{\mathcal{G}(i,j)}{c_j} < \frac{\mathcal{G}(i,j+1)}{c_{j+1}}$. Intuitively, this means that higher price service instances have a larger service capacity and the larger service capacity the customer have purchased, the higher discount the customer can enjoy. This kind of quantity discount is a common promotion strategy, such as the pricing of GoGrid IaaS cloud [114]. We believe these are meaningful for some practical applications.

We first present an important attribute of the MPS problem in the special case. We have the lemmas as follows.

Lemma 5.3.1. *If x^*_{ij} is the optimal number of service instance j selected for application i for the MPS problem satisfied with the 'quantity discount' assumption, the following inequality holds: for $j = 1, 2, ..., N-1$, $x^*_{ij} < \lfloor \frac{\mathcal{G}(i,j+1)}{\mathcal{G}(i,j)} \rfloor$.*

Proof. Suppose for the sake of contradiction that we have $x^*_{ij} \geq \lfloor \frac{\mathcal{G}(i,j+1)}{\mathcal{G}(i,j)} \rfloor$ for some index $j < N-1$, and let x^*_{iN} be a number large enough to let $\sum_{j=1}^{N} \mathcal{G}(i,j) x^*_{ij} > \mathcal{K}$. For this, we construct a new solution $z_{i1}, z_{i2}, ...z_{iN}$, and $z_{ik} = x^*_{ik} - \lfloor \frac{\mathcal{G}(i,k+1)}{\mathcal{G}(i,k)} \rfloor$, and $z_{ik+1} = x^*_{ik+1} + 1$. For other $j \neq k$ or $k+1$, $z_{ij} = x_{ij}$. Therefore we have:

$$\begin{aligned}&\sum_{j=1}^{N}\mathcal{G}(i,j)z_{ij}=\sum_{j=1}^{N}\mathcal{G}(i,j)x_{ij}-\mathcal{G}(i,k+1)-\\&\mathcal{G}(i,k)\tfrac{\mathcal{G}(i,k+1)}{\mathcal{G}(i,k)}\geq\sum_{j=1}^{N}\mathcal{G}(i,j)x_{ij}\geq\mathcal{K}\end{aligned}\tag{5.2}$$

and according to the assumption of 'quantity discount', we have

$$\begin{aligned}&\sum_{j=1}^{N}c_jz_{ij}=\sum_{j=1}^{N}c_jx_{ij}-c_{k+1}-c_k\tfrac{\mathcal{G}(i,k+1)}{\mathcal{G}(i,k)}\\&<\sum_{j=1}^{N}c_jx_{ij}\end{aligned}\tag{5.3}$$

Therefore $x_{ij}^{*}\geq\lfloor\frac{\mathcal{G}(i,j+1)}{\mathcal{G}(i,j)}\rfloor$ is not the optimal solution. □

Lemma 5.3.2. *If $x_{i1}^{*},x_{i2}^{*},...,x_{iN}^{*}$ is optimal for application i of an MPS problem, we either have $x_{ij}^{*}=\lfloor\frac{n_j}{\mathcal{G}(i,j)}\rfloor$ or $x_{ij}^{*}=\lceil\frac{n_j}{\mathcal{G}(i,j)}\rceil$, where $j=1,...,N$ and $n_N=\mathcal{K}$ and $n_{j-1}=n_j-\mathcal{G}(i,j)x_{ij}^{*}$. $\mathcal{K}$ is the required concurrent user number.*

Proof. Obviously, we have $x_{iN}^{*}\leq\lceil\frac{n_i}{\mathcal{G}(i,N)}\rceil$. Now suppose for the sake of contradiction that $x_{iN}^{*}<\lfloor\frac{n_i}{\mathcal{G}(i,N)}\rfloor$. With Theorem 2, we have $x_{ij}^{*}<\lfloor\frac{\mathcal{G}(i,j+1)}{\mathcal{G}(i,j)}\rfloor$. Equivalently, we get $x_{ij}^{*}\leq\lfloor\frac{\mathcal{G}(i,j+1)}{\mathcal{G}(i,j)}\rfloor-1$, and $x_{ij}^{*}\leq\frac{\mathcal{G}(i,j+1)}{\mathcal{G}(i,j)}-1$. Therefore

$$\begin{aligned}&\sum_{j=1}^{N}\mathcal{G}(i,j)x_{ij}^{*}\leq\sum_{j=1}^{N-1}\mathcal{G}(i,j)x_{ij}^{*}+\mathcal{G}(i,N)(\tfrac{n_i}{\mathcal{G}(i,N)}-1)\\&\leq\sum_{j=1}^{N-1}\mathcal{G}(i,j)x_{ij}^{*}+(\mathcal{K}-\mathcal{G}(i,N))\\&=\mathcal{G}(i,N)-\mathcal{G}(i,1)+n_i-\mathcal{G}(i,N)=\mathcal{K}\end{aligned}\tag{5.4}$$

The constrains are not satisfied.

□

We develop MPS-Special() to find the optimal solution for the special case. Following lemma 5.3.2, its procedure is: 1) for each application i, we sort the service instance in ascending order by the value of $\frac{\mathcal{G}(i,j)}{c_j}$; 2) we recursively use Lemma 5.3.2 to generate $2N$

Algorithm 5.3.1 MPS-Special()

Input: $\mathcal{C}$, R, $\mathcal{K}$, Θ, A.

Output: $C_{opt}^{(i)}$ for each $i = 1, 2, ..., M$

1: **for** i = 1 to M **do**

2: **for** j = 1 to N **do**

3: initiate $x_{ij} = 0$ for all i and j;

4: Calculate $\mathcal{G}(i,j) = Max\{Min_k \lfloor \frac{r_k - a_{ij}^{(k)}}{\theta_{ij}^{(k)}} \rfloor, 0\}$

5: $\rho_{ij} = \frac{\mathcal{G}(i,j)}{c_j}$

6: Sort service instance j by price c_j in an ascending order.

7: initiate $C_{opt}^{(i)} = c_N \lceil \frac{\mathcal{K}}{\mathcal{G}(i,N)} \rceil + 1$;

8: initiate $min\_item = -1$;

9: initiate $cost = 0$ and $n' = \mathcal{K}$

10: **for** j = N to 1 **do**

11: $number = \lceil \frac{n'}{\mathcal{G}(i,j)} \rceil$

12: **if** $cost + number \times c_j < C_{opt}^{(i)}$ **then**

13: $C_{opt}^{(i)} = cost + number * c_j$

14: $min\_item = j$;

15: **end if**

16: $cost = cost + (number - 1) \times c_j$

17: $n' = n' - \mathcal{G}(i,j)(number - 1)$

18: **end for**

19: **end for**

20: **end for**

solutions; and 3) we choose the one with the least cost. The time complexity of the sorting is $O(MNlogN)$ and the choice selection (without sorting part) is $O(2MN)$. The detailed algorithm and analysis can be found in Algorithm 5.3.1.

5.3.2 FPTAS for MPS

We develop a Fully Polynomial-Time Approximation Scheme (FPTAS) for MPS. In other words, we achieve a $(1+\epsilon)$ approximation algorithm for MPS.

The intuition of our algorithm is as follows. We divide service instances into two groups, 1) the service instances with a cost that is less than T ($T = \frac{\epsilon \times C_{min}^{(i)}}{2}$ and the evaluation proof is given in Theorem 5.3.1); we call this group ψ and 2) otherwise, Ψ. We conduct dynamic programming to compute an optimal solution with minimum cost for all the service instances in group Ψ. We call this algorithm LargeDP() (see Algorithm 5.3.4). We then use greedy algorithm to cover the unfilled space in group ψ. We call this algorithm SmallGreedy() (see Algorithm 5.3.5). Finally we combine the two costs and achieve our approximation. The essence is that for dynamic programming in Ψ, we can bound its computational complexity (running time), and for greedy algorithm in ψ, we can bound its accuracy.

The inputs of Algorithm MPS-FPATS() (see Algorithm 5.3.2) are $\mathcal{C}$, R, $\mathcal{K}$, Θ, A, $\mathcal{B}$, and the deviation from the optimal cost ϵ. Here $\mathcal{C}$ denotes the set of prices from all service instances $\{c_1, c_2, ..., c_N\}$, R stands for the resource set of service instances $\{\tau_1, \tau_2, ..., \tau_N\}$, Θ is the matrix comprised by all θ_{ij} and A is the matrix comprised by all a_{ij} where $i = 1, 2, ..., M$ and $j = 1, 2, ..., N$. The output is the optimal cost for each application i, $C_{opt}^{(i)}$. This algorithm is comprised by three subroutines:

- An algorithm to compute $C_{min}^{(i)}$, which is the estimated lower bound of the optimal cost for application i;
- A dynamic programming algorithm for the service instances in Ψ. We define this subroutine as LargeDP();
- A greedy algorithm to compute the total minimum cost by covering the unfilled concurrent user space from subroutine LargeDP(). As well, we define this subroutine as SmallGreedy(). The details are as follows.

We first develop function LowerBound() shown in Algorithm 5.3.3. The inputs are $\mathcal{C}$, the cost of all service instances; $\mathcal{K}$, the number of users; $\mathcal{G}$, the matrix that contains $\mathcal{G}(i,j), i = 1,2,...,M, j = 1,2,...,N$, and $\mathcal{G}(i,j)$ is the maximal number of application i users allowed by service instance j; the application set b_i. The output is $C_{min}^{(i)}$, the minimum cost of the optimal construction price for applications i. Note that from Eq. 5.1, we can compute $\mathcal{G}(i,j) = \max\{Min_{1\leq k\leq o}\lfloor\frac{r_{kj}-a_{ij}^{(k)}}{\theta_{ij}^{(k)}}\rfloor, 0\}$. In this function, we use the greedy algorithm to approximate the lower bound of the construction cost for applications.

We then develop function LargeDP(), the dynamic programming algorithm, for the service instances in group Ψ in Algorithm 5.3.4. The inputs are $\mathcal{C}$, the vector of costs of all service instances; $\mathcal{K}$, the number of users; $T = \frac{\epsilon\times C_{min}}{2}$; $J = \frac{\epsilon^2\times C_{min}}{4}$; b_i, the ith application; and $\mathcal{G}$, the matrix that contains $\mathcal{G}(i,j)$. Function LargeDP() firstly calculates the scaled price q_j for each service instance j. The scaled price q_j ($q_j = \lfloor\frac{c_j}{2^k J}\rfloor \times 2^k$) is used instead of the price c_j for service instance j to do the dynamic programming. The output is the table $\mathcal{T}$ for the dynamic programming, which contains the dynamic programming results. These results are triples $\{Q, W(Q), C(Q)\}$. Here, Q is the scaled price of one solution, $W(Q)$ is the associated number of concurrent sessions allowed by this solution with scaled price Q, and $C(Q)$ is the actual price for Q.

We then develop the greedy algorithm for the service instances of ψ, and it is shown in function SmallGreedy() of Algorithm 5.3.5. We use greedy algorithm to cover the unfill space for each solution $\{Q, W(Q), C(Q)\}$ in $\mathcal{T}$ and we let the cost for the unfill space of item $\{Q, W(Q), C(Q)\}$ be $\phi(\mathcal{K}-W(Q))$. The optimal cost for application i, $C_{opt}^{(i)}$ is minimal total cost of $\phi(\mathcal{K} - W(Q))$ and $C(Q)$.

Theorem 5.3.1. *Let C be the cost from algorithm MPS-FPTAS() and C^* be the optimal cost, we have $C \leq (1+\epsilon)C^*$.*

Proof. For each application i, here we let C_i^* be the optimal cost for the MPS problem, and let $C_{min}^{(i)}$ be the lower bound for the optimal cost C_i^*. Besides that, we also let $\phi(w)$ denote

Algorithm 5.3.2 MPS-FPTAS()

Input: $\mathcal{C}$, R, $\mathcal{K}$, Θ, A, $\mathcal{B}$, ϵ.

Output: for each application requirement b_i, the optimal solution $C_{opt}^{(i)}$,

1: **for** i = 1 to M **do**

2: **for** j = 1 to N **do**

3: $\mathcal{G}(i,j) = Max\{Min_k\lfloor\frac{r_{kj}-a_{ij}^{(k)}}{\theta_{ij}^{(k)}}\rfloor, 0\}$ and $\rho_{ij} = \frac{\mathcal{G}(i,j)}{c_j}$;

4: **end for**

5: $C_{min}^{(i)} = LowerBound(c_j, \mathcal{K}, \mathcal{G}(i,j), b_i)$

6: $T = \frac{\epsilon \times C_{min}^{(i)}}{2}$; $J = \frac{\epsilon^2 \times C_{min}^{(i)}}{4}$;

7: **for** j = 1 to N **do**

8: **if** $c_j \geq T$ **then**

9: $j \rightarrow \Psi$;

10: **else**

11: $j \rightarrow \psi$;

12: **end if**

13: **end for**

14: $\mathcal{T} = LargeDP(b_i, \mathcal{C}, \mathcal{K}, T, J, \Psi, \mathcal{G})$;

15: $C_{opt}^{(i)} = SmallGreedy(b_i, \mathcal{G}(i,j), c_j, \mathcal{T}, \psi)$;

16: **end for**

the cost for w concurrent users from the SmallGreedy() for application i. We let $C^{(i)}$ denote cost calculated by MPS-FPTAS() of application i. We have the equation as Eq. 5.5.

$$C^{(i)} = \min_{(Q,W,C)\in\mathcal{T}}\{JQ + \phi(K - W)\} \tag{5.5}$$

Algorithm 5.3.3 LowerBound()

Input: $\mathcal{C}$, $\mathcal{K}$, $\mathcal{G}$, b_i.

Output: $C_{min}^{(i)}$

1: Initialization:

2: **for** $j = 1$ to $\mathcal{N}$ **do**

3: $\rho_j = \frac{\mathcal{G}(i,j)}{c_j}$;

4: **end for**

5: Select j' where $\rho_{j'}$ is the largest among ρ_j and $\mathcal{G}(i, j') \leq 2\mathcal{K}$;

6: $C_{min}^{(i)} = \frac{1}{2}\lceil \frac{\mathcal{K}}{\rho_{j'}} \rceil \times c_{j'}$;

For the optimal result C_i^*, we suppose the $C_L^{(i)}$ is the cost from Ψ and the associated supported number of concurrent users is $W_L^{(i)}$ and the scaled price $Q_L^{(i)}$, and $C_S^{(i)}$ is the cost from the ψ and the associated supported number of concurrent users is $W_S^{(i)}$. As Q and W are the optimal solution from the LargeDP() of MPS-FPTAS(), we have $Q \leq Q_L^{(i)}$ and $W \geq W_L^{(i)}$. Therefore, we have Eq. 5.6.

$$C^{(i)} \leq JQ + \phi(K - W) \leq JQ_L^{(i)} + \phi(K - W). \tag{5.6}$$

Besides that, we can have Eq. 5.7

$$C_i^* = C_L^{(i)} + C_S^{(i)} > J(Q_L^{(i)} - \frac{C_i^*}{T}) + C_S^{(i)} \tag{5.7}$$

From Eq. 5.6 and Eq. 5.7, we have

$$\begin{aligned} C^{(i)} - C_i^* &\leq JQ_L^{(i)} + \phi(K - W) - J(Q_L^{(i)} - \tfrac{C_i^*}{T}) - C_S^{(i)} \\ &= \phi(K - W) - C_S^{(i)} + \tfrac{J}{T}C_i^* \end{aligned} \tag{5.8}$$

Algorithm 5.3.4 LargeDP()

Input: b_i, $\mathcal{C}$, $\mathcal{K}$, T, J, Ψ, $\mathcal{G}$.

Output: Table $\mathcal{T}$.

1: Initialization:

2: **for** $j = 1$ to $|\Psi|$ **do**

3: $k = \lfloor \log_2 \frac{c_j}{T} \rfloor$, $q_j = \lfloor \frac{c_j}{2^k J} \rfloor \times 2^k$;

4: $\rho_j = \frac{\mathcal{G}(i,j)}{c_j}$;

5: **end for**

6: $\mathcal{T} \leftarrow \{0, 0, 0\}$;

7: **while** drop in q_j **do**

8: For each q_j, select j' with largest ρ_j

9: $n = \lceil \frac{4}{\epsilon} \rceil$;

10: **for** $l = 1$ to $\log_2 \lceil \frac{4}{\epsilon} \rceil$ **do**

11: $q_{j'}^{(l)} = 2^l q_{j'}$, $\mathcal{G}_{j'}^{(l)} = 2^l \mathcal{G}(i, j')$

12: **for** each Q' in Table $\mathcal{T}$ **do**

13: **if** $W(Q' + q_{j'}^{(l)}) < W(Q') + \mathcal{G}_{j'}^{(l)}$ **then**

14: $W(Q' + q_{j'}^{(l)}) = W(Q') + \mathcal{G}_{j'}^{(l)}$;

15: Update $\mathcal{T} \leftarrow \{Q' + q_{j'}^{(l)}, W(Q' + q_{j'}^{(l)}), c(Q' + q_{j'}^{(l)})\}$;

16: **end if**

17: **end for**

18: **end for**

19: **end while**

Because $K - W \leq K - W_L^{(i)}$, and if the $SmallGreedy()$ exactly fills all the $(K - W)$ concurrent session space,we have $\phi(K - W) \leq C_S^{(i)}$. We further have $\phi(K - W) - T < C_S^{(i)}$.

Algorithm 5.3.5 SmallGreedy()

Input: b_i, $\mathcal{G}$, $\mathcal{C}$, $\mathcal{T}$, ψ. **Output:** the optimal solution $C_{opt}^{(i)}$,

1: Initialization:

2: **for** $j = 1$ to $|\psi|$ **do**

3: $\rho_j = \frac{\mathcal{G}(i,j)}{c_j}$;

4: **end for**

5: Select j' where $\rho_{j'}$ is the largest among ρ_j;

6: $C_{opt}^{(i)} = \min_{\{Q,W(Q),C(Q)\}\in\mathcal{T}}\{C(Q) + \max\{0, \frac{\mathcal{K}-W(Q)}{\rho_{j'}}\}\}$;

As a result, we have

$$\begin{aligned} & C^{(i)} - C_i^* \leq \phi(K - W) - C_S^{(i)} + \tfrac{J}{T}C_i^* \\ & = T + \tfrac{J}{T}C_i^* \leq \epsilon C_i^* \\ & \Longrightarrow C^{(i)} \leq (1+\epsilon)C_i^* \end{aligned} \tag{5.9}$$

Therefore, for all applications, we have $C \leq (1+\epsilon)C^*$. □

Theorem 5.3.2. *The complexity of algorithm MPS-FPTAS() is* $O(M(\mathcal{N} + \frac{1}{\epsilon^4}))$

Proof. For each application i, the time complexity of algorithm MPS-FPTAS() is determined by $LargeDP()$ and $SmallGreedy()$.

$LargeDP()$ is a process of dynamic programming, which can be divided into different iterations. In each iterations, the service instances with the scaled profit q_j are considered for the final solution. Therefore, the time complexity of $LargeDP()$ is determined by the size of table $\mathcal{T}$, which is bounded by the size of the scaled profit space, and the maximal *copies of service instances* for computation (e.g. we consider using one service instance

j as one copy of service instance, and consider using one and two service instances j, as two copies of service instances). We let the size of table $\mathcal{T}$ be H_j. As it is bounded by the scaled profit space, $H_j = \lfloor \frac{C_i^*}{2^s J} \rfloor \times 2^s \leq \frac{C_i^*}{J} \leq \frac{8}{\epsilon^2}$ (where $s = \lfloor \log_2 \frac{C_i^*}{T} \rfloor$). We let the maximal copies of service instances used for $LargeDP()$ be h_j. Obviously, the copies of service instances are determined by the number of of q_j values and the number of copies for each q_j values. The number of q_j values obtained from c_j interval $(2^k T, 2^{k+1} T]$(here $k = \lfloor \log_2 \frac{c_j}{T} \rfloor$) is at most $\frac{2}{\epsilon} - 1$, independent of k. The number of copies in Ψ priced q_j can be contained in feasible solution is at most $\lfloor \frac{C_i^*}{J \times q_j} \rfloor = \frac{3}{2^k \times \epsilon}$. Therefore, we have the total number of copies of service instances considered for $LargeDP()$ is bounded by $h_j < \frac{2}{\epsilon}(\frac{3}{\epsilon} + \frac{3}{2 \times \epsilon} + ... + \frac{3}{2^k \times \epsilon}, ...) = \frac{6}{\epsilon^2}(1 + \frac{1}{2} + ... \frac{1}{2^k}, ...) < \frac{12}{\epsilon^2}$. Finally, we have the time complexity of $LargeDP()$ is $O(H_j \times h_j) = O(\frac{96}{\epsilon^4})$.

We can easily have the time complexity of SmallGreedy() is $O(N)$. Then for each application i, the time complexity is $O(\frac{96}{\epsilon^4} + N)$. For M applications, the time complexity if $O(M(\mathcal{N} + \frac{1}{\epsilon^4}))$. □

5.3.3 A Greedy Algorithm for MPS

Our approximation algorithm provides a bound for the MPS problem. It may have a large running time if ϵ is small. As such, we introduce a fast greedy algorithm.

The details of our greedy algorithm are in Algorithm 5.3.6 and we name it as MPS-Greedy(). The essence is that for each application i, we recursively pick up the service instance j with the highest 'session number-price' ratio $\rho_{ij} = \frac{\mathcal{G}(i,j)}{c_j}$ to fill the unsolved concurrent user space, until all the concurrent user numbers of application i are satisfied. The greedy algorithm solves a typical MPS problem in the time complexity (the choice part, without the sorting part) of $O(MN)$, where N is the total number of service instances and M is the number of applications.

Algorithm 5.3.6 MPS-Greedy()
Input: $\mathcal{C}$, R, $\mathcal{K}$, Θ, A

Output: for each application requirement b_i, the optimal solution $C_{opt}^{(i)}$,

1: **for** i = 1 to M **do**
2: **for** j = 1 to N **do**
3: initiate $x_{ij} = 0$;
4: Calculate $\mathcal{G}(i,j) = Max\{Min_k \lfloor \frac{r_k - a_{ij}^{(k)}}{\theta_{ij}^{(k)}} \rfloor, 0\}$
5: $\rho_{ij} = \frac{\mathcal{G}(i,j)}{c_j}$
6: **end for**
7: $n_i' = 0, C_{opt}^{(i)} = 0$;
8: **while** $n_i' < \mathcal{K}$ **do**
9: choose j with largest value of ρ_{ij};
10: $x_{ij} = Round(\frac{n_i - n_i'}{\mathcal{G}(i,j)})$;
11: $n_j' = n_j' + x_{ij} \times \mathcal{G}(i,j)$;
12: $C_{opt}^{(i)} = C_{opt}^{(i)} + c_j x_{ij}$;
13: remove j from the service instance of
14: application requirement i;
15: **end while**
16: **end for**

5.3.4 The Problem with More Constrains

We then develop two algorithms for MPS$_S$ and MPS$_C$.

1) The MPS$_S$ problem

We can see that the differences between MPS$_S$ and MPS lie in the wholesales provided by some IaaS vendors. Here, suppose in the MPS$_S$ problem, there are N' wholesales

Algorithm 5.3.7 MPS-S()
Input: $\mathcal{C}$, R, $\mathcal{K}$, Θ, A, $\mathcal{B}$, Ω

Output: $C_{opt}^{(i)}$ for each $i = 1, 2, ...M$

1: $\{R^w, \Theta^w, A^w, \mathcal{G}^w\} = KnapSackForSalePacket(\Omega, \mathcal{B})$;

2: $\{C_{min}^1, C_{min}^2, ..., C_{min}^M\} = algorithm(\mathcal{C} \cup \mathcal{C}^w, R \cup R^W, \mathcal{K}, \Theta \cup \Theta^W, A \cup A^W, \mathcal{B})$;

in all service providers. Here we let j' denote the sequence number of wholesales and let $\mathcal{G}^w(i, j')$ denote maximal number of concurrent users allowed by the wholesale j', and let $\mathcal{C}_{j'}^w$ denote the cost of wholesale j'. Similarly, we also define $\Theta_{ij'}^w$ as the amount of resource consumed by one user of application i in wholesale j', and $A_{ij'}^w$ be basic resource consumed by application i in wholesale j'.

Therefore, for each application i, we divide the procedure of solving the MPS$_S$ problems into the following two steps:

- firstly, for each wholesale j' we calculate a combination of service instances, and this combination should have a maximal value of $\mathcal{G}^w(i, j')$. According to this combination, we have $\Theta_{ij'}^w$ and $A_{ij'}^w$ for each wholesale j';
- secondly, since we have $\Theta_{ij'}^w$ and $A_{ij'}^w$ for each wholesale j', we solve the problem using $algorithm(\mathcal{C} \cup \mathcal{C}^w, R \cup R^w, \mathcal{K}, \Theta \cup \Theta^w, A \cup A^w, \mathcal{B})$, where $\mathcal{C}^w = \{c_1^w, c_2^w, ..., c_{N'}^w\}$, $R^w = \{r_1^w, r_2^w, ..., r_{N'}^w\}$, Θ^w is comprised by $\Theta_{ij'}^w$, A^w is comprised by $A_{ij'}^w$, and $r_{j'}^w$ is the j'th set of wholesale resource (here $i = 1, 2, ..., M$ and $j' = 1, 2, ..., N'$). Here algorithm() can be chosen from MPS-FPTAS(), MPS-Greedy() or MPS-Special().

We present the algorithm for MPS$_S$ in Algorithm 5.3.7. The input variables of MPS-S() are $\mathcal{C}$, R, $\mathcal{K}$, Θ, A, $\mathcal{B}$ and Ω. Ω denotes the set of wholesales provided by all IaaS vendors, and $\Omega = \{\Omega_1, \Omega_2, ..., \Omega_{\mathcal{L}}\}$. Here Ω_l denotes the wholesales provided by IaaS vendor l and $\Omega_l = \{R_l, C_l, C_l^w, R_l^w\}$. Here, the set R_l denotes the set of the resource of the service instances in IaaS vendor l, C_l denotes the set of the service instance prices in vendor l. Besides that, C_l^w is the price set of the wholesales provided by vendor l, and R_l^w is the set of wholesale resource of all service packets of vendor l.

Function KnapSackForSalePacket(Ω) calculates the combinations of service instances allowed largest number of concurrent users for all wholesales. In our case, we use Knapsack algorithms to find the combination of service instances, which support the largest number of concurrent users, under the total wholesale resource constrain. As the Knapsack problem is widely studied, such as in the book [54], the algorithm KnapSackForSalePacket(Ω) is not discussed here. In our thesis, we use greedy algorithm for Knapsack problem to solve KnapSackForSalePacket(Ω).

The complexity of the algorithm for MPS$_S$ problem lies in the function algorithm(). Typically, if we choose MPS-FPTAS(), the complexity of MPS-S() is $O(M(\mathcal{N}+\mathcal{N}'+\frac{1}{\epsilon^4}))$.

2) The MPS$_C$ problem

Without loss of generality, we assume the application requirements indexed from 1 to M_1 are independent. We assume there are ω groups of applications $m_{ij_0}, m_{ij_1}, \ldots,$ where $i \in [0, \omega-1]$. Each group of applications must use service instances from the same service provider. To solve the problem MPS$_C$ as a subrouting of MPS problem, we follow the steps as follows: 1) we solve the costs for independent application requirement i indexed from 1 to M_1 by the algorithms for MPS problem; 2) For each group of dependent applications i, we solve the problem recursively solve algorithm $(\mathcal{C}^{(l)}, R^{(l)}, \mathcal{K}, \Theta^{(il)}, A^{(il)}, \mathcal{B}_i^w)$ from $l=1$ to $\mathcal{L}$, and choose the one with the lowest cost for dependent application group i. Here $\mathcal{C}^{(l)}$ is the set of the price of all the service instances from vendor l, $R^{(l)}$ is the set of the resource of all the service instances in vendor l, $\Theta^{(il)}$ denotes the set of resource consumption from each application i user in each service instance in vendor l, $A^{(il)}$ stands for the basic resource consumption set of application i of each service instance in vendor l, and $\mathcal{B}_i^w$ is the ith group of constrained applications. We present the algorithm for MPS$_C$ in Algorithm 5.3.8.

The complexity of MPS$_C$ is also bounded by the algorithm complexity selected for function algorithm(). Typically, we suppose the largest service instance number in one vendor is N_l', and we use MPS-FPTAS() for algorithm(), the time complexity of solving MPS$_C$ is $O(M_1(N+\frac{1}{\epsilon^4})+\omega\mathcal{L}(N_l'+\frac{1}{\epsilon^4}))$.

Algorithm 5.3.8 MPS-C()

Input: $\mathcal{C}$, R, $\mathcal{K}$, Θ, A, $\mathcal{B}$.

Output: $C_{opt}^{(i)}$ for each $i = 1, 2, ...M$.

1: **for** i = 1 to M_1 **do**

2: $C_{opt}^{(i)} = algorithm(\mathcal{C}, R, \mathcal{K}, \Theta^i, A^i, b_i)$;

3: **end for**

4: **for** i = 1 to ω **do**

5: min_cost = $+\infty$;

6: **for** l = 1 to $\mathcal{L}$ **do**

7: $\{C_{i1}^l, C_{i2}^l, ...C_{iN_l}^l\} = algorithm(\mathcal{C}^{(l)}, R^{(l)}, \mathcal{K},$

8: $\Theta^{(il)}, A^{(il)}, \mathcal{B}_i^w)$;

9: **if** $\Sigma_{j=1}^{N_l} C_{ij}^l <$min_cost **then**

10: $\{C_{min}^{i1}, C_{min}^{i2}, ...C_{min}^{iN_l}\} = \{C_{i1}^l, C_{i2}^l, ...C_{iN_l}^l\}$

11: min_cost = $\Sigma_{j=1}^{N_l} C_{ij}^l$;

12: **end if**

13: **end for**

14: **end for**

5.4 Performance Evaluation

5.4.1 Simulation Setup

In this section, we compare the costs between multiple provider strategy and single vendor strategy for MPS, MPS$_C$ and MPS$_S$ problems with the following algorithms.

- Greedy
- FPTAS
- Single vendor best

We consider vendors in three different scales: large, medium and small. We suppose the large scale vendors have 50 service instances, the medium scale vendors have 20 service instances, and the small scale ones only have ten. In our simulation, to reduce the simulation in each step, we construct the IaaS vendors as follows: we add the number of vendors ten by ten, and in each added ten vendors, there are two large scale vendors, five medium scale ones, and three small scale ones.

Each vendor may also have service instances in different *service series*. Here, we define a *service series* as a number of service instances with a resource comparison rate. For example, all service instances with the resource comparison rate r_1:r_2:r_3 = 1:2:3 belong to the same service series, while the service instances with comparison rate r_1:r_2:r_3 = 3:2:1 belong to another.

We construct the service instances of IaaS providers as follows. We simulate both one-vendor-one-service series situation and one-vendor-multi-service series situation. For IaaS providers, we randomly generate the service series, and then create different numbers of service instances accordingly.

We simulate that the number of types of resources in each service instance is 10 in our simulation. Each type of resource o has a scope of number ranging from $[L_o, H_o]$. In our simulation, we randomly let the scope of the five types of resource be $[1, 1000]$, three of them range from $[1, 5000]$, and two of them within the domain of $[1, 8000]$.

We assign the price of service instance j as Eq. 5.10. We let the price of the service instance j linear to the number of each type of resource i with a constant coefficient γ_i. To reasonably increase the randomness of the price, we suppose the prices of service instances are with a random number δ. This pricing regulation is widely existed in the IaaS service providers such as Amazon EC2 [112]. We randomly choose γ_i from the integer ranging from 1 to 20, and we intuitively let δ follow the distribution $N(0, 2)$.

$$c_j = \sum_{i=1}^{o} \gamma_i \times r_i + \delta. \tag{5.10}$$

To acquire the knowledge of the appropriate circumstances for different MPS solution algorithms, we vary the parameters as follows: the number of IaaS vendors, the number of applications, as well as the number of required concurrent users. If it is not specified, we by default set the number of IaaS vendors as 30, the number of applications as 20, the number of types of resource as ten, and the number of required concurrent users as 2000.

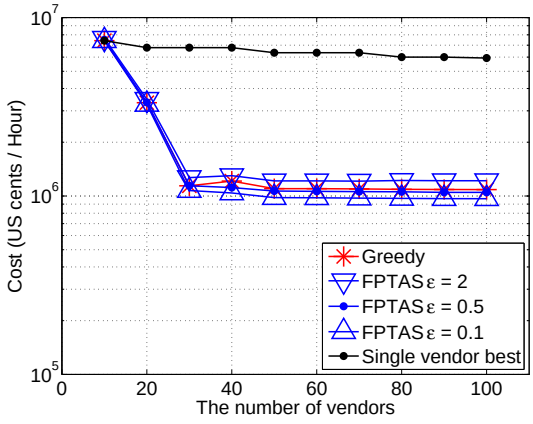
In our simulation, we suppose that, for each application, the resource consumption in each service instance per user is the same. That is, $\theta_{ij}^{(o)} = \theta_i^{(o)}$ and $a_{ij}^{(o)} = a_i^{(o)}$ for all $j = 1, 2, ..., N$. We generate the resource requirement of each applications according to the following rules:

- We randomly choose $\theta_i^{(o)}$, which denotes the number of resource o consumed by each user from application i, from the range of $[0, 0.9]$ times of the lowest bound of resource o, L_o.
- We assume that in our simulation, different applications require different amount of resource for each user. That is, for $i_1, i_2 = 1, 2, ..., M$, if $i_1 \neq i_2$, $\theta_{i_1}^{(o)} \neq \theta_{i_2}^{(o)}$;
- We randomly choose $a_{ij}^{(o)}$ $[1, 10]$ times of $\theta_i^{(o)}$. Admittedly, the upper bound of the random number can be freely given. We set them respectively to be 0.9 and 10, in order to protect the resource requirement of applications being not able to be fulfilled by any service instances.

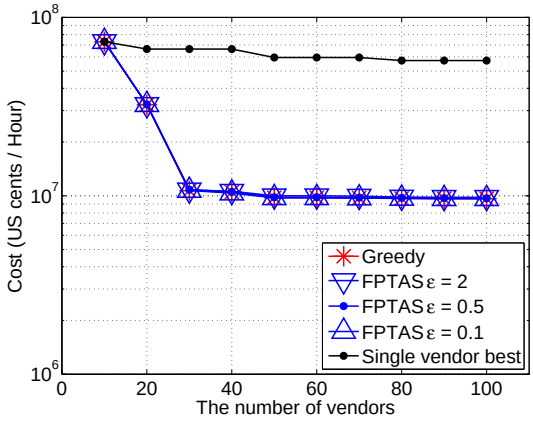
5.4.2 Simulation Results

1) MPS problems: We consider the situations that there are respectively 200, 2000, and 10000 required concurrent users. In each situation, we initially let the number of IaaS vendors be ten, and each time we add ten vendors. Fig. 5.1 presents the comparison of the costs of different algorithms.

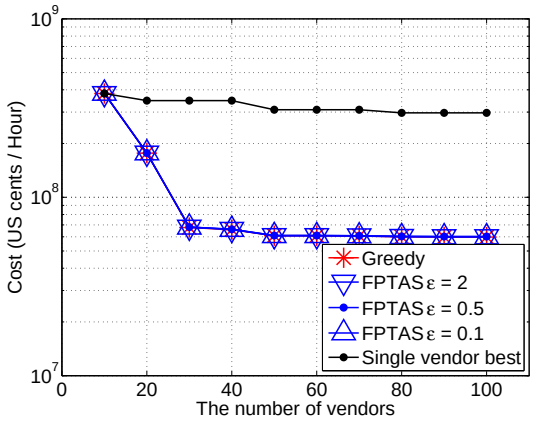
Specifically, Fig. 5.1 (a) presents the situation of providing services for 200 concurrent users. We can see that compared with single vendor strategy, multiple provider strategy



(a) 200 concurrent users



(b) 2000 concurrent users



(c) 10000 concurrent users

Figure 5.1. The costs of hosting the servers, ranging by different number of vendors

can save the costs of hosting the service by roughly 80.0% (from 5.9×10^6 to 1.1×10^6). In detail, we can see the first 30 vendors in multiple provider strategy greatly lower cost of hosting the service by about 85.4% (from 7.4×10^6 to 1.1×10^6). Compared with that, we can see that the price of constructing the cloud service to the vendor number in the single vendor strategy is not as sensitive as that in multiple provider strategy. Compared with only one vendor, 100 vendors only reduce the cost by about 20% (from 7.4×10^6 to 5.9×10^6). This trend is also similar in Fig. 5.1 (b) and Fig. 5.1 (c) while the required concurrent numbers of users are respectively 2000 and 10000.

While comparing performance of greedy algorithm and the FPTAS algorithm, we can see a quite different trend while comparing Fig. 5.1 (a) with Fig. 5.1 (b) and Fig. 5.1 (c). In Fig. 5.1 (a), we can see that if vendor number is 200, FPTAS $\epsilon = 0.1$ enjoy overwhelming advantages compared with greedy algorithm. While the number of vendors is 100, we can see that the approximation algorithm $\epsilon = 0.1$ can save 11.4% costs from the greedy algorithm (from 1086221.4 to 965182.). Compared with that, the FPTAS in the situation with 2000 or 10000 concurrent users do not have such large advantage and the saved cost is below 0.1%. The FPTAS $\epsilon = 2$ and 0.5 even perform worse than the greedy algorithm.

To further analyze cost saving of the multiple provider strategy, we consider multi-service series per vendor situation. We extend each IaaS vendor with more than one service series. We define *percentage of saved cost* of the multiple provider strategy in Eq. 5.11. Fig. 5.2 presents a percentage of saved costs of the multiple provider strategy as a function of the number of service series per vendor. It is observed that the larger number of service series in one vendor, the less percentage of saved cost comes from the multiple provider strategy.

$$percentage\ of\ saved\ cost = \frac{costs\ of\ single\ vendor}{costs\ of\ multiple\ providers} - 1 \tag{5.11}$$

To provide a detailed analysis on the differences between FPTAS algorithm and greedy algorithm, we compare greedy algorithm and the FPTAS $\epsilon = 0.1$ in Fig. 5.3. Fig. 5.3 (a) presents the differences of the costs generated by greedy and FPTAS $\epsilon = 0.1$. We can see

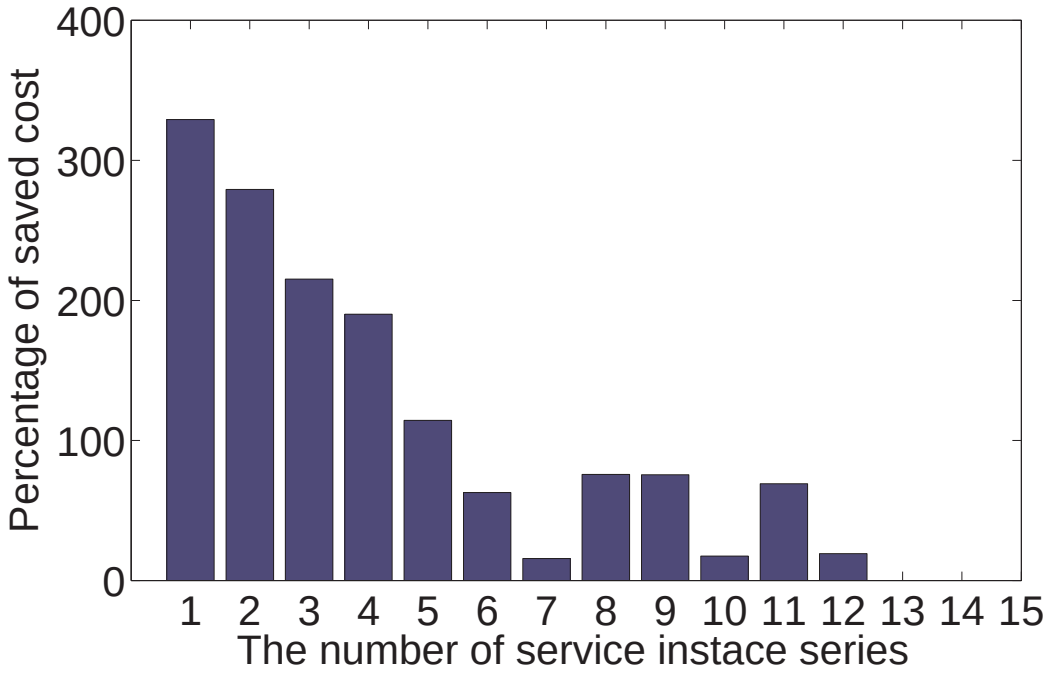
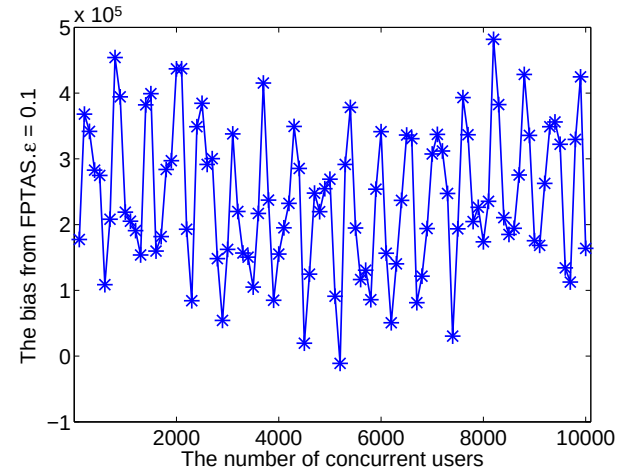


Figure 5.2. The percentage of saved cost of multiple provider strategy

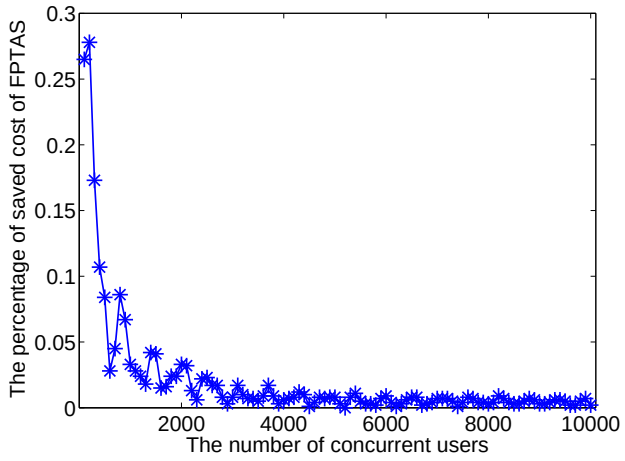
that no matter how large the number of required concurrent users is, the difference between the costs fluctuates around 3×10^5 US cents. However, the total expenditures increase as the number of required concurrent users increases. Fig. 5.3 (b) shows the percentage of the costs saved by FPTAS $\epsilon = 0.1$ from greedy. We can see the advantage of the FPTAS algorithm decreases and near zero when the required concurrent user number is 2000.

We also assess the computation overhead of the FPTAS algorithm with different value of ϵ for the 30 vendors and 2000 concurrent users. We show the computation cycles as a function of ϵ in Fig. 5.4. We can see that when $\epsilon = 0.01$, the computation cycles are over 60 seconds, even if $\epsilon = 0.2$, the computation complexity is also over 2.2 seconds. Compared with the greedy algorithm with a time complexity of $O(N)$, the FPTAS algorithm is with a much higher time complexity.

As a result, we summarize that, we may choose FPTAS algorithm for the situation that concurrent user is small (such as below 1000 concurrent users), since the advantage of FPTAS over the greedy algorithm is obvious. However, while the number of concurrent



(a)



(b)

Figure 5.3. The comparison of the greedy algorithm and the FPTAS algorithm: (a) The differences between the greedy and appr. $\epsilon = 0.1$ costs; (b) The percentage of the costs saved by appr. $\epsilon = 0.1$ from greedy

users is high, we may use greedy algorithm, since the advantage of FPTAS is minor and its computation complexity is much higher.

Besides that, we also show the costs of hosting services with single vendor strategy and multi-provider strategy as a function of the number of resource types and the number of applications in Fig. 5.5. In Fig. 5.5 (a), we can observe that the costs increase with the number of resource types. Nevertheless, it is also obvious that the profit space from multiple

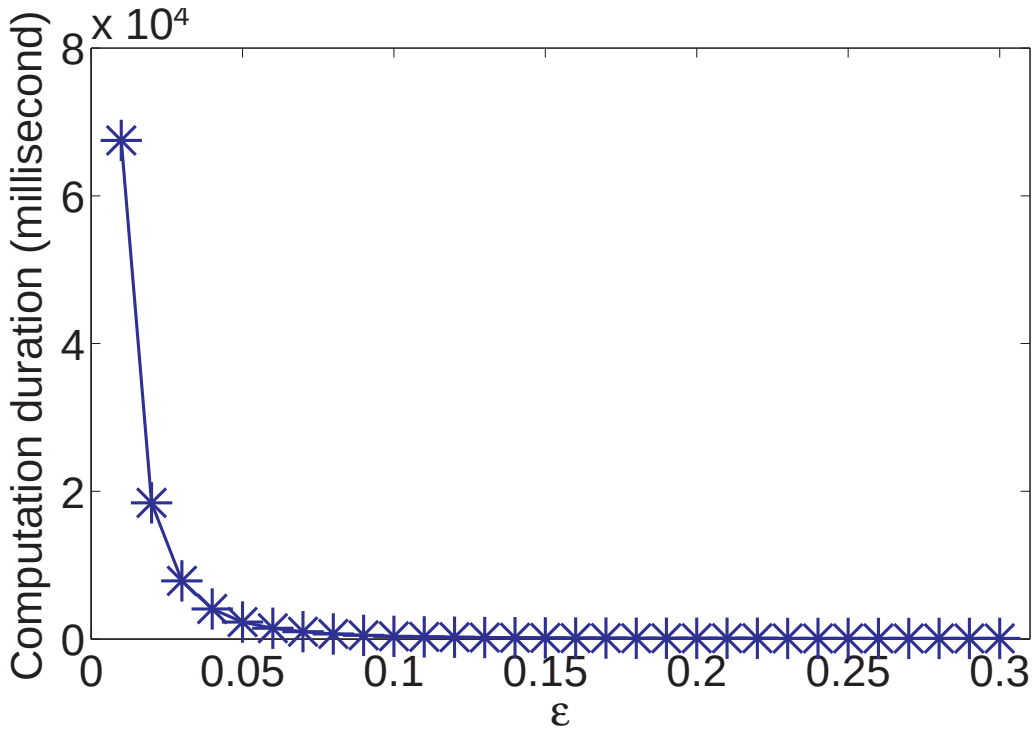
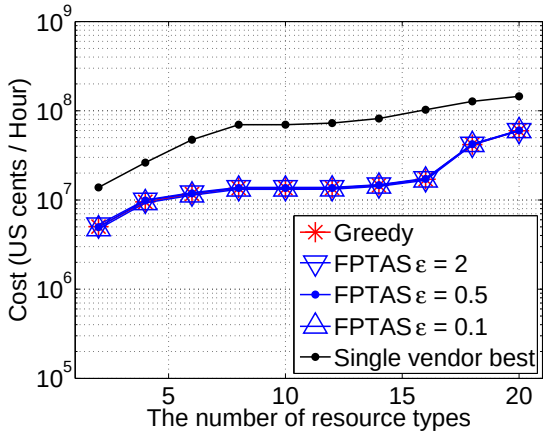


Figure 5.4. The computation cycle of the approximation algorithm

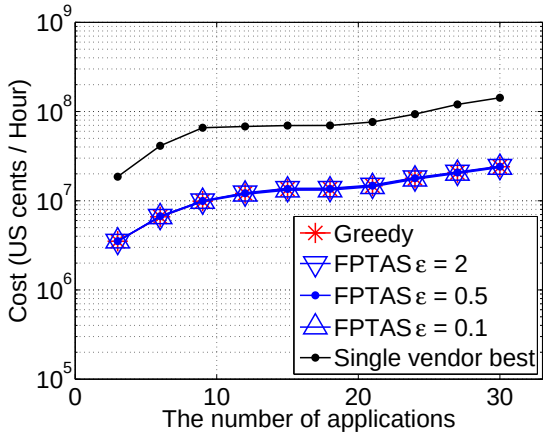
provider strategy is not affected by the number of resource types. While of the number of the resource types ranges from 4 to 20, the percentage of saved cost of the multiple provider strategy also fluctuates from 1.8 to 4.0. The same observation is also apparent in Fig. 5.5 (b), we show that the number of applications increases the total hosting costs, but the profit bias of the multi provider strategy is not affected by the number of applications.

2) MPS$_C$: Here we consider the MPS$_C$ problem. We use the simulation setup as it is introduced in the simulation of MPS. Fig. 5.6 compares the costs of hosting services in the cloud servers between the single vendor strategy and the multi provider strategy as a function of the number of constrained applications. Here, the number of constrained applications refers to the number of applications whose servers should come from the same vendor. We only consider two situations, with the number of required concurrent users respectively 200 and 2000.

Fig. 5.6 (a) presents the comparison of costs in the 200 concurrent user situation. We



(a) The cost comparison as a function of resource types

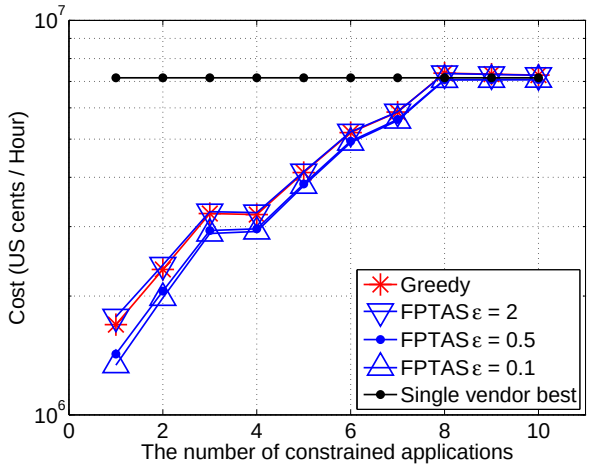


(b) The cost comparison as a function of requirement number

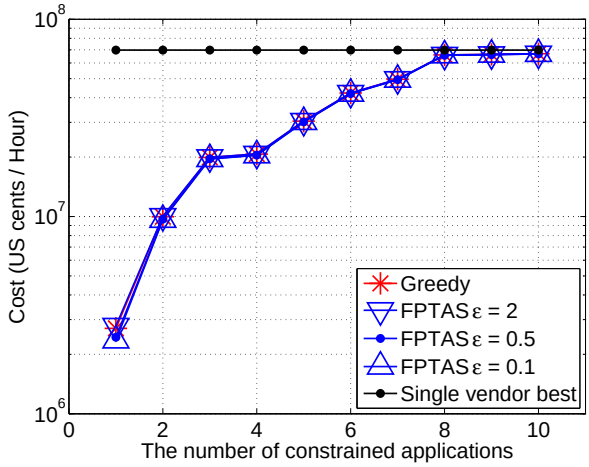
Figure 5.5. The cost comparison as a function of resource types and requirement number

can see that as the number of constrained applications increases, the benefit from the multi provider strategy decreases. In detail, while the number of constrained applications is eight, no benefit is from the multiple provider strategy. This is not surprising that as the more number of applications are constrained, the result is more close to the single vendor strategy. Similar results also can be observed in Fig. 5.6 (b), where there are 2000 concurrent users.

3) MPS$_S$: We consider the different numbers of IaaS vendors with wholesale strategy. Here, we construct the wholesale strategy in our simulation similar with the strategy in GoGrid [114] as follows: we let the resource of the most expensive service instance in one



(a) 200 concurrent users

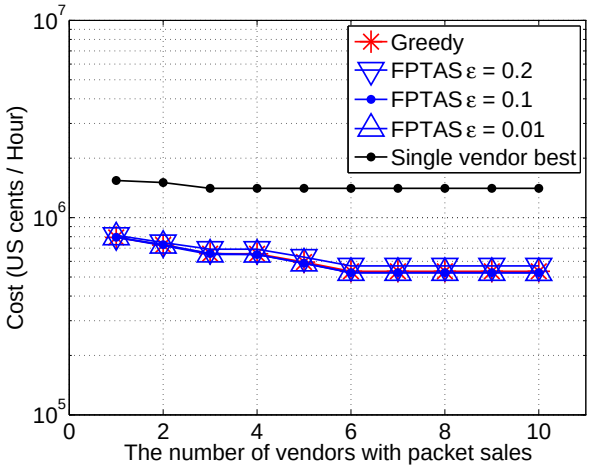


(b) 2000 concurrent users

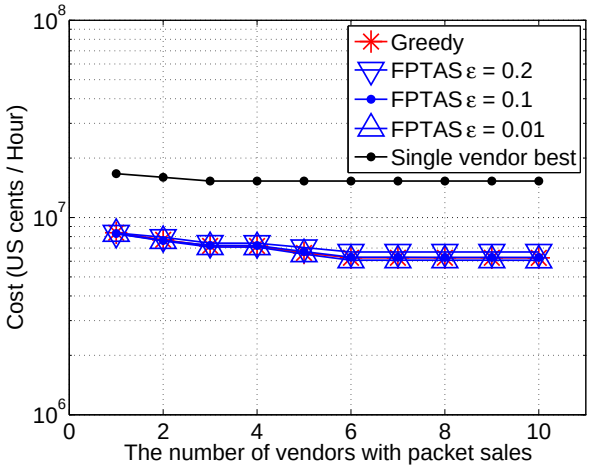
Figure 5.6. The costs of hosting the servers as a function of constrained applications

vendor be τ_0^w and its price be c_0^w. For the ith ($i \geq 1$) wholesale in one vendor, its price and resource amount are respectively as $\tau_j^w = 4 \times \tau_{j-1}^w$ and $c_j^w = 3.2 \times c_{j-1}^w$.

Fig. 5.7 shows the comparison of the multiple provider strategies and the single vendor strategy for MPS$_S$. Fig. 5.7 (a) shows the cost comparison as a function of the number of vendors with wholesale strategy while the required concurrent user number is 200. We can see that the saved costs of the multiple provider strategy increase. This is not surprising that the more vendors provide wholesales, the less advantage from the vendors with wholesale



(a) 200 concurrent users



(b) 2000 concurrent users

Figure 5.7. The costs of hosting the servers, choosing from small vendors, medium vendors to large vendors

strategy. We can also observe that the same phenomenon while the concurrent user number is 2000 in Fig. 5.7 (b). This indicates that compared with our wholesale strategy with 0.8 price discount, the multiple provider strategy can further reduce the costs.

5.5 Case Study: A Web Game Application Experiment

We deploy an open source web game, PHP MMORPG [120] on the cloud servers. We choose service instances from three world-wide IaaS vendors, Amazon EC2, GoGrid and Rackspace. The specification of the service instances are as [112], [114] and [118]. Here, we focus on three types of resource in service instances: CPU, RAM, and storage size.

In the MMORPG game, the players are able to walk in the map, chat with each other and fight against the monsters and other players. A deployed example of this game is in [121]. We divide the application into three sub-applications and we choose the service instances from the three IaaS providers for these three sub-applications. The three sub-applications are as follows.

1. Walk sub-application: Walk sub-application responses to the 'walk' command from the players, which changes the location of the avatars.

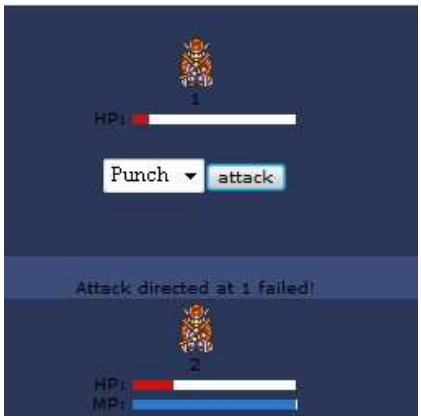
2. Chat sub-application: Chat sub-application manages the messages among the players.

3. PVP sub-application: This sub-application regulates the fighting behaviors among the users and between the users and the monsters. It only update every five seconds. These three sub-applications are portrayed in Fig. 5.8.

To construct the solution for this web game application, we firstly need to profile the resource consumption on the servers. That is, for each application b_i and service instance τ_j, we profile the value of A_{ij} and Θ_{ij}. We initiate service instances in servers in the three vendors as follows: all of the service instances we bought from these two vendors are installed with Ubuntu 10.04, with web server Tomcat 6.00 and database MySQL 5.10. After the PHP MMORPG is deployed on the service instances, we perform resource consumption profiling. To make the software environment configuration of the service instances in one vendor identical, after one service instance in one vendor is profiled, we rebuild the service instance into another service instance without the software reinstalled, and do the profiling again, instead of buying a new service instance and re-configure the new server.

(a)

(b)



(c)

Figure 5.8. The simulation in the web game. (a) Walk sub-application: ten robots random walk in the game; (b) Chat sub-application: ten robots random chat in the game; (c) PVP sub-application

To make our resource consumption profiling more uniform, we develop a game robot to emulate the behaviors of players in the game. According to the three sub-applications, our game robot contains three modules that respectively emulate the walk, chat, and fighting behaviors from the users. Each time, we use exactly one module to profile the resource consumption of one sub-application. In detail, we program the modules of the game robot as follows.

- To profile the resource consumption for walk sub-application, we develop the module that creates any number of avatars, which only walk randomly in the maps of the game.
- To profile for the chat sub-application, we program the module that controls any number of avatars only chatting in the game.
- For the pvp sub-application, the module generates the avatars fighting with each other only. However, according to the JavaScript codes in the web page, players can only post one action every five seconds while fighting, so as the game robot.

For Eq. 5.1, we do the resource consumption profiling as follows: after we rebuild the cloud servers, we restart them to release the extra memory expenditure causing the inaccurate measurement. We profile the base resource consumption A_{ij} when there is no users on the cloud servers. Each time we add 10 concurrent users to service instances, and then record the resource consumption to estimate the resource consumption per user Θ_{ij}. To assure the service capacity, we record the peak consumption value for A_{ij} and Θ_{ij} for each resource.

After profiling the resource consumption, we merge basic resource consumption A_{ij} and the resource consumption per user Θ_{ij} of all service instances in one vendor into the same vector for the following reason:

1. The memory and storage consumption of A_{ij} and Θ_{ij} are the same for all service instances in one vendor. This is because the software environment is the same and the RAM and the storage space are identical in the same vendor.

Vendor	Base resource consumption A_{ij}	Resource consumption per user Θ_{ij}
EC2	{0.24%, 0.194GB, 6.4GB }	{2.71%, 0.0191GB, 0.5GB}
GoGrid	{0.01%, 0.177GB, 15GB }	{0.79%, 0.018GB, 0.5GB}
Rackspace	{0.02%, 0.179GB, 6.0GB }	{0.12%, 0.019GB, 0.5GB}

Table 5.1. The resource consumption of the walk sub-application in the service instances in different vendors

2. The consumption of CPU resource can only be measured by percentage of CPU usage. The CPU resource allocation schemes are different from one to another. Rackspace allocates CPU resource in Linux servers by CPU cycles [117]. Obviously, the CPU performance of the service instance with $nx\%$ total CPU cycles is n times larger than the one with $x\%$ total CPU cycles. For GoGrid and EC2, the CPUs of service instances are allocated in the form of the same types of virtual cores [112] [114]. According to the study in [4], the performance of n cores are approximately n times of a single core. The CPU resource consumption in GoGrid and EC2 is recorded by the percentage of CPU usages of the single virtual core CPU.

As a result, we have the profiling results of the walk sub-application, chat sub-application and PVP sub-application as Table 5.1, Table 5.2 and Table 5.3.

We compare the costs from different algorithms for multiple provider strategy and the single vendor strategy. According to the pricing strategy of GoGrid [114], GoGrid provides four different scales of wholesales in different scales, Fig. 5.9 shows the cost comparison between the multiple provider strategy and single vendor strategy with the wholesale strategy of GoGrid. In Fig. 5.9 (a), we initially choose the required concurrent user number 200, and each time we increase the number by 200 until the total required user number is 2000. For only three applications, we can still see that the improvement of multiple provider strategy is stable and substantial for all concurrent user numbers. Generally, the multiple provider

Vendor	Base resource consumption A_{ij}	Resource consumption per user Θ_{ij}
EC2	{0.24%, 0.194GB, 6.4GB }	{2.10%, 0.0101GB, 0.5GB}
GoGrid	{0.1%, 0.177GB, 15GB }	{0.64%, 0.0104GB, 0.5GB}
Rackspace	{0.1%, 0.179GB, 6.0GB }	{0.09%, 0.0109GB, 0.5GB}

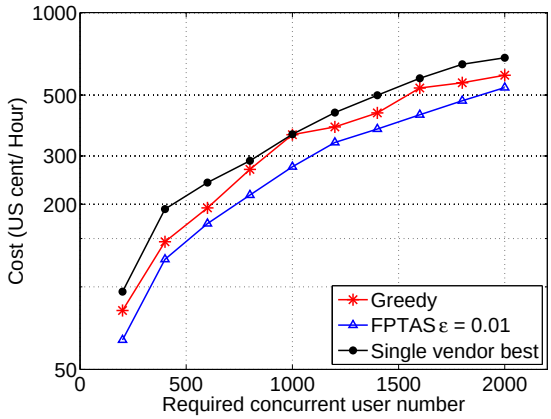
Table 5.2. The resource consumption of the chat sub-application in the service instances in different vendors

Vendor	Base resource consumption A_{ij}	Resource consumption per user Θ_{ij}
EC2	{0.2%, 0.194GB, 6.4GB }	{0.39%, 0.0220GB, 0.0GB}
GoGrid	{0.0%, 0.177GB, 15GB }	{0.18%, 0.0227GB, 0.0GB}
Rackspace	{0.1%, 0.179GB, 6.0GB }	{0.04%, 0.0232GB, 0.0GB}

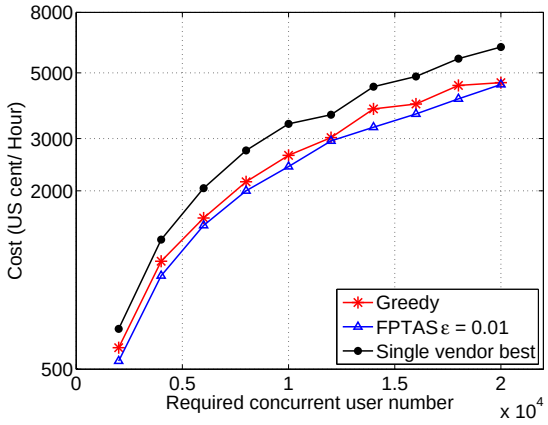
Table 5.3. The resource consumption of the pvp sub-application in the service instances in different vendors

strategy can save the cost by more than 22.5% and sometimes reduction may be up to 30%. In Fig. 5.9 (b), we also initialize the required concurrent user number as 2000, and increase the number by 2000 until the total required user number is 20000, and it also shows the same results as Fig. 5.9 (a).

To make a comparison study, Fig. 5.10 also provides the cost comparison between the multiple provider strategy and single vendor strategy without wholesales of GoGrid being considered. In general, we can see that a similar trend with Fig. 5.9. However, when we scrutinize the data, we can see the minor differences that multiple provider strategy can further reduce costs if the wholesale in GoGrid is considered. This is not surprising that the wholesales in GoGrid provide us more choices to lower the costs.



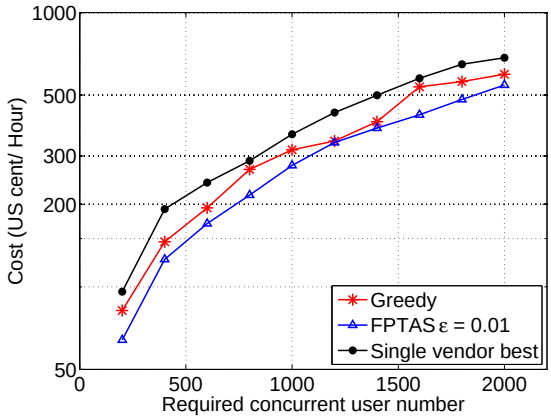
(a)



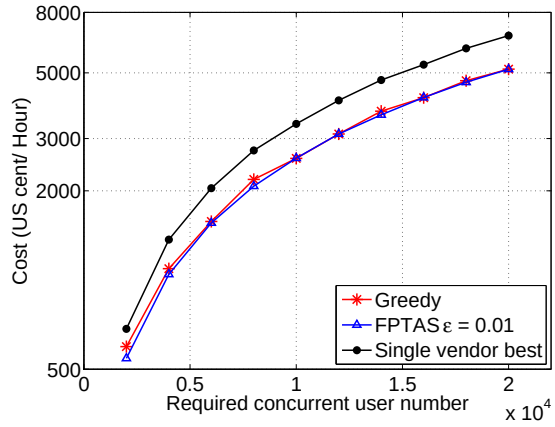
(b)

Figure 5.9. The single vendor costs and the multi-vendor costs in the web game server case (MPS$_S$) Required concurrent user number in (a) 200 user gap (b) 2000 user gap

At last, we test the performance of the multiple provider strategy in MPS$_C$ problems. We consider the situation that the multiple provider strategy save the costs most, where walk sub-application and chat sub-application are constrained. We show the cost comparisons in Fig. 5.11. As well, we compare the results of the vendors in different required concurrent user numbers, from 2000 to 20000 gapped by 2000. Our observation is as follows: compared with the result in Fig. 5.9, the reduction of the costs with walk and chat sub-applications constrained is slightly lower, and it is from about 16% to about 24%. This also conforms to the observation of the simulation result, which indicates constrained applications narrows the gap between the costs from single vendor strategy and multiple provider strategy.



(a)



(b)

Figure 5.10. The single provider and multiple provider costs in the web game server case (MPS): the required user number in (a) 200 user gap (b) 2000 user gap

5.5.1 Summary

In this section, we present the simulation and the experiment for MPS problems, and we summarize important observations, which can be important for the cloud clients, as follows.

- The larger number of applications with different requirements of resource will profit more from the multiple provider strategy.
- To choose service instances, we suggest using FPTAS algorithm while the required concurrent user number is small, and using the greedy algorithm while the number is large.

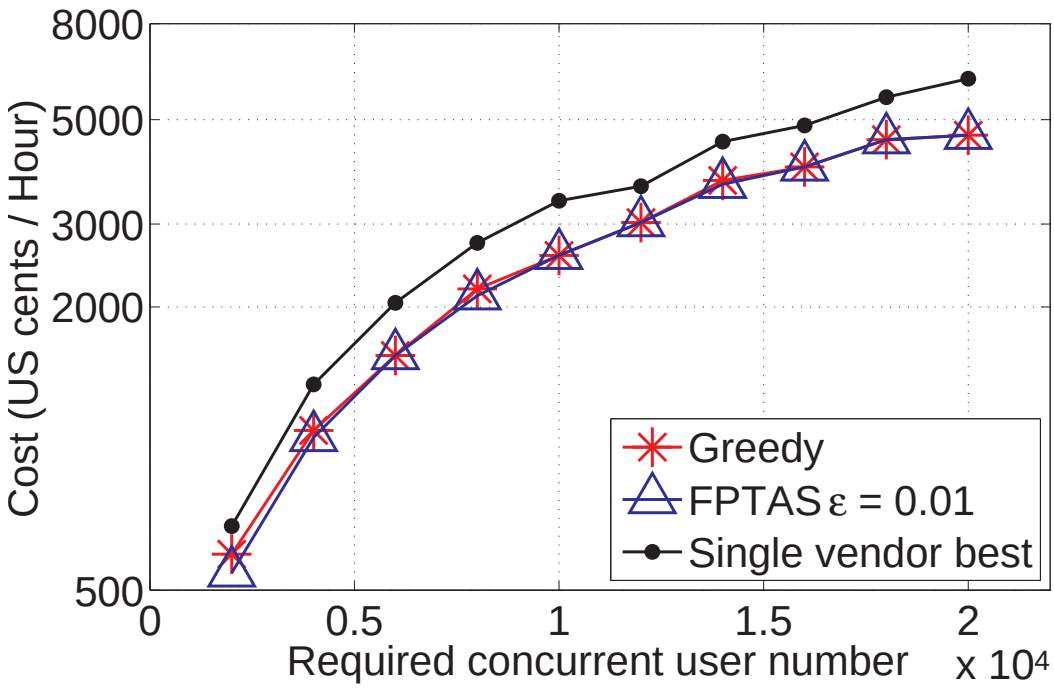


Figure 5.11. The single vendor costs and the multi-vendor costs in the web game case: Walk and chat sub-applications are constrained

- The larger number of constrained applications is, the less benefit from the multiple provider strategy.

5.6 Conclusion

In this chapter, we proposed a scheme of hosting user applications by selecting service instances from different service providers. We modeled the resource requirement of user applications and different service instances and price models of service providers. We formulated a general Multi-Provider Selection problem and two variants. We proved that these problems are NP-hard. We develop an optimal algorithm for a special case and a non-trivial (1+ϵ) algorithm.

In our simulation, we find that multi-provider strategy can reduce the cost by as much as 80%. We conduct a real MMORPG web game case using service instances of Amazon EC2, Google GoGrid and Rackspace. We observe a cost reduction by 30%.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Cloud computing has become a popular paradigm all over the world for the enterprise users to host their services over the Internet, as it frees the users from the trivial matters such as hardware and software configuration, as well as the infrastructure management. However, unlike the traditional applications, which only provide services for a single user, the services hosted in cloud need to meet the requirement from multiple users over the Internet. In this thesis, we studied the whole process of migrating the Internet applications to the cloud servers from a user traffic perspective. Specifically, we 1) measured the user traffic of one example of Internet applications, the video sharing site; 2) provided a stable workload management for the cloud servers under the fluctuating user traffic; and 3) proposed a cost-efficient deployment scheme. The details of our thesis are as follows.

- For the user traffic measurement, we studied in detail an important aspect of video sharing sites: the external links. The external links provide a unique way for the video sharing sites to accelerate the distribution of the videos. We observed that the external links can play a non-trivial role both in terms of the number of external links on a video, and the number of views contributed to the video. We also observed that the external links have quite different impacts on YouTube and Youku. We studied the external links for different video categories. We also discussed the correlations of the external links and the internal related video links. We showed that the number of internal related video links have less impact on the external links than the total views

of the video. We also study the characteristics of external links in different video age groups. We see that videos are possible to get external links and external views in all age groups. We believe that our work can provide the foundation for the video sharing sites to make more targeted advertisement, customized user development, etc.

- For the workload management, we noticed the stability problem in the sequential job scenario, and aimed at minimizing the total SLA violation loss under the constrained system resources. We first found that the system instability comes from the close loop impact between the predicted request arrival rate and the request response time. We evaluated such instability in experiment and model the root cause of the instability. We proposed a stability-aware proxy based on IEKF to minimize the SLA violation loss as well as to stabilize the system under the SLA constrain. Our design only needs to plug an SLA module to other systems, and minimally affects other parts of the systems. In experiment, we found that our scheme can maintain the system stable and achieve the minimization of the SLA violation loss.

- For the cost-efficient deployment, we proposed a scheme of hosting user applications by selecting service instances from different service providers. We modeled the resource requirement of user applications and different service instances and price models of service providers. We formulated a general Multi-Provider Selection problem and two variants. We proved that these problems are NP-hard. We develop an optimal algorithm for a special case and a non-trivial $(1+\epsilon)$ algorithm. In our simulation, we find that multi-provider strategy can reduce the cost by as much as 80%. We conduct a real MMORPG web game case using service instances of Amazon EC2, Google GoGrid and Rackspace. We observe a cost reduction by 30%.

The measurement of the Internet application user traffic, the workload management scheme to provide stable performance under the fluctuating user traffic, and a cost-efficient deployment method can be integrated into a whole process of cloud migration. As a supplement of the previous studies, our thesis aims to understand the user traffic from the perspective of the external links, our proposed workload management scheme improves the

stability of the performance of the state-of-the-art works, and our deployment scheme also substantially lowers the service construction costs.

6.2 Future Work

In the future, we extend our studies by analyzing more detail features on the online social networks and we also try to applied these features on the performance improvement of the cloud computing. In detail, they are as follows:

First, we try to collect the user traffic from different online social networks as well as the traditional websites (e.g. the news websites or forums), and analyze the different traffic patterns of these websites;

Secondly, since we have understood the features of the online social network, we may try to improve the performance of cloud systems with the features of online social networks. Given that in online social networks, the friends share more commons in interests and possibly also in locations, we may try to share the cloud computing resource among the friends.

REFERENCES

[1] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly, "Symbiotic Routing in Future Data Centers", In *Proc. ACM SIGCOMM'10*, New Delhi, India, Aug. 30 - Sept. 3, 2010.

[2] V. Adhikari, S. Jain, Y. Chen, and Z. Zhang, "Vivisecting YouTube: An Active Measurement Study", In *Proc. IEEE Infocom'12*, Orlando, Florida, USA, Mar. 25 - 30, 2012.

[3] M. Ahmad, and M. Chang, "Comparison of the Simple EKF, IEKF, and UKF for a Sensorless Full-Digital PMSM Drive", *Technical Report*, Electrical Engineering Department, University of Southern California, Dec. 1st, 2005.

[4] S. Alam, R. Barrett, and J. Kuehn, "Characterization of Scientific Workloads on Systems with Multi-Core Processors", In *Proc. IEEE IISWC'06*, San Jose, the USA, Oct. 26 - 28, 2006.

[5] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture", In *Proc. ACM SIGCOMM'08*, Seattle, the USA, Aug. 17 - 22, 2008.

[6] D. Ardagna, B. Panicucci, and M. Passacantando, "A Game Theoretic Formulation of the Service Provisioning Problem in Cloud Systems", In *Proc. ACM WWW'11*, Hyderabad, India, Mar. 28 - Apr. 1, 2011.

[7] M. Armbrust, A. Fox, and R. Griffith, etc, "Above the Clouds: A Berkeley View of Cloud Computing", *Technical Report*, Electrical Engineering and Computer Sciences, University of California at Berkeley, No. UCB/EECS-2009-28, Feb. 10, 2009.

[8] P. Barford, and M. Crovella, “Generating Representative Web Workloads for Network and Server Performance Evaluation”, In. . *Proc. ACM SIGMETRICS’98*, Madison, WI, USA, Jun 22 - 26, 1998.

[9] L. Backstrom, and J. Kleinberg, “Network Bucket Testing”, In *Proc. ACM WWW’11*, Hyderabad, India, Mar. 28 - Apr. 1, 2011.

[10] D. Bianculli, W. Binder, and M. Drago, “Automated Performance Assessment for Service-Oriented Middleware”, In. *Proc. ACM WWW’10*, Raleigh, North Carolina, USA, Apr. 26 - 30, 2010.

[11] P. Boldi, M. Rosa, and S. Vigna, “HyperANF: Approximating the Neighbourhood Function of Very Large Graphs on a Budget”, In *Proc. ACM WWW’11*, Hyderabad, India, Mar. 28 - Apr. 1, 2011.

[12] Y. Borghol, S. Mitra, S. Ardon, N. Carlsson, D. Eager, and A. Mahanti, “Characterizing and Modelling Popularity of User-generated Videos”, In *Performance Evaluation*, pp. 1037 - 1055, 68, October 2011.

[13] A. Brocco, A. Malatras, and Y. Huang, “ARiA: A Protocol for Dynamic Fully Distributed Grid Meta-Scheduling”, In *Proc. IEEE ICDCS’10*, Genoa, Italy, Jun. 21 - 25, 2010.

[14] A. Brodersen, S. Scellato, and M. Wattenhofer, “YouTube Around the World: Geographic Popularity of Videos”, In *Proc. ACM WWW’12*, Lyon, France, Apr. 16 - 20, 2012.

[15] C. Budak, D. Agrawal, and A. Abbadi, “Limiting the Spread of Misinformation in Social Networks”, In *Proc. ACM WWW’11*, Hyderabad, India, Mar. 28 - Apr. 1, 2011.

[16] G. Buttazzo, G. Lipari, and M. Caccamo, “Elastic Scheduling for Flexible Workload Management”, In. *IEEE Transactions on Computers*, Vol. 51, No. 3, Mar. 2002.

[17] H. Cai, K. Zhang, M. Wang, J. Li, L. Sun, "Customer Centric Cloud Service Model and a Case Study on Commerce as a Service", In *Proc. IEEE International Conference on Cloud Computing 2009*, Bangalore, India, Sept. 21 - 25, 2009.

[18] V. Cardellini, M. Colajanni, and P. Yu, "Dynamic Load Balancing On Web-Server Systems", In *IEEE Internet Comptuing*, May and June, 1999.

[19] C. Castillo, M. Mendoza, and B. Poblete, "Information Credibility on Twitter", In *Proc. ACM WWW'11*, Hyderabad, India, Mar. 28 - Apr. 1, 2011.

[20] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S, Moon, "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System", In *Proc. ACM IMC'07*, San Diego, California, USA, Oct. 24 - 26, 2007.

[21] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements", In. *IEEE IWQoS'03*, Monterey, Canada, Jun. 2 - 4, 2003.

[22] W. Chen, J. Chu, and J. Luan, "Collaborative Filtering for Orkut Communities: Discovery of User Latent Behavior," In *Proc. ACM WWW'09,* Madrid, Spain, Apr. 20 - 24, 2009.

[23] X. Cheng, C. Dale, and J. Liu, "Statistics and Social Network of YouTube Videos", In *Proc. IEEE IWQoS'08,* Enschede, The Netherlands, May 28 - 30, 2008.

[24] X. Cheng, and J. Liu, "NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing", In *Proc. IEEE Infocom'09*, Rio de Janeiro, Brazil, Apr. 19 - 25, 2009.

[25] X. Chu, K. Nadiminti, C. Jin, S. Venugopal, and R. Buyya, "Aneka: Next-Generation Enterprise Grid Platform for e-Science and e-Business Applications ", In *Proc. IEEE International Conference on e-Science and Grid Computing*, Dec. 10 - 13, 2007.

[26] Cisco Systems Inc. "Cisco Data Center Infrastructure 2.5 Design Guide", *Cisco Validated Design I*, December 6, 2007

[27] A. Clauset, C. Shalizi, and M. Newman, "Power-Law Distributions in Empirical Data", In *SIAM Review*, Vol. 51, No. 4, 2009.

[28] M. Crovella, and A. Bestavrosm "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", In. *IEEE Transactions on Networking*, Vol. 5, No. 6, Dec. 1997.

[29] A.Curtis, S. Keshav, and A. Lopez-Ortiz, "LEGUP: Using Heterogeneity to Reduce the Cost of Data Center Network Upgrades", In. *Proc. ACM Co-NEXT'10*, Philadelphia, USA, Nov. 30 - Dec. 3, 2010.

[30] R. Das, G. Tesauro, and W. Walsh, "Model-Based and Model-Free Approaches to Autonomic Resource Allocation", *Technical Report*, Thomas J. Watson Research Center, IBM Research Division, Nov. 16, 2005.

[31] J. Dejun, G. Pierre, and C. Chi, "Autonomous Resource Provisioning for Multi-Service Web Applications", In. *Proc. ACM WWW'10*, Raleigh, North Carolina, USA, Apr. 26 - 30, 2010.

[32] G. Demange, and G. Laroque, "Finance and the Economics of Uncertainty", Chap 3, pp. 71-72. Blackwell Publishing, 2006.

[33] Y. Ding, Y. Du, Y. Hu, Z. Liu, and L. Wang, "Broadcast Yourself: Understanding YouTube Uploaders", In *Proc. ACM IMC'11*, Berlin, Germany, Nov. 2 - 4, 2011.

[34] F. Ergun, R. Sinha, and L. Zhang, "An Improved FPTAS for Routing using Restricted Shortest Paths", In *Information Processing Letters*, 2002.

[35] F. Figueiredo, F. Benevenuto, and J. Almeida, "The Tube over Time: Characterizing Popularity Growth of YouTube Videos", In *Proc. ACM WSDM'11*, Hong Kong, China, Feb. 9 - 12, 2011.

[36] A. Finamore, M. Mellia, M. Munafo, R. Torres, and S. Rao, "YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience", In *Proc. ACM IMC'11*, Berlin, Germany, Nov. 2 - 4, 2011.

[37] T. Fond, and J, Neville, "Randomization Tests for Distinguishing Social Influence and Homophily Effects", In *Proc. ACM WWW'10,* Raleigh, USA, Apr. 26 - 30, 2010.

[38] A. Fuxman, P. Tsaparas, K. Achan, and R. Agrawal, "Using the Wisdom of the Crowds for Keyword Generation", In *Proc. ACM WWW'08,* Beijing, China, Apr. 21 - 25, 2008.

[39] A. Gambi, G. Toffetti, and M. Pezze, "Protecting SLAs with surrogate models", In. *Proc. ACM PESOS'10, International Workshop on Principles of Engineering Service-Oriented Systems*, Cape Town, South Africa, May 1 - 2, 2010.

[40] A.Ganapathi, Y. Chen, and A. Fox, "Statistics-Driven Workload Modeling for the Cloud", In. *Proc. IEEE ICDEW'10*, Long Beach, California, USA, Mar. 1 - 6, 2010.

[41] M. Garey, and D. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. H. Freeman and Company, San Francisco, 1979.

[42] P. Gill, M. Arlittz, Z. Li, and A. Mahanti, "YouTube Traffic Characterization: A View From the Edge", In *Proc. ACM IMC'07*, San Diego, California, USA, October 24 - 26, 2007.

[43] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Characterizing YouTube user sessions", In *Proc. SPIE Multimedia Computing and Networking Conference MMCN'08*, San Jose, CA, Jan. 20 - 31, 2008.

[44] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: a Scalable and Flexible Data Center Network", In *Proc. ACM SIGCOMM'09*, Barcelona, Spain, Aug. 17 - 21, 2009.

[45] S. Goel, R. Muhamad, and D. Watts, "Social Search in "Small-World" Experiments", In *Proc. ACM WWW'09*, Madrid, Spain, Apr. 20 - 24, 2009.

[46] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. "Dcell: A Scalable and Fault-tolerant Network Structure for Data Centers", In *Proc. ACM SIGCOMM'08*, Seattle, the USA, Aug. 17 - 22, 2008.

[47] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers", In *Proc. ACM SIGCOMM'09*, Barcelona, Spain, Aug. 17 - 21, 2009.

[48] C. Guo, G. Lu, and S. Yang, "SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantee", In *Proc. ACM Co-NEXT'10*, Philadelphia, USA, Nov. 30 - Dec. 3, 2010.

[49] Shadi Ibrahim, B. He, and H. Jin, "Towards Pay-As-You-Consume Cloud Computing", In *Proc. IEEE ICSC'11*, Hong Kong, Mar. 16 - 18, 2011.

[50] M. Hajjat, Xin Sun, Y. Sung, "Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud", In. *Proc. ACM SIGCOMM'10*, New Delhi, India, Aug. 30 - Sept. 3, 2010.

[51] D. Henriksson, Y. Lu, and T. Abdelzaher, "Improved Prediction for Web Server Delay Control", In. *Proc. IEEE ECRTS'04*, Catania, Sicily, Italy, Jun. 30 - Jul. 2, 2004.

[52] J. Jiang, C. Wilson, X. Wang, P. Huang, and W. Sha, "Understanding Latent Interactions in Online Social Networks", In *Proc. ACM IMC'10*, Melbourne, Australia, Nov. 1 - 3, 2010.

[53] G. Jung, M. Hiltunen, and K. Joshi, "Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures", In. *Proc. IEEE ICDCS'10*, Genoa, Italy, Jun. 21 - 25, 2010.

[54] H. Kellerer, U. Pferschy, and D. Pisinger, "Knapsack Problems", Springer, 2004.

[55] K. Kumar, J. Feng, Y. Nimmagadda, and Y. Lu, "Resource Allocation for Real-Time Tasks using Cloud Computing", In *Proc. IEEE ICCCN'11*, Maui, Hawaii, the USA, Jul. 31 - Aug. 4, 2011.

[56] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The Slashdot Zoo: Mining a Social Network with Negative Edges", In *Proc. ACM WWW'09*, Madrid, Spain, Apr. 20 - 24, 2009.

[57] K. Lai and D. Wang, "A Measurement Study of External Links of YouTube", In *Proc. IEEE Globecom'09*, Hawaii, USA, Nov. 30 - Dec. 4, 2009.

[58] K. Lai and D. Wang, "Towards Understanding the External Links on Video Sharing Sites: Measurement and Analysis", in *Proc. ACM NOSSDAV'10*, Amsterdam, Netherland, Jun. 2 - 4, 2010.

[59] C. Lampe, N. Ellison and C. Steinfield, "A Familiar Face(book): Profile Elements as Signals in an Online Social Network", In *Proc. ACM CHI'07*, San Jose, California, USA, Apr. 28 - May 3, 2007.

[60] K. Lerman, and T. Hogg. "Using a Model of Social Dynamics to Predict Popularity of News", In *Proc. ACM WWW'10*, Raleigh, USA, April 26 - 30, 2010.

[61] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, "Statistical Properties of Community Structure in Large Social and Information Networks", In *Proc. ACM WWW'08*, Beijing, China, Apr. 21 - 25, 2008.

[62] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting Positive and Negative Links in Online Social Networks", In *Proc. ACM WWW'10*, Raleigh, USA, Apr. 26 - 30, 2010.

[63] Z. Liu, M. Squillante, and J. Wolf, "On Maximizing Service-Level-Agreement Profits", In. *Proc. ACM EC'01*, Tampa, Florida, USA, Oct. 14 - 17, 2001.

[64] L. Li, H. Ji, and J. Luo. "The Iterated Extended Kalman Particle Filter", In *Proc. IEEE ISCIT*, 2005.

[65] Q. Li, J. Feng, Z. Pend, Q. Lu, and X. Liang. "An iterated extend Kalman particle filter for multi-sensor based on pseudo sequential fusion", In *Proc. IEEE International Conference on Robotics and Biomimetics* , 2007.

[66] Y. Lu, T.F. Abdelzaher, C. Lu, L. Sha, and X. Liu, "Feedback Control with Queueing-Theoretic Prediction for Relative Delay Guarantees in Web Servers", In *Proc. IEEE RTAS'03*, Washington DC, the USA, May 27 - 30, 2003.

[67] M. Mao, and M. Humphrey, "Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows", In *Proc. ACM Super Computing*, Nov. 14 - 17, 2011.

[68] Y. Matsuo and H. Yamamoto, "Community Gravity: Measuring Bidirectional Effects by Trust and Rating on Online", In *Proc. ACM WWW'09*, Madrid, Spain, Apr. 20 - 24, 2009.

[69] A. Mishra, J. Hellerstein, and W. Cirne, "Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters", In. *Proc. ACM SIGMETRICS'10*, New York, USA, Jun. 14 - 18, 2010.

[70] A. Mislove, M. Marcon, K. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks", In *Proc. ACM IMC'07*, San Diego, California, USA, Oct. 24 - 26, 2007.

[71] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti, "Characterizing Web-based Video Sharing Workload", In ACM Transactions on the Web, Vol.5, No. 2, May 2011.

[72] A. Mohaisen, A. Yun, and Y. Kim, "Measuring the Mixing Time of Social Graphs", In *Proc. ACM IMC'10*, Melbourne, Australia, November 1 - 3, 2010.

[73] L. Popa, S. Ratnasamy, G. Iannaccone, "A Cost Comparison of Data Center Network Architecture", In. *Proc. ACM Co-NEXT'10*, Philadelphia, USA, Nov. 30 - Dec. 3, 2010.

[74] B. Ribeiro, and D. Towsley, "Estimating and Sampling Graphs with Multidimensional Random Walks", In *Proc. ACM IMC'10*, Melbourne, Australia, Nov. 1 - 3, 2010.

[75] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft, "Track Globally, Deliver Locally: Improving Content Delivery Networks by Tracking Geographic Social Cascades", In *Proc. ACM WWW'11*, Hyderabad, India, Mar. 28 - Apr. 1, 2011.

[76] S. Sen, J. Vig, and J. Riedl, "Tagommenders: Connecting Users to Items through Tags", In *Proc. ACM WWW'09*, Madrid, Spain, Apr. 20 - 24, 2009.

[77] L. Sha, X. Liu, Y. Lu, and T. Abdelzaher, "Queueing Model Based Network Server Performance Control", In *Proc. IEEE RTSS'02*, Austin, Texas, the USA, Dec. 3 - 5, 2002.

[78] U. Sharma, P. Sheno, S. Sahu and A. Shaikh, "A Cost-aware Elasticity Provisioning System for the Cloud", In *Proc. IEEE ICDCS'11*, Minneapolis, Minnesota, the USA, Jun. 20 - 24, 2011.

[79] G. Singh, C. Kesselman, and E. Deelman, "Adaptive Pricing for Resource Reservations in Shared Environments", In *Proc. IEEE Grid'07*, Austin, Texas, the USA, Sept. 19 - 21, 2007.

[80] C. Stewart, and K. Shen, "Performance Modeling and System Management for Multi-component Online Services", In *Proc. USENIX NDSI'05*, Houston, the USA, Apr. 4 - 6, 2005.

[81] G. Szabo and B. Huberman, "Predicting opularity of online content", In *Communications of the ACM*, Vol. 53, Issue 8, August, 2010.

[82] J. Tang, M. Musolesi, and C. Mascolo, "Temporal Distance Metrics for Social Network Analysis", In *Proc. ACM SIGCOMM WOSN'09*, Bacelona, Spain, Aug. 17, 2009.

[83] TBS LLC, "Public Cloud Computing: True Cost Reduction, Not Just Cost Transfer", *Technical Report*, TBS LLC., Copyright 2010.

[84] M. Torkjazi, R. Rejaie, and W. Willinger, "Hot Today, Gone Tomorrow: On the Migration of MySpace Users", In *Proc. ACM SIGCOMM WOSN'09*, Bacelona, Spain, Aug. 17, 2009.

[85] K. Tsakalozos, M. Roussopoulos, and Vangelis Floros, "Nefeli: Hint-based Execution of Workloads in Clouds", In. *Proc. IEEE ICDCS'10*, Genoa, Italy, Jun. 21 - 25, 2010.

[86] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, and L. Zhou, "Distributed Systems Meet Economics:Pricing in the Cloud", In *Proc. USENIX HotCloud'11*, Portland, the USA, Jun. 14 - 7, 2011.

[87] G. Welch, and G. Bishop, "An Introduction to the Kalman Filter", In *Proc. ACM SIGGRAPH*, 2001, course 8, Copyright 2001 by ACM, Inc.

[88] P. Wendell, J. Jiang, and M. Freedman, "DONAR: Decentralized Server Selection for Cloud Services", In. *Proc. ACM SIGCOMM'10*, New Delhi, India, Aug. 30 - Sept. 3, 2010.

[89] T. Zheng, J. Yang, and M. Woodside, "Tracking Time-Varying Parameters in Software Systems with Extended Kalman Filters", In. *Proc. ACM CASCON'05*, Toronto, Ontario, Canada, Oct. 18 - 20, 2005.

[90] M. Valafar, R. Rejaie, and W. Willinger, "Beyond Friendship Graphs: A Study of User Interactions in Flickr", In *Proc. ACM SIGCOMM WOSN'09*, Bacelona, Spain, Aug. 17th, 2009.

[91] B. Viswanath, A. Mislove, and M. Cha, "On the Evolution of User Interaction in Facebook Bimal", In *Proc. ACM SIGCOMM WOSN'09*, Bacelona, Spain, Aug. 17, 2009.

[92] K. Astrom, and T. Hagglund "PID Controllers: Theory, Design, and Tuning", Instrument Society of America, ISBN 1556175167, Jan. 1st, 1995.

[93] S. Haykin, "Adaptive Filter Theory (4th Edition)", Prentice Hall, ISBN: 9780130901262, Sept. 24th, 2001.

[94] R. Young, "Internet-Based Games", North Carolina State University, by CRC Press LLC, 2001. available at http://liquidnarrative.csc.ncsu.edu/pubs/phic.pdf

[95] R. Xiang, J. Neville, and M. Rogati, "Modeling Relationship Strength in Online Social Networks", In *Proc. ACM WWW'10*, Raleigh, USA, Apr. 26 - 30, 2010.

[96] L. Xie, A. Natsev, J. Kender, M. Hill, and J. Smith, "Visual Memes in Social Media: Tracking Real-World News in YouTube Videos", In *Proc. ACM MM'11*, Scottsdale, Arizona, USA, Nov. 28 - Dec. 1, 2011.

[97] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges", In *Journal Of Internet Services And Applications*, Vol. 1, No. 1, 2010.

[98] J. Zhang, Y. Song, and T. Leung, "Improving Video Classification via YouTube Video Co-Watch Data", In *Proc. ACM SBNMA'11*, Scottsdale, Arizona, USA, Dec. 1, 2011.

[99] S. Zhang, H. Wu, W. Wang, B. Yang, P. Liu, A. Vasilakos, "Distributed Workload and Response Time Management for Web Applications", In *Proc. ACM CNSM'11,* Paris, France, Oct. 24 - 28, 2011.

[100] J. Zhou, Y. Li, V. Adhikari, and Z. Zhang, "Counting YouTube Videos via Random Prefix Sampling", In *Proc. ACM IMC'11*, Berlin, Germany, Nov. 2 - 4, 2011.

[101] R. Zhou, S. Khemmarat, and L. Gao, "The Impact of YouTube Recommendation System on Video Views", In *Proc. ACM IMC'10*, Melbourne, Australia, Nov. 1 - 3, 2010.

[102] P. Xiong, Z. Wang, S. Malkowski, Q. Wang, D. Jayasinghe, and C. Pu "Economical and Robust Provisioning of N-Tier Cloud Workloads: A Multi-level Control Approach", In *Proc. IEEE ICDCS'11*, Minneapolis, Minnesota, the USA, Jun. 20 - 24, 2011.

[103] "Alexa", http://www.Alexa.com.

[104] "The Data Sets of External Links from YouTube and Youku", http://www4.comp.polyu.edu.hk/cskflai/external-link.html.

[105] "Kaixin is not happy: suffering from the loss of 60% traffic in one year" http://news.xinhuanet.com/it/2011-08/04/c_121812027.htm

[106] "YouTube", http://www.youtube.com.

[107] "Youku", http://www.youku.com.

[108] "V8 - V8 JavaScript Engine", http://code.google.com/p/v8/.

[109] “Manjrasoft Products- Aneka”, http://www.manjrasoft.com/products.html

[110] “AWS Case Study: 6 Waves Limited Rides the AWS Wave”, http://aws.amazon.com/solutions/case-studies/6waves/.

[111] “AWS Case Study: 99designs”, http://aws.amazon.com/solutions/case-studies/99designs/.

[112] “Amazon EC2 Pricing”, http://aws.amazon.com/ec2/pricing/

[113] “RSS, Cloud Computing and cost reduction”, http://wiki.services.eoportal.org/tiki-view_blog_post.php?postId=28

[114] “GoGrid Pricing”, http://www.gogrid.com/cloud-hosting/cloud-hosting-pricing.v2.php

[115] “Nagios”, http://www.nagios.com/

[116] “Oracle - Clustering and Securing Web Applications: A Tutorial”, http://developers.sun.com/appserver/reference/techart/load-balancing.html

[117] “Rackspace Cloud Servers FAQs”, http://www.rackspace.com/cloud/cloud_hosting_products/servers/faq/

[118] “Rackspace Cloud Service”, http://www.rackspace.com/cloud/cloud_hosting_products/servers/pricing/

[119] “Splunk”, http://www.splunk.com/

[120] “The PHP MMORPG Project ”, http://sourceforge.net/projects/php-mmo-rpg/.

[121] “MMORPG- The phpRPG Project”, http://50.57.179.94/rpg.

[122] Zenoss “http://www.zenoss.com/”

[123] “Google Apps”, http://apps.google.com/.

[124] “Google Docs” http://docs.google.com/.

[125] “DropBox”, http://www.dropbox.com/.

[126] “Tinyproxy: light-weight HTTP proxy daemon”, https://www.banu.com/tinyproxy.

[127] “Zattoo - watch the online TV”, http://www.zattoo.com/.