LATENCY-AWARE PRICING IN THE CLOUD MARKET

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Yang Zhang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Seokcheon Lee, Chair

School of Industrial Engineering

Dr. Christopher J. Quinn

School of Industrial Engineering

Dr. Mario Ventresca

School of Industrial Engineering

Dr. Yu Xiang

AT&T Labs Research

Approved by:

Dr. Steven J. Landry

Head of the School Graduate Program

献给我的爸爸和妈妈

Dedicated to my mom and dad

for being my captains, pillows, role models, commentators, and cheerleading squads

whenever needed

ACKNOWLEDGMENTS

The past five years at Purdue have been an unforgettable and invaluable experience to me. I owe my sincere gratitude to all those people who help me during my Ph.D. life.

First and foremost, my greatest thanks to my advisor Dr. Seokcheon Lee. Five years ago, I could not even speak English fluently. He gave me the chance to come to Purdue. He always gives me freedom to explore on my own and provides constructive criticism when I make mistakes. I might be immature, crazy, and a "trouble maker", but he always smiles to me and cheers me up when I am upset.

It is my great honor to have Dr. Christopher Quinn, Dr. Mario Ventresca and Dr. Yu Xiang on my thesis committee. Thanks for their advice and guidance on my research, and their time and effort to help me fulfill the degree requirements.

I am lucky to have many great labmates and friends at Purdue IE: Mohammad Moshref-Javadi (now at MIT), Chul Hun Choi (now at Samsung), Ashutosh Nayak (now at UC Davis), Kyle Cooper (now at TCS), Wenqi Wang (now at Facebook), Julian Archer (now at PepsiCo), Viplove Arora, Rahul Sucharitha, Sungbum Jun, Cansu Agrali, Xueyan Niu, and many others, who make my Ph.D. experience a memorable one: 300 days out of one year, coming to school in the early morning, and going back home around 11pm, we have nothing to offer but blood, toil, tears and sweat. Our most common memory is in Grissom Hall (Thanks Purdue for the great decorated building). Thanks for their accompany and support, which make me never feel alone.

During my Ph.D., I have two wonderful internships. I thank my mentors Dr. Devadatta M. Kulkarni and Dr. Mark Zhuravel at Tata Consultancy Services Innovation Labs, Dr. Yu Xiang and Dr. Moo-Ryong Ra at AT&T Labs Research. I still keep the book named "Bold" Dr. Kulkarni gave me, and remember he encouraged me to be

confident and talk about my ideas bravely. It is the intern experience that makes me believe I can make contributions to the society in different industries from production to telecommunication.

I would like to thank my undergraduate advisor Dr. Jia Shu at Southeast University for his constant encouragement and support. Without him, I would not even think about applying for Ph.D. when I was an undergraduate student. Thanks Kaike Zhang (now at Bayer), my alumni from Southeast University, for his support and suggestions whenever needed. I am forever grateful to have that friend in the US.

Last but not least, I thank my mom, dad, brother, and boyfriend for their unconditional love and support.

Thank you all!

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CSP	Cloud service provider
ICSP	Individual cloud service provider
SLA	Service Level Agreement
MILP	Mixed-integer linear program
CDF	Cumulative distribution function
PDF	Probability density function
AWS	Amazon Web Services
Amazon EC2	Amazon Elastic Compute Cloud
Amazon S3	Amazon Simple Service
IaaS	Infrastructure as a service
PaaS	Platform as a service
SaaS	Software as a service
STaaS	Storage as a service
SECaaS	Security as a service
VM	Virtual machine
OTR-EG	One-time requests with expected guarantee
OTR-P	One-time requests with penalty
PR	Persistent request

ABSTRACT

Zhang Yang Ph.D., Purdue University, May 2019. Latency-Aware Pricing in the Cloud Market. Major Professor: Seokcheon Lee.

Latency is regarded as the Achilles heel of cloud computing. Pricing is an essential component in the cloud market since it not only directly affects a cloud service provider's (CSP's) revenue but also a user's budget. This dissertation investigates the latency-aware pricing schemes that provide rigorous performance guarantees for the cloud market. The research is conducted along the following major problems as summarized below:

First, we will address a major challenge confronting the CSPs utilizing a tiered storage (with cold storage and hot storage) architecture - how to maximize their overall profit over a variety of storage tiers that offer distinct characteristics, as well as file placement and access request scheduling policies. To this end, we propose a scheme where the CSP offers a two-stage auction process for (a) requesting storage capacity, and (b) requesting accesses with latency requirements. Our two-stage bidding scheme provides a hybrid storage and access optimization framework with the objective of maximizing the CSP's total net profit over four dimensions: file acceptance decision, placement of accepted files, file access decision and access request scheduling policy. The proposed optimization is a mixed-integer nonlinear program that is hard to solve. We propose an efficient heuristic to relax the integer optimization and to solve the resulting nonlinear stochastic programs. The algorithm is evaluated under different scenarios and with different storage system parameters, and insightful numerical results are reported by comparing the proposed approach with other profit-maximization models. We see a profit increase of over 60% of our proposed method compared to other schemes in certain simulation scenarios.

Second, we will resolve one of the challenges when using Amazon Web Services (AWS). Amazon Elastic Compute Cloud (EC2) provides two most popular pricing schemes–i) the *costly* on-demand instance where the job is guaranteed to be completed, and ii) the *cheap* spot instance where a job may be interrupted. We consider a user can select a combination of on-demand and spot instances to finish a task. Thus he needs to find the optimal bidding price for the spot-instance, and the portion of the job to be run on the on-demand instance. We formulate the problem as an optimization problem and seek to find the optimal solution. We consider three bidding strategies: one-time requests with expected guarantee, one-time requests with penalty for incomplete job and violating the deadline, and persistent requests. Even without a penalty on incomplete jobs, the optimization problem turns out to be non-convex. Nevertheless, we show that the portion of the job to be run on the on-demand instance is at most half. If the job has a higher execution time or smaller deadline, the bidding price is higher and vice versa. Additionally, the user never selects the on-demand instance if the execution time is smaller than the deadline. The numerical results illustrate the sensitivity of the effective portfolio to several of the parameters involved in the model. Our empirical analysis on the Amazon EC2 data shows that our strategies can be employed on the real instances, where the expected total cost of the proposed scheme decreases over 45% compared to the baseline strategy.

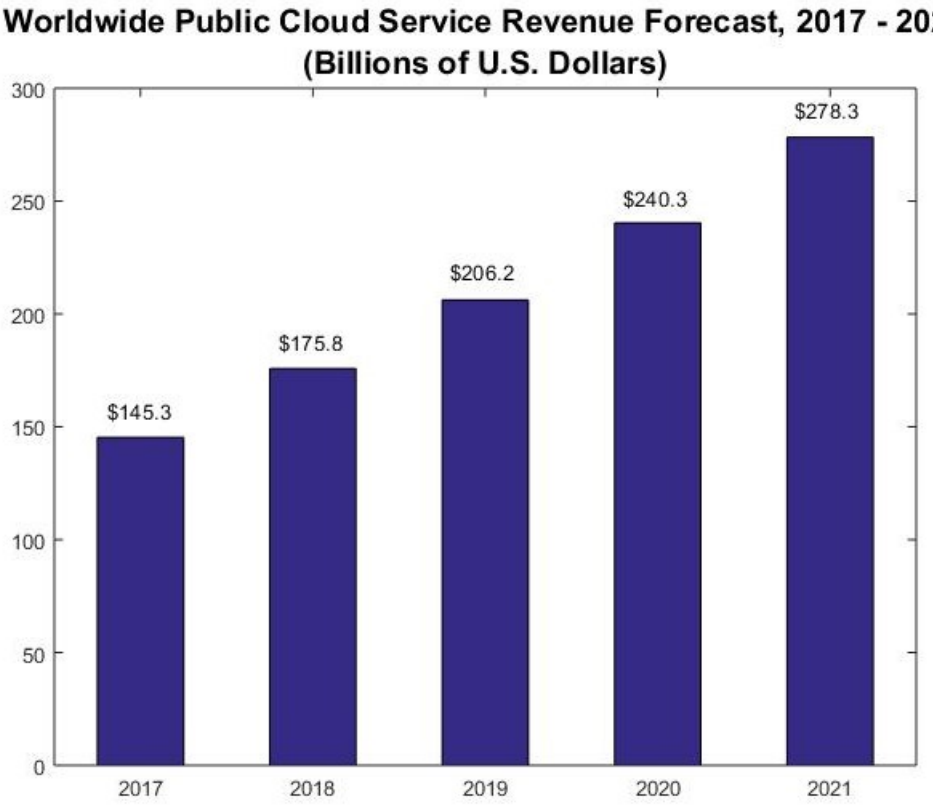
1. INTRODUCTION

1.1 Motivation

Cloud computing, incorporated new economic and financial models in the IT service market, is changing the traditional business plans. It enables the users to manage, store and process data more efficiently with reasonable prices. A research and advisory firm Gartner predicts that the worldwide public cloud services (including Cloud Business Process Services, Cloud Application Infrastructure Services, Cloud Application Services, Cloud Management and Security Services, and Cloud System Infrastructure Services) revenue will grow 17.3% to $206.2 billion from $175.8 billion in 2018, and increase to $278.3 billiion 2021 with a four-year average compound growth rate of 17.64% (see Fig. 1.1) [1].

Cloud computing, serving as the IT infrastructure to drive business intelligence, is proliferating across various organizations today. On the other hand, the cloud business drives the cloud service providers' (CSPs') profits. For example, the profit of Amazon has been driven by its cloud business, one of the leading CSPs–Amazon Web Services (AWS), in recent years. From Figure 1.2 we can see that AWS made nearly 58% of Amazon's profit in the fourth quarter of 2018 with only 10% of Amazon's net sales, up from 26% in 2015 [2]. With the drive of cloud business profits, more small scale CSPs, such as Ready Space [3] and GoGrid [4], are emerging quickly along with the large scale CSPs' stable growth such as AWS [5], Windows Azure [6] and Google Cloud Platform [7].

The cloud computing become an overwhelming popular topic in the end of 2000s, but actually has a long history - as early as 1960s, computer scientist John McCarthy was the first to suggest the computer time-sharing technology might lead to the future, where computer resources, acting like electricity and water, became a public



(a) The Rapid Growth of Cloud Computing, 2017-2021

	2017	2018	2019	2020	2021
Cloud Business Process Services (BPaaS)	42.2	46.6	50.3	54.1	58.1
Cloud Application Infrastructure Services (PaaS)	11.9	15.2	18.8	23.0	27.7
Cloud Application Services (SaaS)	58.8	72.2	85.1	98.9	113.1
Cloud Management and Security Services	8.7	10.7	12.5	14.4	16.3
Cloud System Infrastructure Services (IaaS)	23.6	31.0	39.5	49.9	63.0
Total Market	**145.3**	**175.8**	**206.2**	**240.3**	**278.3**

(b) Worldwide Cloud IT Infrastructure Market Forecast By Service Models 2017 -2021 (shares based on Value)

Fig. 1.1.: Worldwide Public Cloud Service Revenue Forecast (Billions of U.S. Dollars) [1]

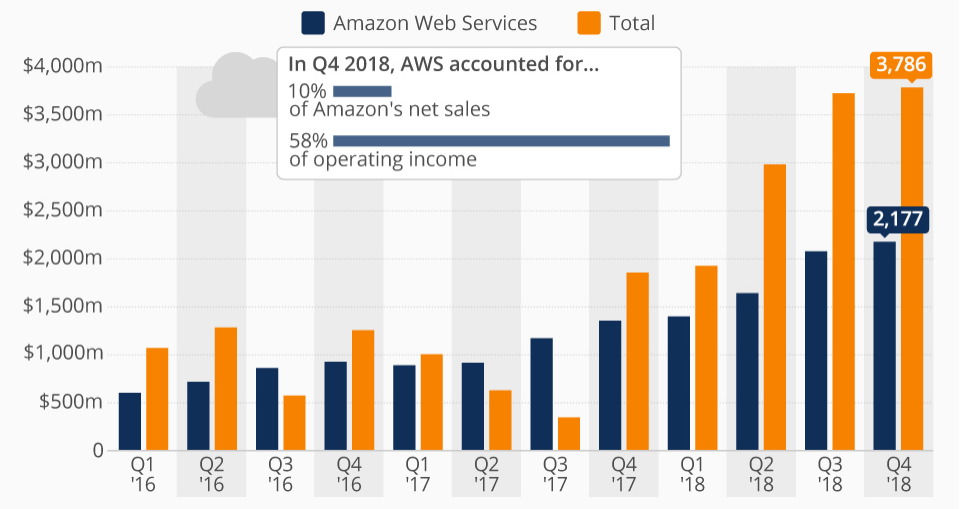


Fig. 1.2.: Amazon's quarterly operating profit (in million U.S. dollars)

utility. However, because of the limitations of technology, the cloud computing did not take off until 1990s. During 1990s, the development of technology finally satisfied the requirements of cloud computing and bring it to our life.

One of the major benefits of cloud computing is to provide organizations with on demand scalability/provisioning and pricing models. As the cloud's prominence continues growing, the IT organizations can see considerable improvement of their on-demand flexibility in computing resources, which are beyond their requirements and expectations. However, the broader performance problem of latency from cloud to the end users across the Internet are still under addressed, which has a direct impact on the user's satisfaction and is regarded as the Achilles heel of cloud computing [8].

Pricing is important for any business, which defines the value that the product/service is worth for the product sellers/service providers to provide and for the product buyers/service users to use. Same to the cloud market, pricing is one of the most important factors controlling the demand of the cloud product/service [9]. It is claimed that the success of cloud computing in the IT market can be realized only by developing adequate pricing strategies [10]. Each CSP has his framework to calculate the prices for the provided cloud services and each user has her evaluation for the

cloud services. The CSP's objective is to maximize his profit while user's goal is to minimize her cost for the required service. Thus, an optimal pricing plays a vital role in meeting the both parties' requirements.

In 2016, it was reported that the ride-hailing giant, Uber, solicited bids from Google, Microsoft, and Amazon to move part of its infrastructure to the public cloud. At that time, Uber operated at 69 countries and wanted to make sure that it could get access to fundamentally unlimited supercomputing power across form these countries. In order to guarantee the high performance and availability, uber wanted its server facilities were in the neighborhood of the customers [11]. Because in this ride-hailing business, *milliseconds count*. Therefore, we can see that pricing and latency are two of the most important factors affecting the customers' experience, utilities and the CSP's profits.

In this thesis, latency and pricing are the two major aspects that we consider. First we will cover the background in cloud computing: key concepts in cloud computing, cloud computing service models, deployment models, latency in the cloud, and the current practices of the pricing models in the cloud market. Then we explore two research directions and summarized the results and contributions in Section 1.2.

1.2 Overview of Results and Contributions

The contributions of this thesis are summarized as follows:

1. **Latency-aware pricing in the cloud storage**: Proposing a framework to address a major challenge confronting the CSPs utilizing tiered storage architecture - how to maximize their overall profit over a variety of storage tiers that offer distinct characteristics, as well as file placement and access request scheduling policies. Specifically:

    - *Comprehensive future consideration*: This chapter aims to propose a systematic framework that integrates both file storage and file access, which optimizes the system over four dimensions: file acceptance decision, place-

ment of accepted files, file access decision and access request scheduling. The proposed framework encompasses future access information such as bidding price for access, latency requirements and expected access request arrival rates.

- *Two-Stage, Latency-Aware Bidding*: The existing pricing schemes do not provide any latency guarantees. We consider a two-stage market–in the first stage, the users bid to store their files in order to achieve an expected latency. The CSP stores the files either in the hot or cold storage depending on the expected latencies required by the users; in the second stage, the users again bid to access the files based on their realized latency requirements. The CSP maximizes the profit by accepting those bids whose latency requirements can be fulfilled.
- *Computational Efficiency*: We quantify the service latency with respect to both hot and cold storage. The proposed optimization is modeled as a mixed-integer nonlinear program (MINLP), which is hard to solve. We propose an efficient heuristic to relax the integer optimization and solve the non-convex problem.
- *Insightful Numerical Results*: The performance of the proposed approach is evaluated in various cases. It is observed that the profits obtained from the proposed method are higher than those of other methods, and the access request acceptance rate (ARAR) also dominates that of other methods as the capacity of the cold storage or the service rate of hot storage increases. For example, we see a profit increase of over 60% of our proposed method compared to other baselines as the capacity of cold storage increases beyond 500TB with our simulation scenario.

2. **Latency-aware pricing in the cloud computing**: Developing a scheme to address a major challenge for users using Amazon EC2 - how to minimize the

users' total cost over two types of computing instances - on-demand instance and spot instance using different bidding strategies.

- **User's optimal or local optimal bidding strategies**: For the one-time request and persistent request job, we formulate the cost minimization problem as an optimization problem. The problem turns out to be non-convex. Nevertheless, we find analytical expression for the optimal solutions for the one-time request without penalty and the persistent request. However, for the one-time request with penalty, we provide algorithms for solving the proposed non-convex problem. Specifically,
- **Analytical Results**: Our analytical result shows that only when the deadline is smaller than the execution time, the user should select the on-demand instances. We show a threshold type behavior for one-time request. When the penalty is above a certain threshold, the user opts for the on-demand instances. However, below the threshold, the portion of the job that is run on the on-demand instance becomes independent of the penalty parameters. Our result shows that the persistent requests reduce the expected cost of the user compared to the one-time-request.
- **Numerical Evaluation**: We, empirically, evaluate the impact of different parameters on the portion of the job should be run on the spot instances, and the bidding price. Our result shows that the expected cost, and the portion of the job that is run on the on-demand instance decreases with the increase in the deadline. The bidding price in the persistent request instance decreases with the increase in the deadline. However, the bidding price in the one-time request increases with the increase in the deadline in the one-time request.
- **Real time Data**: Using the real time data, we show the strength of our approach compared to the baseline strategies readily employed by the users. Specifically, we compute the optimal bidding strategy in the spot-

instance, and the optimal portion of the job should be run on the on-demand instance. Finally, we show that the user's cost is reduced using our approach compared to the baseline ones.

Note that in cloud storage, we consider from the CSP's perspective to maximize his total profits from data storage and access, while in cloud computing where the spot instance is used, we consider from a user's perspective to minimize his total cost for running the job. However, we need to guarantee that the user's latency requirements are satisfied. In cloud storage, the CSP's file acceptance decision, placement of accepted files, file access decision and access request scheduling policy will highly impact the users' latency, thus we consider from a CSP's perspective; while in cloud computing, the user's decision on how much to run on the spot instance and how much to bid for the spot instance is highly related to the user's latency, thus it is the user's responsibility to take care of the latency.

1.3 Thesis Outline

The rest of the dissertation is organized as follows.

- Chapter 2: we give an overview of various definitions of cloud computing, service models, deployment models, latency and pricing models in the cloud market.
- Chapter 3: we present a model of latency-aware pricing in cloud storage. We consider the CSPs has a tiered storage architecture and address a major challenge confronting them: how to maximize the CSP's total net profit over four dimensions: file acceptance decision, placement of accepted files, file access decision and access request scheduling policy. We propose a scheme where the CSP offers a two-stage auction process for (a) requesting storage capacity, and (b) requesting accesses with latency requirements. We develop a two-stage stochastic optimization model, which is a mixed-integer nonlinear program, and propose an efficient heuristic to solve the model. The proposed model and heuristic are

evaluated under different scenarios with different storage system parameters. This chapter is based on our work [12].

- Chapter 4: we introduce a model of latency-aware pricing in cloud computing. We consider a user can select a combination of on-demand and spot instances to finish a task before a deadline: with the *costly* on-demand instance, the job is guaranteed to be completed, however, with the *cheap* spot instance, a job may be interrupted. The objective is to help the user to decide optimal bidding price for the spot-instance, and the portion of the job to be run on the on-demand instance. We consider three bidding strategies: one-time requests with expected guarantee, one-time requests with penalty, and persistent request. We formulate the problem as an optimization problem and seek to find the optimal solutions. The numerical results illustrate the sensitivity of the effective portfolio to several of the parameters involved in the model. This chapter is based on our work [13].

- Chapter 5: we will finally conclude in Chapter 5, give a review of the fog computing, which resides in between cloud and IoT devices, and introduce future works related to fog computing.

2. BACKGROUND

In this chapter, we aim to provide an overview of different concepts and models in cloud computing. We begin with the various definitions of cloud computing, followed by service models and deployment models, then we introduce the latency and current practices of pricing models in the cloud market.

2.1 The Technical View of Cloud

In this section, we will present a technical view of the cloud: the definitions of the cloud computing, deployment models, service models and characteristics [14]. The NIST definitions lists four deployment models: public, private, hybrid and community cloud, three service models: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS), and five essential characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity or expansion, and measured service [14, 15]. Fig. 2.3 represents the logic of the above concepts. We will give more thorough introductions of the background and concepts in the following subsections.

2.1.1 What is Cloud Computing

Cloud computing allows scalable on-demand sharing of resources and costs among a large number of end users, which enables end users to process, manage, and store data efficiently at high speeds but reasonable prices [9]. Many definitions have been proposed for cloud computing [15–18]. In [17], cloud computing is defined as "a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power,

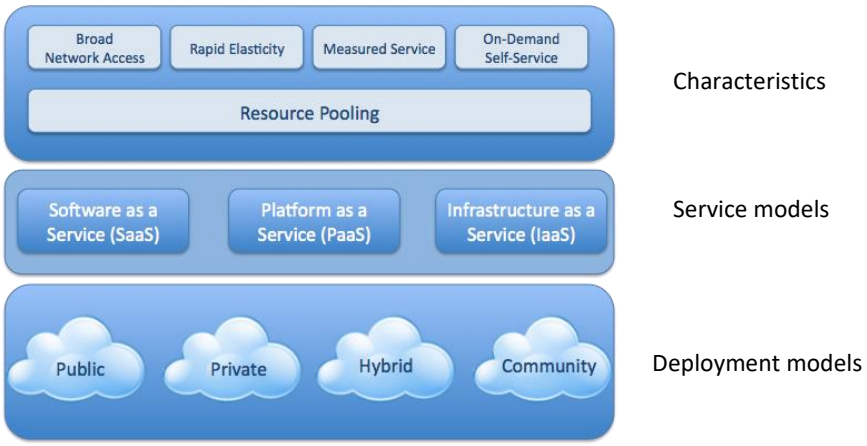


Fig. 2.1.: The technical view of cloud [14]

storage, platforms, and services are delivered on demand to external customers over the Internet." In [18], it is defined as "Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay per-use model in which guarantees are offered by the Infrastructure Provider by means of customized Service-Level Agreements (SLAs)." National Institute of Technology defines it as " Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [15].

CSPs provide the users opportunity to create virtual machines (VMs) and exploit theses VMs on-demand. Next we will introduce what the VM is. A VM was originally defined as "an efficient, isolated duplicate of a real computer machine" [19]. In cloud computing, a VM is an emulation of a computer system, which can provide functionality of a physical computer by sharing resources (CPU, memory, NIC, disk, etc.) with other VMs. In other words, it is creating a computer within a computer.

One example is presented in Fig. 2.2, where each VM is isolated from other VMs while they are sharing the resources like CPU, memory, etc. Additionally, the amount of these resources can be specified by the user [5, 6]. The physical hardware is generally called as "host", and the VMs running on the physical hardware is generally called as "guest". From the above description, we can see that a host can host many guests, and each of the guest can emulate various operating systems and hardware platforms and support different applications. In Amazon EC2, these VMs are called as instances [20].

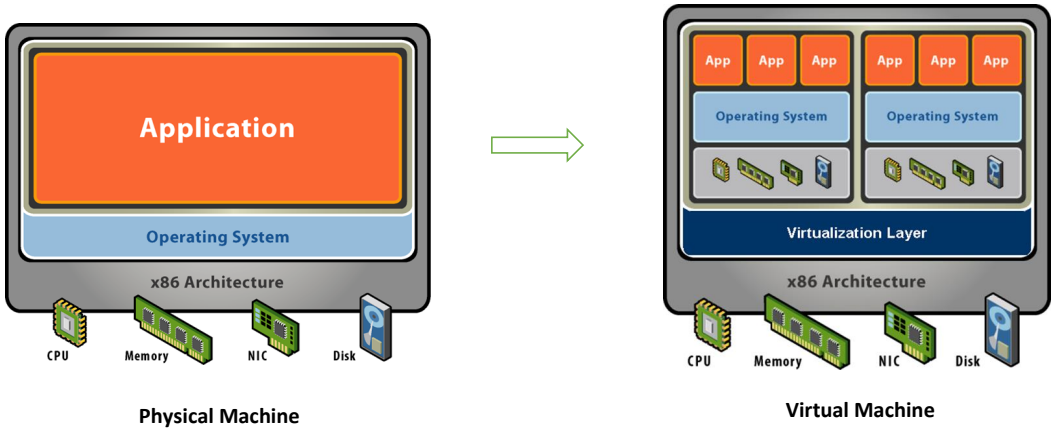


Fig. 2.2.: From physical machine to virtual machine

All definitions above demonstrate that cloud computing consists of the next generation of delivery and deployment of a new paradigm of IT (including hardware and applications). Industries surges to adopt cloud computing to solve their computing and storage problems. The main reasons are: (1) reduction in hardware cost, rise in storage capacity and computing power (e.g., from CPU to GUP), and development of multi-core architectures and supercomputers; (2) the increased popularity of Web 2.0 applications and service computing; (3) growing data size in scientific simulation and research archive and publishment, and wide-spread adoption of big data analytics [21].

2.1.2 Cloud Computing Deployment Models

In the literature, there are four major deployment models in cloud computing: public cloud, private cloud, hybrid cloud and community cloud, where the categorization depends on the dedicated audience requirements, service limits, etc. [15, 21–23]. In this subsection, we will review the definitions, benefits and pitfalls of different cloud computing deployment models.

In public cloud, the resources and services are offered to the public by CSPs. The users can scale their usage on demand without purchasing the hardware. Examples of public cloud providers are Google, Amazon, Microsoft, etc. Public cloud is easy to setup, flexible scale-up or scale-down, pay as per use. However, there are some drawbacks that the users should be consider, e.g., data security and privacy [24].

Unlike public cloud, the private cloud resources and services are operated solely for a single organization or its partners [15,25]. It gives the organization more control of the infrastructure and computing resources, and greater authority over security and privacy [26]. However, when a company decides to exploit private cloud, he also needs to allocate budget to purchase equipment, software and staffing, etc. [24], thus it will incur higher costs than public cloud. If the company has limited budget, it means it can afford finite cloud resources, which will lead to finite scalability.

In terms of community cloud, its target customers falls between public and private cloud: the resources and services are provided to two or more organizations that have similar privacy, security, regulatory considerations, etc. [26]. On the one hand, similar privacy, security, and regulatory considerations in the community cloud lead to easier management compared to public cloud, and dividing the total costs among all the community cloud participants induce lower cost compared to private cloud. On the other hand, there are still some pitfalls in the community cloud. Compared to public cloud, the costs are higher. Because two or more organizations sharing the cloud resources, the bandwidth and data storage capacity are limited [24].

Hybrid clouds are more complex than the above three deployment models. It allows a business to take advantage of the cost and scale benefits of public clouds, and data security and privacy benefits of private clouds [27,28]. For more advantages and drawbacks of hybrid cloud, please refer to [24].

We have summarized the definitions, pros and cons of each type of deployment model in cloud computing in Table 2.1.

2.1.3 Cloud Computing Service Models

CSPs provide many services to the users, including infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS) [9]. When it comes to different service models, CSPs provide the users with cloud resources and services at different levels of abstraction, and further, CSPs and customers share different responsibilities. The basic types of cloud service models are shown in Fig. 2.3(a), and the different responsibilities of CSPs and users among different service models are summarized in table Fig. 2.3(b) and Table 2.2.

IaaS provides the infrastructure such as physical or virtual machines and other resources like file-based storage, IP addresses, etc., and the users can use or install any software, OS or composition. The CSP is responsible for running, maintaining and updating the services. IaaS usually adopts usage-based pricing scheme, which allows users to pay as they grow. IaaS provides users with more flexibility than PaaS and SaaS as the users can deploy any software on the operating system. However, it is the users' responsibility to maintaining the operating system at the IaaS level [9,22,23]. Examples of IaaS are Google Compute Engine [29], Windows Azure Virtual Machines [30], and Amazon CloudFormation [31].

PaaS provides computing platforms including operating system, software development frameworks, Web server, etc. The users can rent complex hardware and operating system combinations dynamically to meet the system requirements, which makes it easier to develop business applications over the Internet. However, the draw-

back is the PaaS may lack of flexibility and some of the users' requirements cannot get met [9, 22] . Examples of PaaS are Amazon Elastic Beanstalk [32], Google App Engine [33], and Windows Azure Compute [34].

SaaS provides the users with access to application services installed at a server, where the users do not need to worry about anything including installation or maintenance, instead they can simply access to the software that have been developed and offered as a service over the web. On the one hand, SaaS enables the users to have easier administration, elasticity, accessibility and compatibility; on the other hand, users have no control of the underlying infrastructure [9, 22]. Examples of SaaS are Gmail, Dropbox, and Microsoft Office 365.

2.2 Latency in the Cloud

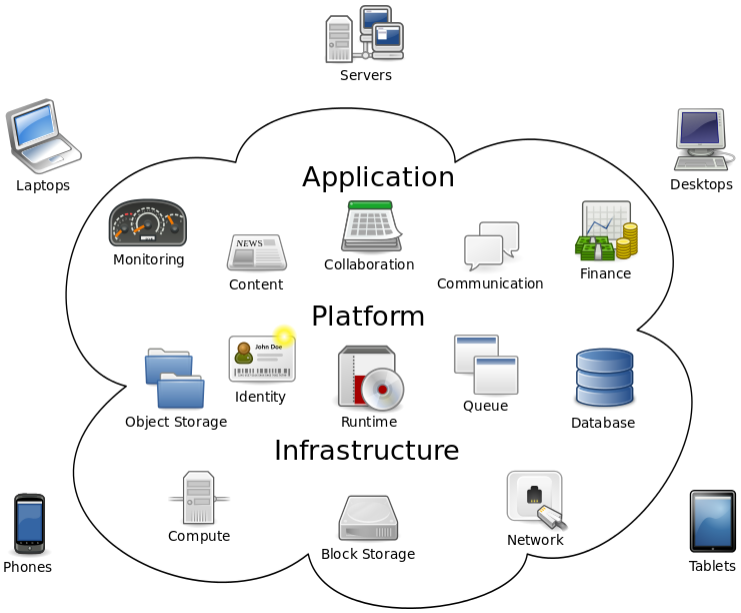
Latency (also called delay) is one of the fundamental network performance metrics [35]. In [35], latency is defined as "how long it takes a message to travel from one end of network to the other", which is strictly measured in terms of time. In [36], latency is defined as "a time delay between the moment something is initiated, and the moment one of its effects begins or becomes detectable". The following three elements are often involved in the latency in a standard network transmission: Propagation, Transmission and Queue Delays. The total latency is defined as follows:

$$\text{Latency} = \text{Propagation} + \text{Transmission} + \text{Queue Delays}$$

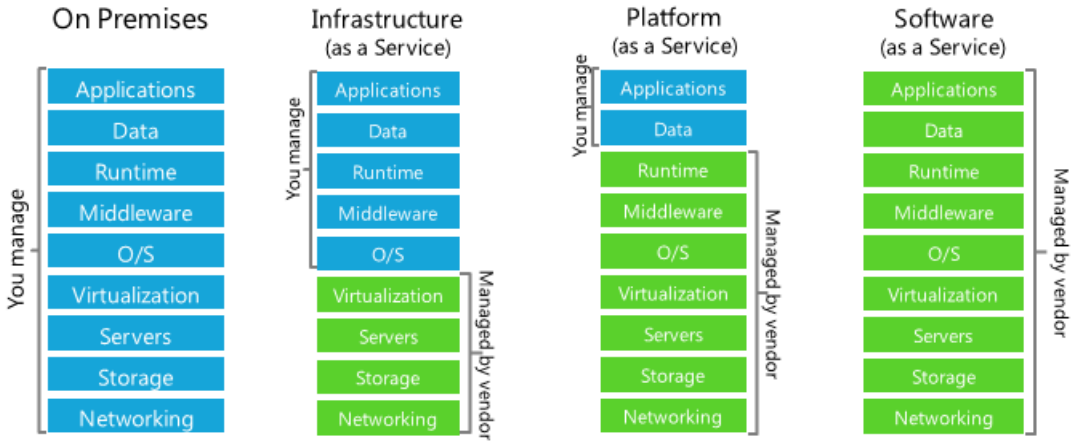
$$\text{Propagation} = \text{Distance} \,/\, \text{Speed of Light}$$

$$\text{Transmission} = \text{Size} \,/\, \text{Bandwidth}$$

where the propagation delay is because nothing can be faster than light, and the bandwidth is defined as "the number of bits that can be transmitted over the network in a certain period of time" [35]. In cloud storage, the latency considered in the literature is the sum of the queuing delay and service time [37, 38].



(a) Cloud service models



(b) Basic difference between service models

Fig. 2.3.: Cloud Service Models and Associated Difference

There have been some research studies that examined the latency impact on the website performance. According to [39], "For every second of latency over normal expectations of that page, a Web transaction accumulates a demerit", it also discusses the risk of losing revenue because of site abandonment. In one of the earliest studies on the "system response time" to a user's command or inquiry in a man-computer

transaction, [40] shows that 10 seconds is the threshold for user's attention, after which the user may get impatient. However, the web user's attention threshold has reduced to 2 seconds because of the development of e-commerce environment [41] [42]. [41] also suggests that the 2-second rule can be also used to determine the hardware and software requirements and investments for service providers, and increase of response time may induce lost revenue and customer satisfaction. For example, Google found that the user's satisfaction will reduce because of an extra half second in the search page generation with consequent 20% traffic drop [43], and Amazon sales will drop by 1% for every 100 ms delay [44]. In order to make the entire experience lightning-quick and smooth and keep users using the web services happily, google works hard to make the web faster.

In [8], cloud computing is regarded as "the achilles heel of cloud", it also shows "every millisecond counts to demanding end-users" and suggests the end-user requirements or demands especially for latency requirements should be put high priority when assessing cloud computing environments and the associated usage. Applications such as algorithmic or high-frequency trading, video streaming, complex web and database services, 3D engineering modeling are in that category are demanding and require low latency [45].

2.3 Pricing in the Cloud

Pricing is an essential component in the cloud market because it not only affects the CSP's profits but also the users' budgets [46]. The success of cloud computing in the IT market can be achieved only by developing adequate pricing techniques [10]. Besides Service Level Agreement (SLA), pricing model should also get approved by the CSP and users.

In this section, we mainly focus on analyzing the common factors that affect the pricing, and pricing schemes exploited by the three leading public cloud service

providers: Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP).

2.3.1 Common Factors that Influence Cloud Pricing

Cloud pricing varies depending on many factors, including the instance types you are choosing, the location of the data center, the operating system and software, etc. In this subsection, we will review the most pertinent factors that impact the pricing in cloud computing [21].

- Initial investment/cost: the amount of money that a CSP will spend to buy the computing resources annually.
- Contract time/lease period: the time that the cloud user wants to lease the computing resources from the CSP. In some cases, the longer the user rents for the resources, the higher discounts the user can get. One example is the volume-discount cloud computing pricing scheme (see Table 2.5).
- Rate of depreciation: the rate of financial value loss of the hardware provided by the CSP, which is due to the wear and tear or the development of the new technology.
- Quality of service: the quality that is guaranteed to the cloud users by CPS, including latency, accuracy, privacy, resource availability, etc. The better the quality of servie, the higher the price will be. One example is the cloud storage prices for hot, warm, cool and cold storages (see Table 2.3).
- Cost of maintenance:the amount of money that the CSP will spend to maintain the cloud resources.

2.3.2 Pricing in the Cloud Storage

Cloud Storage is " cloud storage is a service model in which data is maintained, managed and backed up remotely and made available to users over a network" [47]. Hosting companies operate the data centers, and virtualize the resources according to the customer's requirements.

Before introducing the pricing schemes in the public cloud storage, we want to introduce different types of storage services, which are composed of different disks. In [48], three tiers are defined. Specifically:

- Tier 1 (mission critical data): accessible and fast but very expensive (e.g., solid state drive (SSD));
- Tier 2 (nearline): easy to access but a little expensive (e.g., solid state hybrid drive (SSHD));
- Tier 3 (archival and backup): slow to access, but cheap and reliable (e.g., hard disk drive (HDD)).

Figure 2.4 represents the detailed information and compares the characteristics of different tiers. From Tier 1 to Tier 3, the cost is getting lower but the performance including latency, throughput, availability, MTBF and consistency are getting better.

	Use Case	Cost	Latency	Throughput	Availability	Durability	MTBF	Consistency
Tier 1	Performance	$$$$$	Low	High	High	Low	High	Strong
Tier 2	Balance	$$$	Medium	Medium	Medium	Medium	Medium	Strong or Weak
Tier 3	Capacity	$	High	Lower	Low	Extreme	Low	Eventual

Fig. 2.4.: Comparison of different tiers [48]

In order to leverage the SSDs to support high inputs/outputs per second, e.g., transaction processing, batch processing, decision support analysis, but also save

cost, companies want to integrate SSDs into tiered storage architectures, where SSD is integrated on the top of traditional hard disks. It is important to put the most inputs/outputs intensive and latency sensitive data on the SSD tier to maximize the benefits of SSD given the limited capacity of SSD [49].

For public storage, [50] groups all storage services into 4 tiers: hot, cool and cold, where the unit prices and service rates are decreasing. Table 2.3 compares different cloud storage services prices for US standard region Amazon S3, US East for Microsoft Azure, and US region for Google Cloud. Table 2.4 compares the discounted prices for high volume data in hot storage tier, where Amazon S3 provides the lowest price and Google dose not offer any discounts.

2.3.3 Pricing in the Cloud Computing

In cloud computing, VMs are the virtualization of the IaaS resources. The users can run their jobs on their requested VMs with specific requirements. With regard the pricing schemes in cloud computing, there are many types. However, the top three most popular pricing models employed in cloud computing are: auction-based (or market-dependent) pricing, volume-discount pricing and pay-as-you-go (or usage-based) pricing. Pay-as-you-go pricing is user pay a fixed price per unit (per instance per hour) of use [5, 6, 51], which is public so that the user is aware of the exact cost to be paid.

When the user uses the cloud resources extensively, it is cheaper to exploit volume-discount pricing scheme, which charges a discounted unit price if the user's usage exceeds a given threshold. For example, Google Cloud Platform provides sustained use discounts: if a user uses a VM for 50% of one month, he will get an effective discount for 10%. The longer time he rent, the higher discount he will get, which can get up to 30% [52].

Pay-as-you-go pricing and volume-discount pricing schemes are two ways for CSPs to stabilize user's demand. Auction-based pricing introduces more freedom to respond

to the demand in the market. One example is the spot instance provided by Amazon Elastic Compute Cloud (EC2). Users bid for the instances and Amazon set a dynamic threshold to accept the users' bids. Although the unit price for spot instance is the lowest compared to pay-as-you-go pricing and volume-discount pricing, the job run on spot instance can be interrupted before completion [53]. We have summarized the characteristics of different pricing models in cloud computing in Table 2.5.

2.4 Summary

In this chapter, we provided an overview of the technical view of cloud, including the definition of cloud computing, cloud computing deployment models and service models, Latency in the cloud, and pricing schemes in the cloud computing service market. This chapter help us understand more of cloud computing definitions and why latency and pricing are important in the cloud market. The following two chapters study latency-aware pricing from a CSP's and a user's point of view respectively in the cloud market.

Table 2.1.: Comparison of Cloud Computing Deployment Models

Deployment Models	Definition	Pros	Cons
Public Cloud	Cloud services are made available the general public groups, owned by a third organization.	Data availability and continuous uptime; 24/7 technical expertise; On demand scalability; Easy and inexpensive setup; No wasted resources.	Lack of reliability, data security and privacy.
Private Cloud	Cloud services are provided only to the users inside of the organization or its partners.	Control of data and information; Data security and privacy.	Finite scalability, Higher cost than public cloud
Community Cloud	Cloud services are exclusive to two or more organizations that have common privacy, security, regulatory considerations, etc.	Cheaper than private cloud; Easy management.	Higher costs than public cloud; Limited bandwidth and capacity.
Hybrid Cloud	Combination of two or more cloud models (private, public or community cloud).	Data security and privacy; Cost-effective; On-demand scability;	Lack of security and visibility; Data integration challenge.

Table 2.2.: Cloud Service Models Comparison

Service Models	CSP's Responsiblity	Example
IaaS	Offer computers as physical or virtual machines; Run and maintain the service.	Google Compute Engine [29], Windows Azure Virtual Machines [30], Amazon CloudFormation [31]
PaaS	Offer and maintain computing platform including operating systems, hardware, servers, and databases.	Amazon Elastic Beanstalk [32], Google App Engine [33], Windows Azure Compute [34]
SaaS	Install, operate and maintain software applications in the cloud.	Gmail, Dropbox, Microsoft Office 365

Table 2.3.: Comparison of per-GB prices for the first 1 TB stored in Amazon S3, Microsoft Azure and Google Cloud Storage

		Amazon Web Services	Microsoft Azure	Google Cloud Storage
Hot	Amazon S3 Standard Microsoft Azure Blob Hot Tier Google Cloud Storage standard	$0.023	$0.023	$ 0.026
Cool	Amazon S3 Standard I/A Microsoft AzureBlob Cool Tier Google Cloud Storage Nearline	$ 0.0125	$0.0125	$ 0.01
Cold	Amazon Glacier Microsoft Azure Blob Archive tier Google Cloud Storage Coldline	$ 0.004	$0.002	$ 0.007

Table 2.4.: Volume Discount Prices for Hot Storage Tier

	Amazon Web Services	Microsoft Azure	Google Cloud Storage
2-50 TB/month	$0.0230	$0.0184	$0.026
50-500 TB/month	$0.0220	$0.0177	$0.026
500+ TB/month	$0.0210	$0.017	$0.026

Table 2.5.: Current cloud computing pricing in leading companies

Scheme	Job Runtime	Unit Price	Typical Job Types	Leading Companies
Auction-based	Interruptible No guaranteed deadline	Lowest	Word counting Multimedia processing	Amazon EC2 spot instance
Volume-discount	Long-term	Discounted	Real-time web service Remote monitoring	Amazon EC2 reserved instance Microsoft Azure monthly plans Google Cloud Platform
Pay as you go	Short-term	Highest	Scientific computing Backend batch processing	Amazon EC2 on-demand instance Microsoft Azure Google Cloud Platform

3. TIERED CLOUD STORAGE VIA TWO-STAGE, LATENCY-AWARE BIDDING

In cloud storage, the digital data is stored in logical storage pools, backed by heterogeneous physical storage media and computing infrastructure that are managed by a Cloud Service Provider (CSP). One of the key advantages of cloud storage is its elastic pricing mechanism, in which the users need only pay for the resources/services they actually use, e.g., depending on the storage capacity consumed, the number of file accesses per month, and the negotiated Service Level Agreement (SLA). To balance the tradeoff between service performance and cost, CSPs often employ different storage tiers, for instance, cold storage and hot storage. Storing data in hot storage incurs high storage cost yet delivers low access latency, whereas cold storage is able to inexpensively store massive amounts of data and thus provides lower cost with higher latency.

In this chapter, we address a major challenge confronting the CSPs utilizing such tiered storage architecture - how to maximize their overall profit over a variety of storage tiers that offer distinct characteristics, as well as file placement and access request scheduling policies. To this end, we propose a scheme where the CSP offers a two-stage auction process for (a) requesting storage capacity, and (b) requesting accesses with latency requirements. Our two-stage bidding scheme provides a hybrid storage and access optimization framework with the objective of maximizing the CSP's total net profit over four dimensions: file acceptance decision, placement of accepted files, file access decision and access request scheduling policy. The proposed optimization is a mixed-integer nonlinear program that is hard to solve. We propose an efficient heuristic to relax the integer optimization and to solve the resulting nonlinear stochastic programs. The algorithm is evaluated under different scenarios and with different storage system parameters, and insightful numerical results are

reported by comparing the proposed approach with other profit-maximization models. We see a profit increase of over 60% of our proposed method compared to other baseline algorithms in certain simulation scenarios.

3.1 Introduction

3.1.1 Motivation

The demand for online data storage is increasing at an unprecedented rate due to growing trends such as cloud computing, big data analytics, and E-commerce activities [54], and recently by the rise of social networks. Cloud storage service is now provided by multiple cloud service providers (CSP) such as Amazon's S3, Amazon's Cloud drive, Dropbox, Google Drive, and Microsoft Azure [55]. For instance, Amazon S3 offer 3 major storage classes for different use cases: i) Amazon S3 Standard for general-purpose storage of frequently accessed data; ii) Amazon S3 Standard for Infrequent Access for long-lived, but less frequently accessed data, and iii) Amazon Glacier for long-term archive, while Dropbox has a simple pricing framework, providing two types of storage (Standard and Advanced) for individuals and enterprises, respectively. However, as clients' need for service latency and access frequency may vary significantly over time (e.g., during peak/idle hours or due to bursty demands), existing pricing practice that only offers inelastic service latency at a single, large time-scale would be either inadequate or too expensive to address the dynamics of clients' demands. This motivates our 2-stage pricing mechanism, allowing clients to purchase the storage services that are better aligned with their time-varying requirements.

3.1.2 Market Architecture

We consider a two-stage pricing model, consisting of storage and access latency auctions at different time scales. Specifically, in the first stage, the clients submit

storage bids in order to host their files in the cold and/or hot storage. Once files are accepted into storage, each client is assigned a basic service latency by default. Then, there is a small time-scale auction that runs more frequently in the second stage. During each auction interval, clients with higher latency requirements can place bids for improved service during the interval, thus enabling dynamic, latency-dependent pricing for file access. Using our two-stage bidding platform, we formulate a novel optimization problem, which maximizes CSP's total profit (defined as revenue minus storage costs), while meeting individual client's access requirements. The clients with rejected bids can still access the files, however, the latencies may not be guaranteed.

3.1.3 Challenges

While our two-stage pricing runs on two different time-scales, the storage and latency auction problems are tightly coupled. Not only does first-stage auction (for storage) directly determine which clients would participate in second-stage auction (for latency), the expected outcome of second-stage auction also affects how much clients are willing to pay for storing their files in first-stage auction. In this chapter, we consider the profit optimization problem faced by CSPs under this new pricing mechanism. That is, given the price customers are willing to pay, and the expectation of future access rates, how can a CSP maximizes its overall profit (defined as the total revenue from two auctions minus the necessary storage cost), over a combined decision space, including file storage decisions, file access decisions, and access request scheduling policies. This challenge necessities novel pricing/optimization solutions that go beyond existing approaches such as resource-based pricing, usage-based pricing, time-dependent pricing in cloud computing and online storage.

3.1.4 Contribution

In order to store the files in the *cold storage* or *hot storage*, we propose a systematic framework for two-stage, latency-dependent bidding, which aims to maximize the

cloud storage provider's net profit in tiered cloud storage systems where tenants may have different budgets, access patterns and performance requirements as described in Section 3.3. The proposed two-stage, latency-aware bidding mechanism works as follows. The cloud service provider (CSP) has two tiers of storage: hot storage and cold storage with different service rates. Users can bid for storage and access, in two separate stages, without knowing how the CSP stores the contents. In the first stage (*request for storage*), the user specifies storage size, expected access rates, and latency requirements. If the CSP decides to accept the bid, it will place two copies of data: one in the cold storage and another one in either the hot storage or cold storage. In the second stage (*request for access*), the CSP can decide whether to accept the access requests based on the bid and where to retrieve the files from to meet the access latency requirements. The second-stage auction runs on a shorter time scale (every hour) and the first-stage auction runs on a longer time scale (every day) since the access pattern of files changes faster.

The second-stage decision inherently depends on the first-stage decision. For example, if the CSP decides to store both the original file and its copy in cold storage, the file can be accessed from the cold storage only. However, if accessing from cold storage does not meet the access latency requirement (storage servers might get congested due to high request arrival rates and low service rates), the CSP may not be able to serve the request at once. In this case, the CSP will lose profit due to the loss of the access bids from the users. The optimal first-stage decision decision inherently depends on the second-stage decision. For example, if a user bids at a low price for storage, the file may be stored in the cold storage; however, the user may then bid at a higher price with lower latency requirement in the second stage. In that case, its bid may not be accepted as the latency requirement may not be matched because the file was stored in the cold storage in the first place. Unfortunately, the access bids, latency requirements, and the access arrival rates all are random variables, and the realization of these random variables are not known beforehand.

We first formulate the second-stage decision problem whether to accept the bids and scheduling decision (whether to access the file from the cold or hot storage) given the first-stage decision as an integer programming problem with non-convex constraints. Since the second stage parameters are random, we consider multiple random realizations of these variables and average the objective function over these realizations (or scenarios). We then formulate the first-stage decision problem as a deterministic equivalent program where we maximize the profit from the storage and the expected second stage profit while satisfying the latency requirements for each scenario (Section 3.4). However, the problem again turns out to be an integer programming with non-convex constraints. We first relax the integer constraints by using sigmoid function as the penalty, which closely matches the required penalty function. The relaxed problem is smooth and we can obtain a local solution using the KKT conditions. The solution of the relaxed problem is then converted to the nearest integers. Because of the sigmoid function, the solution attained by the relaxed problem and the feasible one is quite close. In Section 3.6, we show the strength of our proposed method in achieving significantly higher profit as compared to the other algorithms which do not consider the second stage recourse decision while taking the first-stage decision.

Our solution exploits a number of key design tradeoffs. First, any efficient cloud storage and access strategies must meet both the service provider's constraints and customers' requirements. The constraints from the service provider might come from tiered cloud storage architecture, storage-related costs, reliability level and capacities of each tier of storage. The requirements from customers include bidding prices of storage and access, latency requirements and expected access request arrival rates. Second, while placing as much content as possible in cold storage could potentially reduce storage cost, it may be insufficient to meet clients' latency requirements. On the other hand, although storing more content in hot storage improves service latency, it results in higher storage price, which might cause customer churn. A solution exploiting this tradeoff is thus necessary to determine the optimal placement (and

duplication strategy) of files in tiered storage. As a result, jointly scheduling all the file access requests to avoid congestion in each storage tier becomes challenging and must take into account the impact of request patterns and access decisions of all clients.

The main contribution in this chapter can be summarized as follows:

1. *Comprehensive future consideration*: This chapter aims to propose a systematic framework that integrates both file storage and file access, which optimizes the system over four dimensions: file acceptance decision, placement of accepted files, file access decision and access request scheduling. The proposed framework encompasses future access information such as bidding price for access, latency requirements and expected access request arrival rates.

2. *Two-Stage, Latency-Aware Bidding*: The existing pricing schemes do not provide any latency guarantees. We consider a two-stage market–in the first stage, the users bid to store their files in order to achieve an expected latency. The CSP stores the files either in the hot or cold storage depending on the expected latencies required by the users; in the second stage, the users again bid to access the files based on their realized latency requirements. The CSP maximizes the profit by accepting those bids whose latency requirements can be fulfilled.

3. *Computational Efficiency*: We quantify the service latency with respect to both hot and cold storage. The proposed optimization is modeled as a mixed-integer nonlinear program (MINLP), which is hard to solve. We propose an efficient heuristic to relax the integer optimization and solve the non-convex problem.

4. *Insightful Numerical Results*: The performance of the proposed approach is evaluated in various cases. It is observed that the profits obtained from the proposed method are higher than those of other methods, and the access request acceptance rate (ARAR) also dominates that of other methods as the capacity of the cold storage or the service rate of hot storage increases. For example, we see a profit increase of over 60% of our proposed method compared to other baselines as the capacity of cold storage increases beyond 500TB with our simulation scenario.

The rest of the chapter is organized as follows. Section 3.2 describes the related work. The system model for the tiered architecture and the two-stage auction framework is described in Section 3.3, and the two-stage optimization problem is formally defined in Section 3.4. Section 3.5 gives the proposed solution for the mixed integer non-linear program and Section 3.6 validates our proposed policy and evaluates its performance using numerical studies. Finally, Section 3.7 presents our conclusions.

3.2 Related Literature

Tiered storage has been used in many contexts so as to achieve better cost-performance tradeoffs by placing the workload on a hybrid storage that includes multiple hot and cold storage tiers [56–62]. However, the pricing solution for multi-tier cloud storage is quite limited to resource/usage-based pricing, as shown in [63]. Some of the recent pricing schemes for online storage providers including AWS S3, Dropbox, and Google Drive, and their current pricing plans can be found at [7,64,65], respectively. Typically, they often offer a flat price for the storage service with a limited storage capacity or access rates. For example, Amazon provides three types of storage facilities depending on the access rates. However, our model is different from the existing practices. First, we consider a two-stage auction model where in the first-stage, the users can move its file to (tiered) cold/hot storage by adjusting their bids. In the second-stage, the users bid to access the files. Note that the first-stage auction is run once in a day (or week), while the second-stage once an hour (or day). Thus, it provides a greater flexibility to the users to adjust their bids according to their daily requirements. In contrast, the user has to pay a flat rate price for a month if one wants to achieve a faster access rate in the Amazon. Second, in contrast to the pricing mechanisms of Amazon and Dropbox, we consider the latency requirements of the users while accepting the bids even at the first-stage.

Game theory and auctions are broadly adopted as mechanisms for cloud service. For example, in [66], a game-theoretic model is used to induce a truthful cloud storage

selection mechanism where the service providers bid for the quality of service. In [67], an online procurement auction mechanism is proposed to maximize the long-term social welfare. A Vickrey Clarke Grove (VCG) auction-based dynamic pricing scheme is proposed for cloud services in [68]. Recently, a stackelberg game model is proposed in [69] to derive the pricing scheme. The stackelberg game consists of two stages– i) in the first stage, the service provider determines a price which is both time and location dependent, ii) in the second stage, the users decide the schedule of the mobile traffic depending on the prices. However, compared to the above chapters, we consider a scenario where the users bid in a two-stages, two-time scale mechanism – in the first stage, the users bid in order to store their files in the hot and/or cold storage; in the second stage, the users bid for the access latency in various auction interval, given the access arrival request rates. Note that the first problem is inherently challenging even without the second stage parameters because of the integer decision variables and constraints. Further, the CSP in the first stage is unaware of the bids of the clients in the second stage. Thus, the lack of information poses additional challenges to the CSP in profit optimization.

To the best of our knowledge, such kind of auction mechanisms have not been considered in the literature yet. Additionally, the above chapters mainly considered Vickrey-Clarke-Groves (VCG) type auctions [70] or their variants. However, our problem turns out to be a complex non-convex optimization problem. A VCG-type auction will have high complexity and the optimality cannot be guaranteed because of the non-convexity of the problem.

3.3 System Model

3.3.1 Tiered Architecture

We consider a cloud storage provider (CSP) that has a tiered storage architecture. Each file is stored in an inexpensive *back-up* storage facility. For example, Amazon Web service (AWS) charges 0.023 per GB per month for standard storage. The back-

up storage can be considered to consist of hard disk drive (HDD) which is inexpensive, but, the service rate is slow and unreliable. Since it is inexpensive, the latency cannot be guaranteed as a lot of files can be stored. In order to provide a faster service, the CSP can offer two types of storage – i) *cold storage* and ii) *hot storage*. Cold Storage is made of SSHD (combination of solid state drive (SSD) and HDD) which is expensive compared to the HDD, however, the service rate is faster and there is more reliability against disk failure. The hot storage is the most expensive one as it is made of SSD, however, the service rate is also the fastest. Thus, if files are stored in the hot storage, they will have faster access.

3.3.2 Two-Stage Auction Framework

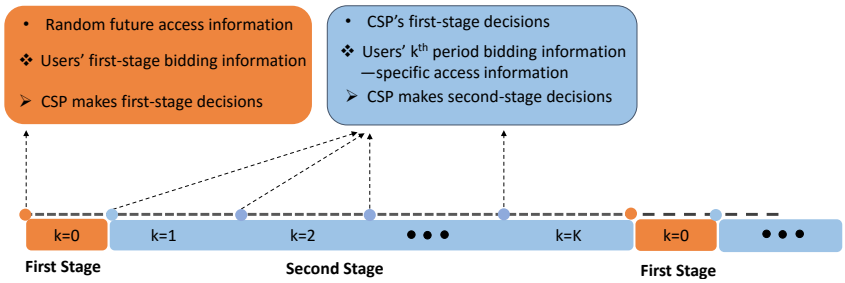


Fig. 3.1.: Two-Stage Auction Framework

In order to store the files in the *cold storage* or *hot storage*, the CSP will operate a market. In the *first stage*, the users [1]bid to store their files in the upgraded storage facilities. The CSP decides whether to accept the file and where to store the original file and its copies. We model the storage platform as providing dual replication of files, so each file has a duplicated copy. [2] To ensure data durability and availability, data replication is broadly adopted by data center storage systems, such as Hadoop

[1]We denote all the clients of cloud service providers as users. Thus, users may be the individuals, enterprises, or organizations

[2]Multiple copies of the file can be created in practice. However, it will increase the storage requirement and the computational complexity of computing the acceptance/rejection of bid, and the access probabilities. The consideration of the scenario where any specific number of copies can be stored is left for future work.

Distributed File System [71], RAMCloud [72], and Google File System [73]. If a file is accepted for storage, the user pays the bidding price, otherwise, it pays nothing. The CSP stores one copy in the cold storage. The CSP also decides whether to store the other copy either in the hot or the cold storage.

In the *second stage*, the users whose files get accepted for storage, bid again for accessing the files. The CSP needs to decide whether to accept the access requests and if accepted, from where the files should be accessed (either cold or hot storage) in order to meet the access latency requirements. If the user's request is accepted, it pays the bidding price, otherwise, it pays nothing. The second stage decision inherently depends on the first stage decision. For example, if the CSP decides to store the both the original file and its copy in cold storage, the file can be accessed from the cold storage only. However, if accessing from cold storage does not meet the access latency requirement (storage servers might get congested due to high request arrival rates and low service rates), the CSP may not be able to serve the request at once. In this case, the CSP will lose profit due to the loss of the access bids from the users. Fig. 3.1 depicts graphically the major considerations in the two-stage problem.

The *optimal* first stage decision of the CSP inherently depends on the second stage decisions. For example, if the CSP decides to store a file in the cold storage because of its low storage bid, it can bid a high value for the access in the second stage. However, the CSP may not accept the bid because of the lower service rate of the cold storage. Hence, the CSP's profit will be reduced. These access bids, the access arrival requests, and the latency requirements are random variables which cannot be known during the first stage decision process, and thus, finding an optimal first stage decision is inherently difficult. We assume that the two bidding stages take place at different time-scales. In particular, while users' files typically remain in the storage system for a long time period (e.g., a day, or several days in Stage 1), the latency-dependent file access decisions (in Stage 2) can be adjusted more frequently on a much smaller time-scale (e.g., every hour), e.g., during busy and off-peak hours. Intuitively, the user's need to access a file changes on a shorter time scale compared

to its storage decision. Hence, the second stage auction must be run more frequently. Note that the frequency of the second stage auction can be changed depending on the change of the access request rates of some of the files.

Note that not all the access bids of the files stored in the cold or hot storage will be accepted. The acceptance depends on the access bids and the latency requirements. However, it is still useful for the user to participate in the first stage, *i.e.*, paying a higher price to store its file in the cold or hot storage. This is because the user may have to access the file only for a certain number of hours in a day, the user can participate in the first stage auction where its files will be stored either in the cold storage or the hot storage at the start of the day. When the user needs to access the file, it bids in the second stage auction. Note that since both the cold storage and the hot storage have higher service rates as compared to the back-up storage, the users can access files at a much faster rates compared to the traditional back-up storage even if their access bids are not accepted at all.

Also note that we have a back-up storage for all the files. Initially, all the files are stored in back-up storage. The users then bid in order to store their files slightly faster cold storage or the fastest hot storage. After this fist-stage auction, the files that are accepted will be copied and moved to store in cold or hot storage. However, the rest of files will be stored in the back-up storage. If the user's storage bid is rejected, she will still be able to access those files from the back-up storage. Our second stage bidding is only designed for premium data access, while a standard, basic service to access data is provided to all files stored in the system. If a user's access bid is not accepted by CSP in the second stage, she still be able to access the file from hot or cold storage. *Thus, service availability is indeed guaranteed.* However, there will be no guarantee on the latency or the speed of accessing the files in the above two cases.

3.4 Problem Formulation

In this section, we formally define the two-stage optimization problem.

3.4.1 First-Stage Decision

In the first stage, the CSP decides – i) whether to store a file or not, ii) if it decides to store the file whether to keep the duplicated copy of the file in the hot storage or cold storage (Original copy of an accepted file is always stored in cold storage). [3] Let I be the total number of files participate in the first-stage auction.

We consider a first price auction where the user pays the price it bids. This auction is run once in a day or once in a week. Let $A_i = 1$ denote that the file $i = 1, \ldots, I$ is accepted for storage; $A_i = 0$ if it is not accepted. Let $R_i = 1$ denote that the copy of the file $i = 1, \ldots, I$ is stored in the hot storage; otherwise, $R_i = 0$. Note that if $A_i = 0$, then the file is not stored anywhere, thus, $R_i = 0$. However, if $A_i = 1$, R_i can be either 1 or 0. Nonetheless, if $R_i = 1$, A_i must be 1.

Also note that if $R_i = 0$, there are two possibilities: (i) $A_i = 1$, thus, both of the original and duplicated copies will be stored in cold storage (hence the number copies of file i stored in the cold storage is 2); or, (ii) the file storage bid is rejected ($A_i = 0$). *Therefore, the number of copies of file i stored in the hot storage and cold storage is R_i and $2A_i - R_i$ respectively.*

Let S_i be the size of the file i and C_j be the capacity of storage j, where $j = 1$ denotes the cold storage and $j = 2$ denotes the hot storage. Since the total stored files cannot exceed the capacity,

$$\sum_{i=1}^{I} S_i(2A_i - R_i) \leq C_1 \tag{3.1}$$

$$\sum_{i} S_i R_i \leq C_2 \tag{3.2}$$

Since A_i must be 1 if R_i is 1,

$$A_i - R_i \geq 0, \quad \forall i \tag{3.3}$$

[3] Note that we consider storing the first copy in cold storage due to its relatively low cost and large capacity, while the analysis and optimization problem can be easily modified if it is replaced by any other type of storage tier.

3.4.2 Second Stage Decision Problem

After storing the files, the users bid for accessing the files in T different epochs. While bidding, the user also gives the access request arrival rates and the latency requirements in each epoch. This market is run on a shorter time scale (*e.g.*, the duration can be an hour or half an hour). The user can update its bid at different time slots depending on its requirements.

The access request arrival rates, the access bid prices, and the latency requirements are *random variables*, which are governed by the user's requirements. We assume that the random variables can be modeled by K realizations of the random variables (or K scenarios). The decision process is time-dependent and runs on 2 different time-scales. The second stage runs at epochs $t = 1, 2, \cdots, (T-1), T$ (e.g., every hour), and different scenarios, (i.e., the user's bids, average latency, and request arrival rates) can vary over epochs. We make access decisions at each epoch based on the bids. The first stage runs at every T periods, (e.g., every day), and we make storage decisions by considering the possible (stochastic) scenarios and the associated probabilities across the T periods.

The notion of scenarios comes from the fact that while the CSP may not be aware of the exact second stage bidding parameters, estimating the distribution of these parameters is possible, e.g., from past bidding history and traffic patterns. In the first stage, the CSP decides the placement of files in hot/cold storage. Files in hot storage are expected to be accessed more frequently, thus providing lower latency at higher storage cost, whereas cold storage results in high latency with more amicable cost. To exploit this tradeoff, since the exact bids only become available in the second stage auction and can vary over time epochs, CSP in the first stage auction needs to estimate the statistics of the second stage bidding parameters, in order to decide the placement of storage files.

Workloads for accessing data follow some pattern [74–76]. However, the CSP is unaware of the exact joint distribution function of the bidding prices, access ar-

rival request rates, and the latency requirements. However, in the scenario-based approach, we do not need to know any specific distribution function. Specifically, we can generate the empirical distribution from the bidding history. For example, from Fig. 3.1 we know that the first stage auction runs in a larger time scale (e.g., every day) and second stage auction runs in a shorter time scale (e.g., every hour). Then in the following day, the CSP can learn the (joint) empirical distribution of bid price, latency and arrival rate based on the access information from the last (few) day(s).

For each scenario $k = 1, \ldots, K$, we denote the latency requirement of file i as l_i^k, the access bid price as q_i^k, and the access request arrival rate as λ_i^k. These representative scenarios can be constructed from the past history of the user's bidding/demand data, and their probabilities (p^k for scenario k) empirically estimated.

Access Arrival rates

The access requests are independent and in a certain time slot, the number of these requests are integer and can be considered independent of the past requests. It is often assumed that the inter arrival time follows exponential distribution [55, 77]. Thus, we consider a Poisson arrival process. We also assume that a single pool of network resource are used to serve each storage (hot and cold), which is a general assumption in the literature [58, 78]. The distribution of service time from each storage tier follows a general distribution with mean service time $1/\mu_j$ (s/Mb) for storage $j \in \{1, 2\}$. We note that even though the file requests are Poisson, an optimal scheduler must simultaneously consider queue states at all storage servers, and the analysis of service latency in turn depends on the scheduling strategy. The optimal scheduling policy in this case is an open problem with no closed form solution, even for the simple case of minimizing the average latency for a single file since the coupling of scheduling and queue states leads to a state explosion problem in analysis [55]

Access Request Acceptance

The CSP decides whether to accept the bid of the access request of each and every file. Let H_i^k denote the decision that whether the file i is accepted in scenario $k \in \{1, \ldots, K\}$. $H_i^k = 1$ indicates that the access bid is accepted; $H_i^k = 0$ indicates that the access bid is rejected.

Note that when the second stage decision is taken, the first-stage decision variables A_i and R_i are known. If $A_i = 0$, then $H_i^k = 0$ for all $k \in \{1, \ldots, K\}$ since file i is not stored in the cold or hot storage, then its access bid cannot be accepted. On the other hand if $A_i = 1$, H_i^k can be either 0 or 1. This is because even if $A_i = 1$, it cannot be guaranteed that the access bid will be accepted in scenario k. The access bid will be accepted based on how much profit will be made and whether the latency requirement can be satisfied by accepting the bid. Hence,

$$H_i^k \leq A_i \quad \forall i. \tag{3.4}$$

Probabilistic Scheduling

Since the optimal scheduling strategy depends on queue states, obtaining a closed form expression for access latency is an open problem. Thus, we consider a feasible approach, known as the probabilistic scheduling, which was proposed in [37,55]. Bounds on mean latency and tail latency probability for cloud storage have been provided in [55] and [79], respectively. The approach have been further used in distributed storage systems in [80–83]. The probabilistic scheduling based approaches have been successfully applied in display ad allocation problem on the Internet [84] and high–aggregate bandwidth switches [85]. We also note that this scheduling approach has been shown to be optimal for achieving the tail-index (defined as the exponent at which the latency probability diminishes to zero) for cloud storage in [86].

Recall that if file i is accepted for storage, the original copy would be stored in cold storage and the duplicated copy will be either stored in cold storage ($R_i = 0$) or in hot storage ($R_i = 1$). As we have copies of a file in both hot and cold storage in

the latter case ($R_i = 1$), the CSP needs to decide where the file should be accessed according to its bidding price and latency requirement. In probabilistic scheduling, each request for file i has a certain probability to be scheduled to each storage j. For the k-th scenario, we have to decide $0 \leq \pi_{i,j}^k \leq 1$ which denotes the probability that the file i will be fetched from storage j, $j \in \{1, 2\}$ for the k-th scenario. Intuitively, $\pi_{i,j}^k$ denotes how often the file i should be fetched from storage j for scenario k. Needless to say, if $R_i = 0$, then $\pi_{i,2}^k = 0$ for all $k \in \{1, \ldots, K\}$. Hence,

$$0 \leq \pi_{i,2}^k \leq R_i, \quad \forall i. \tag{3.5}$$

Further, $\pi_{i,j}^k = 0$ for files which have not been accepted for access requests. Thus,

$$\sum_{j=1}^{2} \pi_{i,j}^k = H_i^k, \quad \forall i. \tag{3.6}$$

Recall that λ_i^k denotes the access request arrival rate of file i in scenario k within a slot. Thus, the total expected file access request rates for file i to storage j in the k-th scenario within the slot is given by $\lambda_i^k \pi_{i,j}^k$. The total expected file access request rate to storage j must be less that the file service rate (in Mb/s) of storage j; otherwise, the queue length will be ∞ and the storage j cannot handle requests. Hence,

$$\sum_i \lambda_i^k \pi_{i,j}^k S_i < \mu_j, \quad \forall j. \tag{3.7}$$

Latency Analysis

Definition 3.4.1 *Latency is the sum of the time a file access request spends in the queue for service (waiting time) and the service time.*

The users strictly prefer a low latency. Studies show that in internet application even 0.1s increase in the latency can significantly reduce the profit [87]. The latency for file i will inherently depend on the probabilistic scheduling decision $\pi_{i,j}^k$, arrival rate λ_i^k, and the service rate of the storage μ_j. Note that the service rate for hot storage is higher than that of cold storage, thus resulting in lower latency and yet

higher cost under the same arrival rate. In the following, we provide the expression for the expected latency of a file. Before that, we introduce a notation which we use throughout.

Definition 3.4.2 *Let $\bar{T}_i^k, k = 1, \ldots, K$ denote the expected latency for file i request at scenario $k \in \{1, \ldots, K\}$.*

Let Q_j^k denote the waiting time at storage $j, j = 1, 2$ for scenario k. Recall that $\pi_{i,j}^k$ denotes the probability with which file i will be fetched from storage j in scenario k. Hence, the expected waiting time for file i at scenario k is $\sum_j \pi_{i,j}^k \mathrm{E}[Q_j^k]$. Recall that μ_j is the service rate in Mb/s for storage j. Since the size of the file i is S_i and the probability that the request for file i will be sent to storage j at scenario k is $\pi_{i,j}^k$, the expected service time for file i in scenario k is

$$\sum_j \frac{\pi_{i,j}^k S_i}{\mu_j} \tag{3.8}$$

From Definition 3.4.1 we have

$$\bar{T}_i^k = \sum_j \pi_{i,j}^k \mathrm{E}[Q_j^k] + \sum_j \frac{\pi_{i,j}^k S_i}{\mu_j} \tag{3.9}$$

We note that the probabilistic scheduling makes it possible to quantify $\mathrm{E}[Q_j^k]$ in closed form, thus enabling us to solve the 2-stage pricing optimization. This is because the file request is Poisson and due to probabilistic scheduling, the superposition of requests from multiple files is superposition of independent Poisson processes, which is also Poisson. Thus, the overall request arrival at each storage tier is Poisson, with a general service time. Thus, M/G/1 results can be used with a proper analysis of the request and service time distributions accounting for multiple files with different arrival rates and different file sizes. The next result characterizes $\mathrm{E}[Q_j^k]$.

Theorem 3.4.1 *The mean waiting time at storage j for scenario k, $\mathrm{E}[Q_j^k]$ is given as follows.*

$$\mathrm{E}[Q_j^k] = \frac{\sum_i \lambda_i^k \pi_{i,j}^k S_i^2}{\mu_j(\mu_j - \sum_i \lambda_i^k \pi_{i,j}^k S_i)} \tag{3.10}$$

Proof In order to simplify notations, we introduce three auxiliary functions: $f = \sum_i \lambda_i^k \pi_{i,j}^k S_i$, $g = \sum_i \lambda_i^k \pi_{i,j}^k$, and $h = \sum_i \lambda_i^k \pi_{i,j}^k S_i^2$.

Using the moments of the service time given in Appendix A.1, we have $\Lambda_j^k = g$, $\mathrm{E}[X_j^k] = \frac{f}{\mu_j g}$, and $\mathrm{E}[(X_j^k)^2] = \frac{2h}{\mu^2 g}$. Using Pollaczek-Khinchin formula for M/G/1 queues [88], we have $\mathrm{E}[Q_j^k] = \frac{h}{\mu_j(\mu_j - f)}$. Expanding the terms, we get the result as in the statement of the Theorem. ■

By substituting (10) into (9), we have

$$\bar{T}_i^k = \sum_j \pi_{i,j}^k \left(\frac{\sum\limits_i \lambda_i^k \pi_{i,j}^k S_i^2}{\mu_j(\mu_j - \sum\limits_i \lambda_i^k \pi_{i,j}^k S_i)} \right) + \sum_j \frac{\pi_{i,j}^k S_i}{\mu_j} \tag{3.11}$$

Note that by differentiating twice one can easily discern that $\bar{T}_i^k$ is convex in each $\pi_{i,j}^k$. However, $\bar{T}_i^k$ is jointly *non-convex* in $\pi_{i,j}^k$. This is because of the terms $\pi_{i,1}^k \pi_{i,2}^k$, which is not jointly convex in $\pi_{i,1}^k$ and $\pi_{i,2}^k$.

Note that the latency depends on the file size S_i: if the file size is large, the latency will be large. Thus, it shows that for the same access bid the files of smaller sizes will be preferred (given that its latency requirement is satisfied) as it will allow the CSP to accept more access requests. Also note that if $\lambda_i^k \pi_{i,j}^k$ is large for some j, then the latency again increases, hence, the latency of storage facility j increases if too many requests are directed towards j. Thus, the CSP has to judiciously select $\pi_{i,j}^k$. If a large number of requests are directed towards the hot storage, the latency requirement may not be satisfied which may decrease the CSP's profit. Also note that $\bar{T}_i^k = 0$ if the file is not accepted for accessing. Recall that the latency requirement for file i in scenario k is l_i^k. Hence, we must have

$$\bar{T}_i^k \leq l_i^k \tag{3.12}$$

Second Stage Optimization Problem

The second stage profit of the CSP if the scenario $k \in \{1, \ldots, K\}$ is realized is given by

$$\sum_i q_i^k H_i^k \tag{3.13}$$

Recall that q_i^k is the access bid for file i in scenario k. Hence, the second stage optimization problem if scenario k is realized is given by

$$\begin{aligned}
\text{(P2)} \quad \text{maximize} \quad & \sum_i q_i^k H_i^k \\
\text{subject to} \quad & (3.4), (3.5), (3.6), (3.7), (3.11), (3.12) \\
& H_i^k \in \{0, 1\} \quad \forall i && (3.14) \\
& \text{var} \ : H_i^k, \pi_{i,j}^k && (3.15)
\end{aligned}$$

Note that if a user bids high for access, but its size is large or the arrival rate is high, then the latency (3.11) may increase and the CSP will lose the profit as the CSP may satisfy only few requirements of latencies. Problem (P2) is a nonlinear integer program, which is difficult to solve.

3.4.3 Deterministic Equivalent Program

Now, we formally formulate the first-stage stochastic program. Let P_i be the bid price of file i for storage. Let c_1 and c_2 denote the total cost incurred by the CSP for storing file i in the hot and cold storage, respectively. Recall that $j = 1$ ($j = 2$, resp.) denotes that the storage is cold (hot, resp.). Hence, the profit obtained by the CSP for *storage* is

$$\sum_i P_i A_i - \sum_i S_i (2A_i - R_i) c_1 - \sum_i S_i R_i c_2 \tag{3.16}$$

Since the second stage decision variables inherently depend on the first stage and the CSP wants to maximize the total profit, thus, the CSP needs to consider the second

stage decision while taking the first stage decision. Hence, the first stage decision problem is different from the standard knapsack problem.

Note that the CSP knows that the access bid price for file i in scenario k is q_i^k. Recall that the probability with which scenario k is generated is p^k. Hence, the expected profit from the second stage decision is

$$T \sum_{k=1}^{K} \sum_{i=1}^{I} p^k q_i^k H_i^k \tag{3.17}$$

T is the total number of slots where the access auctions are run. In the first stage, the CSP wants to maximize the total expected profit. However, the expected profit also depends on the second stage decision variables. Therefore, we should find H_i^k and $\pi_{i,j}^k$ for each possible scenario. We formulate the first-stage decision problem as the so-called *deterministic equivalent program* [89] in the following:

$$\text{(P1)} \quad \text{maximize} \quad \sum_i P_i A_i - \sum_i S_i(2A_i - R_i)c_1 - \sum_i S_i R_i c_2 \tag{3.18}$$

$$+ T \sum_k \sum_i p^k q_i^k H_i^k \tag{3.19}$$

$$\text{subject to} \quad (3.1) - (3.3), A_i \in \{0,1\}, R_i \in \{0,1\}$$

$$\bar{T_i}^k \leq l_i^k, \quad \forall i, \quad \forall k \tag{3.20}$$

$$\sum_i \lambda_i^k \pi_{i,j}^k S_i < \mu_j, \quad \forall j, \quad \forall k \tag{3.21}$$

$$H_i^k \leq A_i, \quad \forall i, \quad \forall k \tag{3.22}$$

$$\sum_j \pi_{i,j}^k = H_i^k, \quad \forall i, \quad \forall k \tag{3.23}$$

$$0 \leq \pi_{i,2}^k \leq R_i, \quad \forall i, \quad \forall k \tag{3.24}$$

$$H_i^k \in \{0,1\}, \quad \forall i, \quad \forall k \tag{3.25}$$

$$\text{var: } A_i, R_i, \pi_{i,j}^k, H_i^k \tag{3.26}$$

Note that the constraints in (3.20)-(3.25) are for the second stage decisions. Also note that though we solve for $\pi_{i,j}^k$ and H_i^k, the decision variables are of interest in the first stage, which are A_i and R_i. After A_i and R_i are decided, the optimization

problem (P2) is solved if scenario k is realized. In the deterministic equivalent program, the number of scenarios K may be very large which increases the number of constraints and the decision space. One remedy is to discard those scenarios which occur with very low probability.

When considering the first stage decision, the CSP has no knowledge of the exact scenarios that would occur in the small time-scale during later latency-dependent auction. Thus, the CSP will decide whether to accept a bid, and whether to store the file in the cold or hot storage, only based on estimated statistics of all scenarios of the second stage. Note that in the second stage, when the users' latency-dependent bids becomes available, the CSP is aware of the specific scenario for each epoch and will make the second stage decision, such as request scheduling.

Theorem 3.4.2 *Problems (P1) and (P2) are non-convex.*

Proof First, the decision variables A_i, R_i and H_i^k are binary, which make the problem non-convex. Second, $\bar{T}_i^k$ (cf. (11)) has term $\pi_{i,1}^k \pi_{i,2}^k$, which is not jointly convex in $\pi_{i,1}^k$ and $\pi_{i,2}^k$. ■

Hence, standard convex optimization solvers such as CVX, MOSEK or integer linear programming optimization solvers such as CPLEX cannot be used. Some exact approaches have been proposed for MINLPs, such as Spatial Branch-and-Bound [90], Branch-and-Reduce [91], Lagrangian Decomposition [92]. However, the exact algorithms are not practical for large scale problems because of the long running time. Some heuristics have also been presented, such as tabu search [93], genetic algorithm [94], etc. In addition, we have some software packages that can handle non-convex MINLPs, such as BARON, α-BB. However, the above softwares can handle small scale non-convex MINLPs. In this chapter, we propose an efficient heuristic to relax the integer optimization, then any non-convex optimization software can be used to solve the relaxed program. In our case, we use CONOPT, which is a solver for large-scale nonlinear optimization.

Problem (P1) is the first stage problem and (P2) is the second stage problem. Note that the solution to (P1) determines how users' files are placed in hot/cold storage, which in turn determines how users participate in problem (P2). On the other hand, while solving the first stage problem (P1), the CSP needs to consider the distribution of the second stage parameters – the access bids, the arrival rates, and the latency requirements. This is because optimal decisions that will be made in the second stage (and thus, the optimal profit for each epoch) can affect data placement decisions in the first stage. We address this challenge by aggregately considering different scenarios and their estimated probabilities in (P1). Then, in the second stage, the CSP optimizes (P2) for a specific scenario in each epoch, by considering the realized bid prices and the average latency requirements.

3.5 Solution Methodology

3.5.1 Discussion of Computational Complexity

In this subsection, we show that the optimization problems (P1) and (P2) are NP-hard problems.

Theorem 3.5.1 *The optimization problems stated in (P1) and (P2) are NP-hard.*

Proof The proof is provided in Appendix A.2. ■

In order to prove (P1) is NP-hard, we consider a special case where assume that there is a single scenario, and thus the bids for the second stage are perfectly known. Thus, even without any stochastic nature of scenarios, the problem can be shown to be NP-hard. Further, a toy example is provided in Appendix A.2 to illustrate the problem scale.

3.5.2 Integer Relaxation

Problem (P1) and (P2) are non-convex as the variables A_i, R_i and H_i^k are binary. If we relax the binary constraints, the rest of the problem will still be non-convex as the latency function $\bar{T}_i^k$ (cf. (3.11)) is still jointly non-convex in $\pi_{i,j}^k$. However, if we relax the integer constraint then the objective function and the constraints will be differentiable. We can use the solver such as CONOPT [95] to find a locally optimal solution. CONOPT is generally used for smooth continuous functions. The algorithm used in CONOPT solver is based on generalized reduced gradient (GRG) method, which was first proposed by Abadie and Carpentier in 1969 [96], and modified to enable large - scale problems more efficient by Drud in 1985 [97] and 1992 [98]. The idea is to replace the nonlinear objective function and nonlinear constraints by their Taylor approximation at the current value, and then use reduced Gradient method to solve.

However, if we relax the integer constraint, the solution may not be integer, rather a value in the interval $(0, 1)$. To eliminate those solutions, we need to add a penalty function which will put high penalty ($-\infty$ for optimality) when the solution is not either 0 or 1 and 0 penalty when the solution is indeed 0 or 1 (Fig. 3.2).

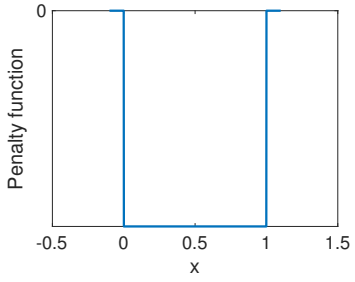


Fig. 3.2.: Desired Penalty function

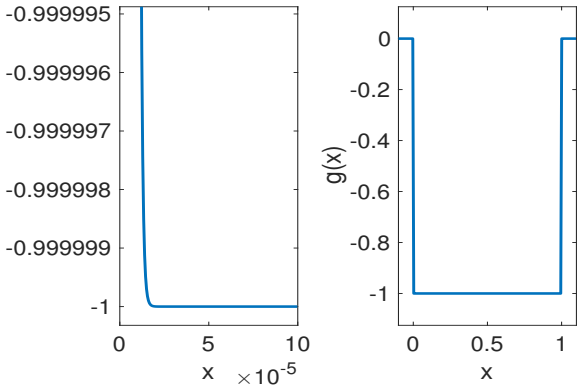


Fig. 3.3.: Penalty function $g(x)$ (cf.(3.27)) for $\alpha = 10^6$. In the left figure $x \in [0, 10^{-4}]$ and in the right hand figure $x \in [-0.1, 1.1]$.

Sigmoid function[4] is a S-shaped function which can closely approximate the step function $U(\cdot)$. Since we have to put zero penalty when the solution is 0 or 1 and a high penalty when it is in between, thus, we consider the following function

$$g(x) = \frac{1}{1+\exp(\alpha x)} - \frac{1}{1+\exp(\alpha(x-1))}. \tag{3.27}$$

Further, let

$$g_1(x) = g(x) + \frac{1}{2} - \frac{1}{1+\exp(\alpha)}. \tag{3.28}$$

Fig. 3.3 shows that $g(x)$ becomes close to -1 at the value $\frac{1}{\alpha}$. Fig. 3.3 also shows that for $\alpha = 10^6$, $g(x)$ closely matches the penalty function that we desire. The function $g_1(x)$ shifts the penalty function to have zero value on the desired extremes and a negative value in the desired range. The next result shows that $g_1(0) = g_1(1) = 0$, with proof in Appendix A.3.

Lemma 1 *The value of function $g_1(x)$ is zero for $x = 0$ and $x = 1$, or $g_1(0) = g_1(1) = 0$.*

Thus, we note that as the function $g_1(x)$ gives 0 penalty when x is 0 or 1. For any $0 < x < 1$, $g_1(x) \to -1/2$ as $\alpha \to \infty$. Further, even for finite α, we note that $g_1(x) < 0$ for $0 < x < 1$. Thus, for large α, this function it will match the ideal penalty function. Note we do not have to lose any differentiability property as $g(\cdot)$ is differentiable. With this penalty function our problem reduces as follows.

[4] $S(t) = 1/(1+\exp(-t))$

$$\text{(P3)} \quad \text{maximize} \quad \sum_i P_i A_i - \sum_i S_i(2A_i - R_i)c_1 - \sum_i S_i R_i c_2 + T\sum_i \sum_k p^k q_i^k H_i^k + \sum_i C(g_1(A_i) + g_1(R_i)) + \sum_i \sum_k C g_1(H_i^k) \tag{3.29}$$

$$\text{subject to} \quad (3.1) - (3.3), (3.20) - (3.25)$$

$$0 \leq A_i \leq 1, 0 \leq R_i \leq 1$$

$$0 \leq H_i^k \leq 1. \tag{3.30}$$

C is the weight corresponding to the penalty functions. Note that the solution will be integer if $C \to \infty$. A_i and R_i are decided by solving (P3). After A_i and R_i are solved for a given realization k, the second stage decisions are taken. In the second stage, the following optimization problem is solved

$$\text{(P4)} \quad \text{maximize} \quad \sum_i (q_i^k H_i^k + C g_1(H_i^k))$$

$$\text{subject to} \quad (3.4), (3.5), (3.6), (3.7), (3.11), (3.12)$$

$$0 \leq H_i^k \leq 1. \tag{3.31}$$

The decision variables are H_i^k and $\pi_{i,j}^k$. Note that we do not need to decide A_i and R_i in the second stage, hence, we do need constraints (3.1)-(3.3). Note that the solution H_i^k will be optimal and integer if $C \to \infty$. Since the problem is non-convex, we cannot guarantee that the solution obtained by CONOPT will be optimal. However, we can infer the following if we find an optimal solution

Proposition 3.5.1 *The optimal solution of the relaxed problem (i.e. (P3), (P4)) is also the optimal solution of the original problem (i.e. (P1), (P2)) as $C \to \infty$.*

Proof We only show the optimality of problem (P1). The optimality proof for (P2) is similar, and hence, it is omitted. If $C \to \infty$, the penalty tends to infinity for any positive α and the objective function will be $-\infty$ in the interval $(0, 1)$ for problem

(P3). Thus, if the solution is non-integer, one can achieve a higher profit by selecting a feasible integral solution. Hence, the solutions A_i, R_i, and H_i^k have to be integers. Thus, the optimal solution must correspond to the optimal solution of the original problem. ■

Note that *ideally*, while solving the first stage problem, second stage solutions are also obtained for each scenario. Thus, a reader may ask the question the necessity of the second stage problem. However, the CSP may obtain non-integer solutions from the relaxed problem in the first stage. Thus, the CSP has to find feasible solutions in the first stage from the solutions of the relaxed problem (P3). Thus, the second-stage optimal solutions may also vary from the obtained solutions of the relaxed problem (P3). Further, we note that in order to reduce the problem complexity, the first stage may use certain samples of scenarios rather than infinite number of scenarios. Thus, in the second stage, we might not have seen all possible scenarios and thus would still need to perform an optimization for the second stage. Thus, we need to solve a second stage problem (P4) again given the obtained solutions from (P3).

3.5.3 Feasible Solution from the Relaxed Problem

When C is ∞, both the first-stage and second-stage decision solutions A_i, R_i, and H_i^k will be integers. If $\alpha \to \infty$, $g(x)$ will match the ideal penalty function. However, in practice, neither C nor α can be set at ∞. Hence, we may find a solution which is not feasible, *i.e.* it is neither 0 or 1. Note that setting α alone to a very high value will not make the solution integer. One also has to make C high to give larger penalty to the fractional solution. However, C has to be larger for smaller α. In the following, we discuss how to find the feasible solution for finite C and α.

Also note that if C is very high, in an optimal solution the solution will only be away from the integral solution by a nominal amount. One can then convert the non-integral solution of either H_i^k, A_i, R_i to the nearest integer. However, the above does not guarantee that the capacity constraints or the latency requirements

will be satisfied. For example, consider that in a solution of the relaxed problem $A_i = 1-\epsilon$, where $\epsilon > 0$ is very small, and $A_i^r = 1$ is the nearest integer solution to the relaxed problem. However, if $\sum_i (2A_i - R_i)S_i = C_1$, then $\sum_i (2A_i^r - R_i)S_i > C_1$ which violates the constraint (cf. (3.1)). Thus, simple converting the solution A_i, R_i, H_i^k of the relaxed problem to the nearest integer may not give a feasible solution. However, in the following, we provide a strategy which can guarantee that even if the solution of the relaxed problem is converted to the nearest integer, then, it will not violate the original constraint.

Proposition 3.5.2 *For every C and α, there exists an $1 > \epsilon > 0$ such that if $C_j = C_j(1-\epsilon)$ and $\mu_j = \mu_j(1-\epsilon)$, such that if the solution A_i, R_i, H_i^k of the relaxed problem (i.e., (P3), (P4)) is converted to the nearest integer (if the value is 0.5, it will be converted to 0) then they will be feasible solution of the original problem (i.e., (P1), (P2)).*

Proof The proof is provided in Appendix A.4. ■

Intuitively, if we make $C_j = C_j(1-\epsilon)$ and $\mu_j = \mu_j(1-\epsilon)$, we solve a restricted problem. Thus, even when we convert the non-integer solutions of the relaxed problem to the nearest integers we will not violate the original constraints. Note that if C is very large, we need a very small ϵ as the solutions of the relaxed problem A_i, R_i and H_i^k will be close to the integers. As $C \to \infty$, $\epsilon \to 0$. C is also larger if α is low. In our numerical results, we set ϵ as 0.001, α as 10^6, and C as 10^9 which gives the feasible solutions as mentioned in the above proposition.

3.6 Numerical Studies

3.6.1 Simulation Setting

To validate our proposed policy and evaluate its performance, we implement the following numerical studies. A new simulation tool we developed is implemented in AML with a time-slotted system to consider the two-stage pricing mechanism. In

order to solve the optimization, GAMS/CONOPT 3 solver was used. Unless stated otherwise, we consider a setting where there are 1,000 files, and the number of slots for the second-stage auction is $T = 20$. The capacities of cold and hot storage are 400 and 200 GB respectively. We consider five types of files: Ty^I, Ty^{II}, Ty^{III}, Ty^{IV} and Ty^V, which are of sizes 64, 128, 256, 512 and 1024 MB respectively. The above five types of files are generated randomly with the same probability. In the first stage, customers will bid for storage. Bidding prices for storage per MB are considered to be a random variable $\sim \mathcal{U}[0.1, 0.3]$, *i.e.,* it is uniformly distributed with a mean of 0.2 cents. Thus, the bid P_i for file i is distributed as $\sim S_i * \mathcal{U}[0.1, 0.3]$. For example, if there is a 64 MB file and the realized price is 0.25 cents for each MB, the bidding price to store this file is $64 * 0.25 = 16$ cents.

We consider $K = 10$ different scenarios for the second-stage parameter. Specifically, we generate 10 different instances of access request arrival rates, access bids, and the latency requirements. We consider Poisson arrivals for file access requests, and the arrival rate λ_i^k is generated independently according to the mean $20, 10, 8, 4$, and 2 per hour for the file sizes of 64, 128, 256, 512 and 1024 MB respectively. This is in accordance with the practice as the smaller size files are accessed more frequently. We assume that the latency requirements l_i^k are related to the file sizes. Specifically, we generate l_i^k independently according to the distribution $\sim \mathcal{U}[30 + \frac{S_i}{5*10^6}, 30 + \frac{S_i}{10^6}]$ in milliseconds. Thus, if the file size is larger, the latency requirement is longer. The authors of [99] showed that the utility is in general convex in the latency and concave in the arrival rates. In this chapter, we let $q_i^k = \frac{50 S_i \log(\lambda_i^k + 1)}{(l_i^k)^2}$, which is convex in latency and concave in the arrival rates. The parameters are described in Table 3.1. After generating the K scenarios we compute the empirical distribution to find the number of times a scenario k (prob. p^k) occurs out of the 10 events. Then scenario is randomly generated among the K scenarios where k-th scenario occurs with probability p^k.

Based on the above specifications, we compare the performances of the proposed method (PM) with three other methods, which are described as follows. We consider

α as 10^6 and C as 10^9 in (P3) and (P4). The factor by which we reduce C_j and μ_j is choisen to be $\epsilon = 0.001$. The solution obtained by the relaxed problem and the proposed method are almost the same. Thus, we do not show the solution of the relaxed problem.

- IS: Problem with Two Independent Stages:
  - Solve the first-stage problem without considering the second-stage recourse decisions to get the first-stage solution A_i and R_i for each i.
  - Given the first-stage solution solve for the realized scenario, *i.e.* solve the second stage optimization problem (P2). We again solve the relaxed version (P4) and then find the optimal solution according to Proposition 4.2 as described in our proposed method
- GH I: Greedy Heuristic Based On q_i^k/S_i:
  - Solve the first-stage problem without considering the second-stage recourse decisions to get the first-stage solution A_i and R_i for each i.
  - In the second stage, we sort the bids based on q_i^k/S_i in the descending order. We keep accepting bids according to the sorted order as long as the realized the latency requirements are met.
- GH II: Greedy Heuristic Based On q_i^k/λ_i^k:
  - Solve the first-stage problem without considering the second-stage recourse decisions to get the first-stage solution A_i and R_i for each i.
  - In the second stage, we sort the bids based on q_i^k/λ_i^k in the descending order if scenario k is realized. We keep accepting bids according to the sorted order as long as the realized the latency requirements are met.

Profit in each algorithm is considered to be the sum of the first-stage and second-stage profits. Note that all the above mentioned base-line algorithms do not solve the

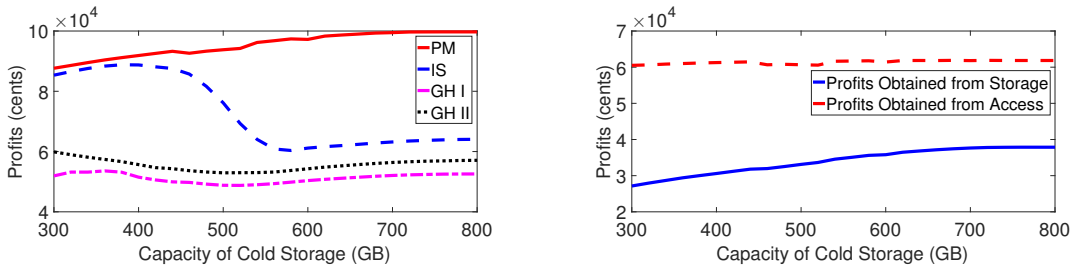
first-stage decision problem by considering the second-stage recourse decision. Algorithm IS solves the second-stage decision problem given the solution of the first-stage decision. However, GH I and GH II are greedy heuristics which accept bids according to some heuristics in order to lower the complexity of finding the optimal solution of the second-stage decision problem. Intuitively, recall from constraint (cf. (3.11)) that the latency of a file in scenario k inherently depends on the access request arrival rates and file size. Specifically, the latency increases as the file size increases or the access request arrival rate increases. Hence, the CSP should prefer the bids which give more profit per unit of the size and the per unit of the access request arrival rate. GH I greedily prefer the bids which pay more per unit of size. On the other hand, GH II strictly prefers the bid which pays more for per unit of access request rate. Before discussing the results, we introduce a notation which we use throughout this section.

$$\begin{aligned} &AccessRequestAcceptanceRate(ARAR) \\ &= \frac{\text{Total Number Of Accessed Files}}{\text{Total Number of Requests}} \end{aligned} \tag{3.32}$$

The above metric shows how much bids are accepted in the second stage among the bids that are accepted in the first stage. This will give an idea pertaining the fairness of the process. In each of the result, each algorithm is run 100 times and an average is taken for the profit and ARAR over these runs.

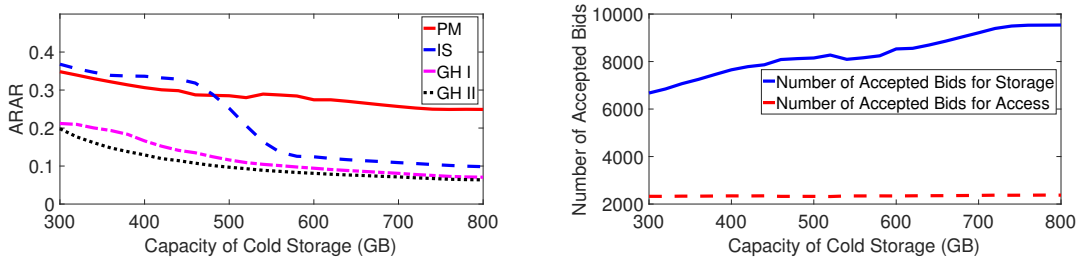
3.6.2 Impact of Storage Capacity

To demonstrate the effectiveness of our proposed heuristic, we fix the hot storage capacity as 200 GB and vary the capacity of cold storage (C_1) from 300 GB to 800 GB in the steps of 20 GB, and plot the total profits and access request acceptance rate (cf. (3.32)) by using different methods. Fig. 4.3(a) shows that as the capacity of cold storage (or ratio C_1/C_2) increases, the profits obtained from all the algorithms except IS increases; however, the rate of increase decreases with the increase in the C_1. Fig. 4.3(d) provides the reason behind this variation. As C_1 increases, more files



(a) Impact of the variation of cold storage capacity on profits

(b) Impact of the variation of cold storage capacity on profits obtained from file storage and access in PM



(c) Impact of the cold Storage capacity on ARAR

(d) Impact of the cold storage capacity on the number of files that are accepted for storage and number of bids accepted for access

Fig. 3.4.: The impact of the variation of Cold storage capacity. The cost of cold storage and hot storage is 50 cents per GB and 80 cents per GB respectively. The service rates of the cold and hot storage are 100Gb/s and 200Gb/s.

can be stored in cold storage which increases profits. However, if C_1 is large enough, no more files can be stored, thus, the profit becomes saturated. Fig. 4.3(c) shows that because of the lower service rate of the cold storage, the profit from accessing the file does not increase with the increase in the capacity of cold storage. This is because files may be stored but cannot be accessed as it violates the latency constraint. *Thus, increasing the cold storage capacity without increasing the hot storage capacity will not fetch more profit after a certain threshold.* Note from Fig. 4.3(a) that the profit achieved by Algorithm IS increases initially, then decreases and again increases as C_1 increases. Intuitively, the Algorithm IS does not consider the second stage decision

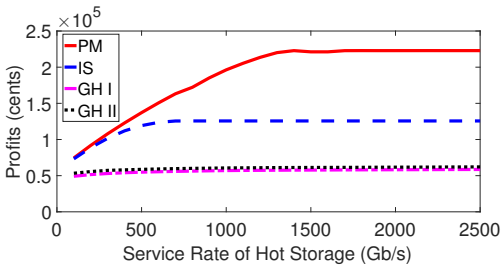
variables in its first-stage decision. Hence, more files are stored in the cold storage as it has a lower cost. However, as almost all the files are stored in the cold storage, the files cannot be accessed fast enough which does not increase the profit from accepting the access bids. Similarly, the profits earned by Algorithms GH I and GH II do not increase much as C_1 increases as they do not consider the second-stage recourse decisions in the first stage. Also note that when C_1 is large, our algorithm outperforms the other base-line algorithms by 50%. This shows the virtue of the consideration of the second stage recourse decision in the first-stage decision.

From the results in Fig. 4.3(c), the Access Req. Acceptance Rate (ARAR, cf. (3.32)) decreases as the capacity of cold storage increases. This is because, by increasing the capacity of cold storage, the number of files accepted for storage increase, however with limited cold storage service rate, the number of files accepted for access is limited (which is also verified by Fig. 4.3(d)). Consequently, the ARAR decreases. Note that the ARAR corresponding to Algorithm IS is higher compared to our proposed method when C_1 is low as vary number of files are stored by the IS compared to our proposed method.

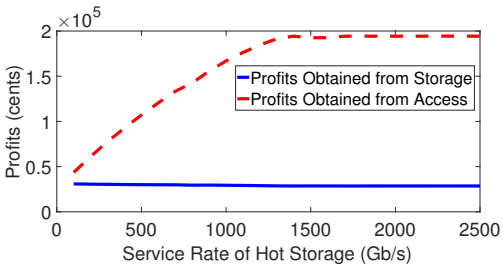
3.6.3 Impact of Service Rate of Hot Storage

In this subsection, we assume that the service rate of cold storage is 100Gb/s, and the service rate of hot storage is varied from 100 to 2500 Gb/s in steps of 100Gb/s. Fig. 3.5(a) shows that the profit increases as the service rate of hot storage increases. This is because more access requests can be accepted as the number of accepted bids increase (Fig. 3.5(d)). Our proposed method outperforms the other methods. In fact, the profit can be increased by 100% compared to the other methods for high service rate.

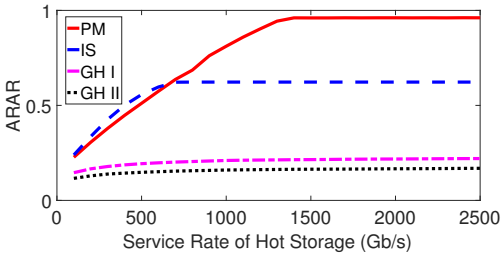
Note from Fig. 3.5(b) that the profit from the second-stage auction increases significantly in our proposed method. However, the profit from the storing files does not increase much. This is because when the service rate is low mostly those files



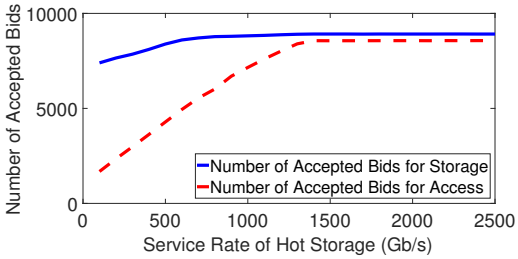
(a) Impact of the hot storage service rate on profits



(b) Impact of hot storage service rate on profits obtained from file storage and access



(c) Impact of hot storage service rate on ARAR



(d) Impact of hot storage service rate on the number of bids accepted for storage and access

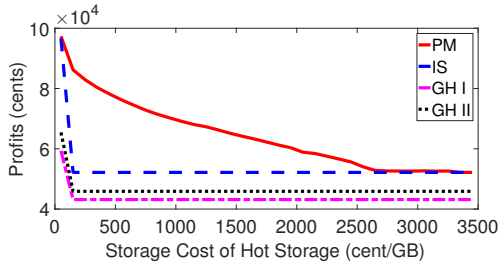
Fig. 3.5.: Profits and ARAR as a function of hot storage service rate. The cost of the cold and hot storage are 50 cents and 80 cents per GB respectively. $C_1 = 400$ GB and $C_2 = 200$GB.

who have lower access requests or lower latency requirements (but can pay more) are accepted. As Fig. 3.5(d) suggests, when the service rate is high, more files are stored, however, the number of accepted access bids increases significantly. This suggests that when the service rate is high, the files which bid lower prices for storage, but still can pay more because of the high access rates are accepted for storing. Hence, the profit from storing the files remains constant as the storage capacities remain constant, however, the profits from accepting bids increase.

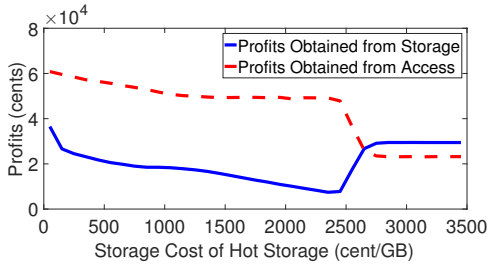
Fig. 3.5(c) shows that the ARAR increases with the service rate of the hot storage for all the algorithms. When the service rate is high, more access bids are accepted, however, the bids accepted for storage remains the same (Fig. 3.5(d)). Hence, the ARAR is high (cf. (3.32)). When the service rate is low, mostly the files those have

lower access requests are stored. However, the IS still can store files which have higher access rates if they pay more because it solves the two stage problem independently. Hence, the IS can achieve more ARAR in this case. However, when the service rate of the hot storage exceeds a threshold, the ARAR attained by our proposed method is the highest. The ARAR attained by the greedy heuristics GH I and GH II are strictly lower compared to our proposed method.

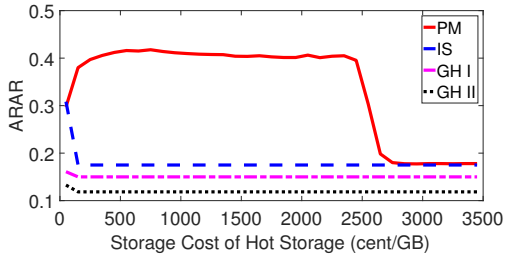
3.6.4 Impact of Storage Cost of Hot Storage



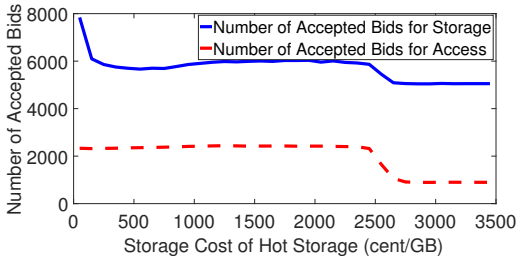
(a) Impact of hot storage cost on profits



(b) Impact of hot storage cost on profits obtained from the file storage and access



(c) Impact of hot storage cost on ARAR



(d) Impact of hot storage cost on the number of accepted bids for storage and access

Fig. 3.6.: Profit and the ARAR as a function of the hot storage cost. $C_1 = 400$ GB and $C_2 = 200$ GB. The service rate of the cold and hot storage are respectively 100Gb/s and 200 Gb/s.

In this subsection, we assume that the storage cost of cold storage is 50 cents per GB, and the storage cost of hot storage is varied from 50 to 3450 cents per GB in the

step of 100 cents per GB. Fig. 4.4(a) shows that as the hot storage cost increases the profit decreases. This is because most of the files are stored in the cold storage which decreases the profit as fewer number of access requests are accepted which is also verified from Fig. 3.6(d). Our proposed method outperforms the baseline algorithms by more than 60% when the storage cost is neither too high nor too low. When the hot storage cost is too low, more files are stored in the hot storage in the first stage and thus, more profit can be attained in the second stage by accepting more access bids. Hence, the profit attained by IS is close to our proposed method when the hot storage cost is low. Note that the profit attained by the IS is also very close to our proposed method when the hot storage cost is high. This is because IS inherently stores more files in the cold storage in the first stage. Since the greedy algorithms GH I and GH II do not optimize the second-stage decision, the profits attained by those are slightly lower compared to the IS.

Note from Fig. 4.4(b) that the profit in our proposed method from accessing the files decreases with an increase in the cost of hot storage as more files are stored in the cold storage. Though the overall profit decreases (see Fig. 4.4(a)), the profit obtained from storage increases. This is because the files which can pay more but do not have low latency requirements can be stored in the cold storage.

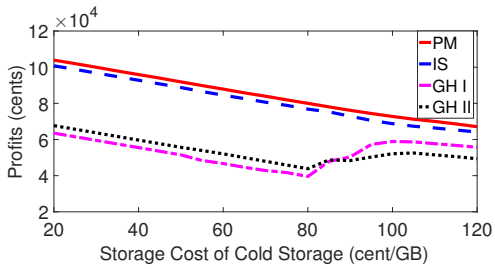
We also plot the impact of the storage cost of hot storage on the ARAR in Fig. 3.6(c). One interesting trend is the access rate obtained from the proposed method first increases, and decreases until close to the one gained from IS, while the others decrease and then become constant with the increase in the cost. This is because we combine the two-stage decision processes, thus, when the cost is moderate, the number of files that are stored decreases without decreasing the number of accepted access bids as shown in Fig. 3.6(d). Hence, the denominator in (cf. (3.32)) decreases, which increases the ARAR. Note that once the hot storage cost is very high, the number of accepted bids also decrease as the latency requirement may not be met because too few files are stored in the expensive hot storage. Hence, the ARAR decreases at very high cost. On the other hand, in the other algorithms, as the cost

of the hot storage increases, very few files are stored in the hot storage; thus, very few files can be accessed (see Fig. 3.6(d)) , which decreases the ARAR (see Fig. 3.6(c)). However, if the cost is too high, no more file can be stored in the hot storage which makes the ARAR constant.

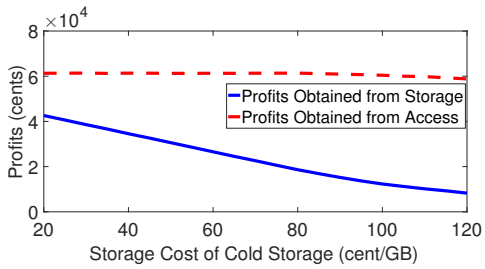
From Fig. 3.6(d), we can see that the number of accepted bids for access stays flat in the very beginning, the reason behind that is these bids are low enough and the associated files are stored in the cold storage, and cannot get accessed. Thus, even we increase the hot storage cost a little, the number of accepted bids for access will not change. Then, it decreases from when the hot storage cost is around 2400 cents/GB, that is because the files, which could be accessed with a lower hot storage cost, cannot get accessed. In the end, when the hot storage cost is high enough, all the files are stored in the cold storage, then the number of accepted bids for access will not change no matter how we increase the hot storage cost.

3.6.5 Impact of Storage Cost of Cold Storage

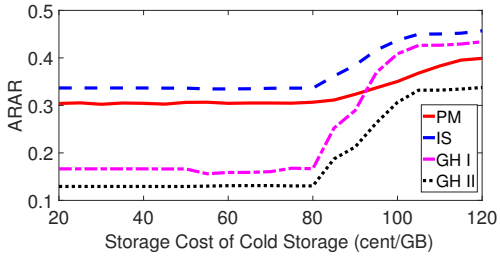
In this subsection, we assume that the storage cost of hot storage is 80 cents per GB, and vary the storage cost of cold storage from 20 cents to 120 cents per GB in steps of 5 cents. Fig. 4.6(a) shows that as the cold storage cost increases the profit attained by our proposed method decreases. As Fig. 3.7(d) shows that when the cost of the cold storage increases the lower number of files are stored. Hence, the profit from the storage decreases (Fig. 3.7(b)). The profit from accepting the access bids remain the same as the number of files stored in the hot storage almost remains the same. Note from Fig. 4.6(a) the profits gained from GH I and GH II decrease first and increase dramatically when the cold storage cost is 80 cents per Gb (i.e., the hot storage cost). The main reason behind this is that beyond this point the hot storage has lower storage cost but higher service rate after that point, which means that the files are prioritized to be stored in the hot storage rather than the cold storage. Thus,



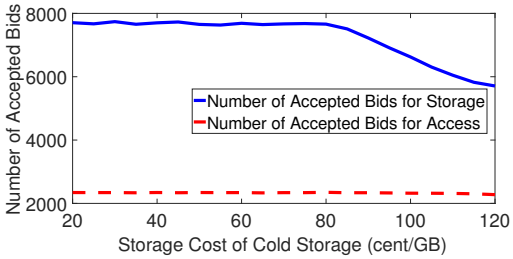
(a) Impact of cold storage cost on profits



(b) Impact of cold storage cost on profits obtained from the file storage and access



(c) Impact of cold storage cost on ARAR



(d) Impact of cold storage cost on the number of accepted bids for storage and access

Fig. 3.7.: Profit and the ARAR as a function of the cold storage cost. $C_1 = 400$GB and $C_2 = 200$GB. The service rates of the cold and hot storages are 100Gb/s and 200Gb/s.

profits from accepting the access bids increase drastically for those greedy heuristics and become close to the optimal.

Note that as the cold storage cost exceeds the hot storage cost fewer number of files are stored. However, the accepted access bids remain the same as depicted in Fig. 3.7(d). That is because whether or not the bid can be accepted for access depends not only on the access bid, but also the first-stage decision. In the second stage, the bids are accepted for access when the latency requirements are satisfied. Since the cold storage has a lower processing speed, thus very few bids are accessed in the second stage from the cold storage. Hence, the denominator of (cf. (3.32)) decreases without decreasing the numerator. Thus, ARAR increases. Note that the IS does better in terms of ARAR. The greedy heuristic GH I also gives a higher

ARAR when the cold storage cost exceeds the hot storage cost. This is because in the first-stage decision the larger files are mostly now stored in the hot storage rather than the cold storage as they pay more. However, the total number of files accepted for cold and hot storage decreases. The larger files are also likely to bid higher in the second stage, thus, the ratio in (cf. (3.32)) increases for GH I as it accepts bids in the descending order of the bids per size. However, GH II accepts bids in a different manner; hence, the ARAR attained by the GH II is strictly lower.

3.7 Summary and Extensions

In order to store the files in the *cold storage* or *hot storage*, this chapter propose a systematic framework for two-stage, latency-dependent bidding, which aims to maximize the cloud storage provider's net profit in tiered cloud storage systems where tenants may have different budgets, access patterns and performance requirements. In the proposed two-stage, latency-aware bidding mechanism, the users can bid for storage and access, in two separate stages, without knowing how the CSP stores the contents. The proposed optimization is modeled as a mixed-integer nonlinear program (MINLP), for which an efficient heuristic is proposed. The numerical results demonstrate that the profits obtained from the proposed method are higher than those of other methods, and the access request acceptance rate (ARAR) also dominates that of other methods as the capacity of the cold storage or the service rate of hot storage increases.

In practice, it is possible that users may collude and modify their bidding behavior for maximizing own objective. The analysis of our two-stage pricing under strategic users and untruthful bids is an interesting future direction. Further, we consider a Poisson arrival process when users come to retrieve their data in the cloud. Analyzing access latency under general service time distribution and its impact of pricing will also be considered in our future work.

Another interesting direction for the future is to extend the model for erasure coding storage system where multiple copies (n) of the files can be stored and a subset of those copies (k) are required to be fetched to get the original file. In that case, the CSP would need to select the n storage systems to store the file and among those k copies are needed to be fetched to get the original file. There are also other competing market places in the real world, the users may have other decision variables, such market selection. We do not provide any competitive analysis, however, we believe that such analysis constitutes an interesting future direction.

Table 3.1.: Bidding Scenario k Generation

	File Size S_i	Bidding Price for Storage P_i	Mean of Arrival Rate λ_i^k	Latency Requirements l_i^k	Bidding Price for Access q_i^k
Ty^I	64	$64 * \mathcal{U}[0.1, 0.3]$	20	$\mathcal{U}[30 + \frac{64}{5*10^6}, 30 + \frac{64}{10^6}]$	$\frac{50*64*\log(\lambda_i^k+1)}{(l_i^k)^2}$
Ty^{II}	128	$128 * \mathcal{U}[0.1, 0.3]$	10	$\mathcal{U}[30 + \frac{128}{5*10^6}, 30 + \frac{128}{10^6}]$	$\frac{50*128*\log(\lambda_i^k+1)}{(l_i^k)^2}$
Ty^{III}	256	$256 * \mathcal{U}[0.1, 0.3]$	8	$\mathcal{U}[30 + \frac{256}{5*10^6}, 30 + \frac{256}{10^6}]$	$\frac{50*256*\log(\lambda_i^k+1)}{(l_i^k)^2}$
Ty^{IV}	512	$512 * \mathcal{U}[0.1, 0.3]$	4	$\mathcal{U}[30 + \frac{512}{5*10^6}, 30 + \frac{512}{10^6}]$	$\frac{50*512*\log(\lambda_i^k+1)}{(l_i^k)^2}$
Ty^V	1024	$1024 * \mathcal{U}[0.1, 0.3]$	2	$\mathcal{U}[30 + \frac{1024}{5*10^6}, 30 + \frac{1024}{10^6}]$	$\frac{50*1024*\log(\lambda_i^k+1)}{(l_i^k)^2}$

4. OPTIMIZED PORTFOLIO CONTRACTS FOR BIDDING THE CLOUD

Amazon EC2 provides two most popular pricing schemes–i) the *costly* on-demand instance where the job is guaranteed to be completed, and ii) the *cheap* spot instance where a job may be interrupted. We consider a user can select a combination of on-demand and spot instances to finish a task. Thus he needs to find the optimal bidding price for the spot-instance, and the portion of the job to be run on the on-demand instance. We formulate the problem as an optimization problem and seek to find the optimal solutions. We consider three bidding strategies: one-time requests with expected guarantee and one-time requests with penalty for incomplete job and violating the deadline, and persistent requests. Even without a penalty on incomplete jobs, the optimization problem turns out to be non-convex. Nevertheless, we show that the portion of the job to be run on the on-demand instance is at most half. If the job has a higher execution time or smaller deadline, the bidding price is higher and vice versa. Additionally, the user never selects the on-demand instance if the execution time is smaller than the deadline.

The numerical results illustrate the sensitivity of the effective portfolio to several of the parameters involved in the model. Our empirical analysis on the Amazon EC2 data shows that our strategies can be employed on the real instances, where the expected total cost of the proposed scheme decreases over 45% compared to the baseline strategy.

4.1 Introduction

Cloud computing is projected to increase to $162 billion in 2020. The latest quarterly results released from Amazon shows that Amazon Web Services (AWS)

realized 43% year-to-year growth, making contribution to 10% of consolidated revenue and 89% of consolidated operating income [100]. However, the success story of the CSPs inherently depends on the user's participation. The cloud service provider's (CSP's) prices affect the users' behavior and the profit of the CSP. CSPs provide different pricing plans to meet customers' service requirements which we describe in the following.

4.1.1 Cloud Pricing Schemes

The most popular pricing schemes broadly adopted are: usage-based pricing, auction-based pricing, and volume-discount pricing [101]. Among the above, the most popular ones are the usage based and the auction-based pricing. In the usage-based pricing, which is also known as pay-as-you-go, asks a fixed price per instance per hour and remains constant (static) over a long time. This type of pricing scheme is commonly implemented in Amazon [102], Google [7], Windows Azure [6], etc. For example, Amazon EC2 on-demand instance provides fixed price short term service with no up-front payment or long-term commitment. The user will certainly get the resource on on-demand instance [102].

On the contrary, in the auction-based pricing (*e.g.*, Amazon EC2 spot pricing), users bid for the service, and the CSP sets a dynamic threshold to decide the successful bids based on the demand and bids. In each time slot, the bids that are above the spot price (which is decided by the CSP) will be accepted, and others will be rejected. The users pay the spot price[1]. Although a user can bid a relatively lower price for the spot instance compared to the price it has to pay for the on-demand instance, the job may be interrupted when the bid is below a threshold [102]. Typical job types like word counting, multimedia processing, etc. can be run using auction-based pricing strategies.

[1]Hence, it has the similarity with the generalized second price auction.

There are two types of spot instance requests: one-time requests and persistent requests. Specifically, the user bids with the instance type, bid price, etc., and the instance will start when the bid is higher than the spot price. When the user's bid price is lower than the spot price, the job will be interrupted and action taken afterwards relies on the request type: the interrupted job will be resumed when the bid price is above the spot price again if it is a persistent request and will be terminated permanently otherwise (i.e., if it is a one-time request). Fig. 4.1 depicts this procedure.

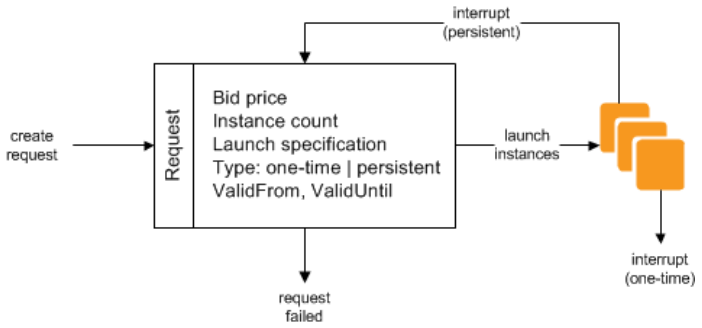


Fig. 4.1.: Spot Instance Requests [53]

4.1.2 Research Challenges and Contributions

The user will be likely to distribute its job over on-demand and spot instances. This is because in the on-demand instance, the user will be able to complete the job. In the spot market, the job may be interrupted. However, the user can pay less. Most of the existing literature considers the profit maximizing spot pricing from the CSP's perspective [46, 103–106]. It is also imperative to investigate the optimal decision of the users. The users needs to select a price to maximize in the spot-instance. The user also needs to select the portion to be run on the on-demand instance. In this chapter, we propose a method that enables the users to decide how to make decisions in order to minimize the expected cost while completing the job within the deadline.

The closest work to ours is [107], which is motivated by Amazon EC2's auction-based spot pricing, they modeled the CSP's spot price setting and derive user's optimal bidding strategies. However, in the cloud infrastructure, more and more cloud jobs are requested for data analysis, such as web logs analysis, weather forecast analysis, finance analysis, scientific simulation, etc. Most of them have hard deadlines, which can be predefined by the companies, or application providers. Failing to do meet the deadline, it may incur a penalty [108]. For example, weather prediction is carried out by exploiting complex mathematical models based on the historical data such as temperature, atmospheric pressure, humidity, etc. The Environmental Modeling Center runs the Global Forecast System model for 16 days into the future [109]. If the computing job misses the deadline, some extreme weather may not be predicted in time, awareness and preparedness for the severe weather will be missed, resulting in large loss of human life, social welfare, and financial resources if the severe weather happens. [107] did not consider the deadline or the penalty incurred when a job misses the deadline. However, the deadline and penalty considered above impacts the bidding prices.

We consider that the users can access both the on-demand instances and the spot instances. On-demand instances exploit the pay-as-you-go pricing scheme which guarantees the availability of the instances and there is no interruption of the job. Unlike on-demand instance, the spot instance uses the auction-based pricing scheme. The user bids for the spot instances, but its job will be interrupted when its bid is below the current spot price [103]. Thus, there is no guarantee that a job can be finished before its deadline if a user selects the spot instance. However, the user is likely to pay less for the spot instance. We consider that a job can be split into independent chunks, which can be processed on different machines in a parallel manner. We have seen many similar workloads in the real world, for example, word counting, multimedia processing, etc. [101, 110, 111].

A user possessing the jobs, which can be run parallelly, may want to know whether a combination of on-demand and spot instances can be used to minimize the total

cost while finishing the job before deadline. The user now has to select the portion of job to be completed via the on-demand instance and the spot instance. Additionally, the user has to select the bidding price for the participation in the spot instance [2]. We propose an economic portfolio model for computing the optimal behaviors when it comes to how to allocate the job with known fixed deadlines to on-demand and spot instances and how much to bid for the spot instance [3].

We consider the job is fault-tolerant and flexible, and can be randomly divided to two portions. Examples include word counting, Hadoop data processing, stateless web services, etc. We suppose the job has a fixed deadline, and a fixed execution time, which is the total time required to complete the job without any interruption. For example, suppose a job requires 30 minutes to finish. If it starts and gets interrupted after 10 minutes, we still need 20 minutes to execute the job. We consider two request mechanisms: one-time requests and persistent requests. Depending on the user's requirements, he can choose one of the above request mechanisms. For example, when a user wants to test some data, if the result from a sub-data set is acceptable, then he can choose to use one-time request. However, if he wants to get the result from the whole data set, he may want to use persistent request, which can guarantee to run for enough consecutive compute time to complete the task. Recall that with a one-time request, if a user's job is interrupted, it will not be resumed on the spot instance. Thus, the user's job may not be finished before the deadline by placing one-time requests. We consider two bidding strategies in one-time requests. The first one considers the user wants to finish its job before the deadline in an expected sense,

[2]Though we consider the cloud computing market, our model can be applied to other markets. For example, in the Display Advertising market of Internet, the spots are allocated in a two-stage process. In the first market, the publisher (e.g., Google's DoubleClick, OpenX, and Yahoo!'s Right Media) promises to deliver a contracted number of impressions within a fixed time slots (over a day); the second market (spot market) runs an auction to allocate the displays in every time frame (in an hour), where the advertisers arrive and bid for the displays [112]. The spot market is operated if an advertiser requires certain spots in the current time frame. Thus, the advertiser has to select how much to bid in the spot market, and how much to buy fixed impressions in the first market.

[3]Our approach can be applied in a MapReduce setting. Suppose we fix the number of instances M to run for each job apriori. We need to split each job into two sub-jobs, and decide whether to run on spot or on-demand instance. Each sub-job will be run on M instances.

and we denote this strategy as *one-time requests with expected guarantee* (OTR-EG) (Section 4.3.1). However, it may not pay a penalty if it is incomplete or misses the deadline. Subsequently, we consider a strategy, where a user pays a penalty if the job is incomplete or misses the deadline, and we denote this strategy as *one-time requests with penalty* (OTR-P) (Section 4.3.2). Finally, we consider the bidding strategy by placing persistent requests, and denote it as *persistent request* (PR) (Section 4.3.3), where the interrupted job can be resumed when the bid price is higher than the spot price again.

Our analysis shows that, in terms of one-time requests, *when the deadline is smaller than the execution time, the user should select the on-demand instances. The optimal bidding price in OTR-EG will decrease first and then increase with the deadline, while the optimal bidding price in OTR-P will increase with the deadline.* We also show the optimal bidding prices on spot instance in OTR-P increase with the increase of the penalty coefficients, and very small or very large penalty coefficient for incomplete jobs will lead to a slower increase of bidding price. However, *when the deadline is larger than the execution time, the user will solely depend on spot instance to finish the job, and the optimal bidding prices for OTR-EG and OTR-P do not change with the increase of the deadline.*

We, subsequently, consider the case where a user places a persistent request for the spot instance. In the persistent request, unlike the one-time request, an interrupted job will be resumed when the bid price exceeds the spot price again. A lower bid can reduce the cost of executing the job on the spot instance, while the number of interruptions may be increased, so dose the total idle time, total recovery time and total completion time, which may exceed the deadline. Thus, it is not apriori clear that how much portion of the job should be run on the spot-instance, and what the bidding price will be if we want to finish the job before the deadline in expectation. *Our result shows that the persistent request reduces the expected cost of the user as compared to the one-time-requests. Similar to the one-time-request, only when the deadline is smaller than the execution time, the user selects the on-demand instances.*

Note that we did not consider any penalty based approach in the persistent request. This is mainly because in the persistent request, the interrupted job is not discarded and thus, it will finish unlike the one-time request.

The main contributions of this chapter can be summarized as follows:

- **User's optimal or local optimal bidding strategies**: For the one-time request and persistent request job, we formulate the cost minimization problem as an optimization problem. The problem turns out to be non-convex. Nevertheless, we find analytical expression for the optimal solutions for the one-time request without penalty and the persistent request. However, for the one-time request with penalty, we provide algorithms for solving the proposed non-convex problem.

- **Analytical Results**: Our analytical result shows that only when the deadline is smaller than the execution time, the user should select the on-demand instances. We show a threshold type behavior for one-time request. When the penalty is above a certain threshold, the user opts for the on-demand instances. However, below the threshold, the portion of the job that is run on the on-demand instance becomes independent of the penalty parameters. Our result shows that the persistent requests reduce the expected cost of the user compared to the one-time-request.

- **Numerical Evaluation**: We, empirically, evaluate the impact of different parameters on the portion of the job should be run on the spot instances, and the bidding price. Our result shows that the expected cost, and the portion of the job that is run on the on-demand instance decreases with the increase in the deadline. The bidding price in the persistent request instance decreases with the increase in the deadline. However, the bidding price in the one-time request increases with the increase in the deadline in the one-time request.

- **Real time Data**: Using the real time data, we show the strength of our approach compared to the baseline strategies readily employed by the users. Specifically, we compute the optimal bidding strategy in the spot-instance, and the optimal portion

of the job should be run on the on-demand instance. Finally, we show that the user's cost is reduced using our approach compared to the baseline ones.

4.1.3 Related Literature

The genre of works can be divided based on the topics they considered.

Portfolio Contract: This type of portfolio contract has been practiced and studied in many other contexts especially in procurement, e.g., Hewlett-Packard (HP) uses a portfolio approach for procurement of electricity or memory products [113]. Motivated by that practice, the procurement has been studied in multi-period [114] and single-period [115] settings. However, the above portfolio contracts did not study the cloud spot market, the deadline, and the execution time.

Deadline-based cloud scheduling: Deadline-based resource allocation has been considered in many cloud research works. While resource allocation approaches are utilized in the cloud context, which aims to meet the jobs' deadlines and utilize the cloud resource more efficiently [116] or minimize the total execution cost [117,118], they only consider from the CSP's perspective. In this chapter, we develop a model to optimize the bidding strategies of the user. Although in [107], optimal one-time request and persistent request bidding strategies are proposed, they do not consider the deadline of the user's job, which may be not practical [116]. In this chapter, we not only consider one-time request without penalty and persistent request bidding strategies, we also include one-time request with penalty model to balance the finished job and penalty for the unfinished job or late completed job. This model can be applied to the type of the job with a soft deadline, which is a deadline when it is unmet, dose not lead to computation useless [119, 120].

Game Theory, Auctions and Bidding: Game Theory has been used to model the interactions between CSPs and users to reach an equilibrium [121–124]. In distributed resource allocation games, auctions have been proposed to be a solution [125, 126], including to ensure truthful bidding in Amazon spot pricing [127].

The remainder of this chapter is organized as follows. Section 4.2 introduces the system model. In Section 4.3, we present three types of bidding strategies: one-time request without penalty (Section III-A) and with penalty (Section III-B), and persistent request (Section III-C). In Section 4.4, extensive simulation results show the benefits of each strategy. We test our proposed model and results using Amazon spot price history in Section 4.5. Finally, Section 4.6 concludes this chapter. We relegate all the proofs in Appendix.

4.2 System Model

We consider a CSP, which can provide two types of computing instances: on-demand instance and spot instances. On-demand instance can guarantee the availability, but the price is fixed and high. In order to provide a reduced-cost service, the CSP also offers spot instance, which may terminate unpredictably since the price fluctuates based on availability and demand, and update spot price in every certain time period, *e.g.*, every 5 minutes. The users can run its job on the spot instance as long as the bid price exceeds the spot price.

We consider a user can select a combination of on-demand and spot instance to finish a task. In other words, the user decides the portion of the job to be run on the on-demand instance and the rest in the spot instance. The spot price is much lower than the on-demand price for every instance type [102]. However, the spot market cannot guarantee that the task is run continuously if her bidding price is not high enough, which means the task may be interrupted and takes extra time to get recovered, so that the task may take longer time to get finished. Therefore, the user should balance the proportion of the job she runs on on-demand instance, with the bidding price in an spot market to run the rest of the job on spot instance. This paper aims to provide a framework to help users to decide how much to run at on-demand instances and how much to bid for spot instances with the objective to minimize the total cost, subject to the constraint that deadline has to be satisfied.

We consider a series of discrete time slots $t \in \{1, 2, \cdots\}$ and denote the spot price at time slot t as $\pi(t)$. We assume the spot prices $\pi(t)$ are i.i.d, upper-bounded by the on-demand price $\bar{\pi}$ for the same instance type and lower-bounded by the marginal cost of the instance $\underline{\pi}$, which is very small and closed to 0 [46], that is, $\underline{\pi} \leq \pi(t) \leq \bar{\pi}$. We use F_π to denote the cumulative distribution function (CDF) of spot price $\pi(t)$, which is heavy-tailed [107], corresponding to the probability density function (PDF) f_π. We use p to denote the user's bid price. $F_\pi(p)$ gives the probability that $p \geq \pi(t)$, that is, the user's bid gets accepted. We assume f_π monotonically decreases, thus $F''_\pi(p) = f'_\pi(p) < 0$, i.e., $F_\pi(p)$ is concave in p, which is consistent with the observations and findings in [107].

Suppose a user has a certain job J, which can be split and run on different machines. First, the user would like to purchase on-demand instances to ensure a certain desired level of finished job in the future; say, q portion of the total amount of task run on on-demand instance. And the rest portion of the job $(1-q)$ will be run on spot instances. Then the user needs to decide how much to bid (p) to the spot market. The strategy of the user is to decide p and q. More formally, we define the strategy of a user in the following

Definition 4.2.1 *The strategy of a user is the vector* $\boldsymbol{x} = (q, p)$.

A user decides (q, p) while minimizing the expected cost. Fig. 4.2 depicts the major considerations that we need to incorporate in the bidding and resource allocation decisions graphically.

Definition 4.2.2 *Latency is the maximum of the completion time of the sub-jobs that are assigned to on-demand and spot instances.*

We assume that the user submits the sub-jobs to on-demand instance and spot instance at the same time. Because the on-demand instance can guarantee availability, it is easy to get the completion time of the sub-job that is assigned to the on-demand instance. On spot instance, the sub-job's completion time may consist of the execution

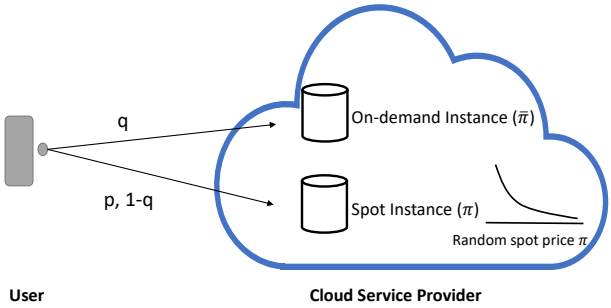


Fig. 4.2.: User Decision Model

time of the sub-job, idle time when the spot price is higher than the bid price, and the time to recover from job interruptions [53]. A lower bid price can reduce the cost of running the job, but may cause job incompletion and more penalty for incomplete or late job (for one-time requests), or extend the job completion time by introducing more interruptions and recovery time (for persistent requests).

We consider three bidding strategies: OTR-EG, OTR-P, and PR, where the first two strategies can be used if the user places a one-time requests, and the third strategy will play a role when the persistent requests are placed. The problem is to design optimal portfolio of contracts and bidding strategies in different settings, so that the expected total cost is minimized, subject to that the expected latency is not larger than deadline. We use $\mathbf{x}^* = (q^*, p^*)$ to denote user's optimal decisions. Then we will investigate the extent of the benefits that can be accrued by managing a portfolio of contracts instead of sticking to on-demand instance contract.

Our notations in this chapter are summarized in Table 4.1. We not only consider the job's characteristics such as its execution time t_e, total completion time T, the recovery time for writing and transferring the data saved after interruption t_r, but also include the deadlines t_s on the job completion times.

Table 4.1.: Key terms and symbols

Symbol	**Definition**
p	User bid price
q	The portion of job that will run on on-demand instance
π	Spot price
$\bar{\pi}$	On-demand price
$\underline{\pi}$	Minimum spot price
t_k	Length of one time slot
T	Total job completion time
t_s	Deadline of the job
t_e	Job execution time (w/o interruptions)
t_r	Recovery time from an interruption
c_I	Penalty coefficient for incomplete job
c_s	Penalty coefficient for late completed job

4.3 User Bidding Strategies

In this section, we first consider OTR-EG, subsequently, OTR-P, and finally, PR for a single instance on each machine type.

4.3.1 OTR-EG

In one-time request, a job on spot instance will not be resumed as soon as the job is interrupted, the user's objective is to minimize the total cost. However, the job has to be completed before the deadline. In order to make sure that there exists at least one feasible solution such that the job can be finished before deadline, we assume that $t_e \leq 2t_s$. The factor 2 comes from the fact that the smallest time a job

can be completed when the half of the job is run on the on-demand instance, and the rest in the spot instance. Thus, for a feasible solution, $0.5t_e \leq t_s$.

First, we compute the expected amount of time that a job will continue running without any interruptions if the user bids p. Note that when a user bids the price p, its bid will only be accepted if the spot price π is lower than p. Thus, the probability that the bid will be accepted at an instance with probability $1 - F_\pi(p)$. Thus, we have the following

Lemma 2 *[107] The expected amount of time that a job will continue running without any interruptions is:*

$$t_u(p) = t_k \sum_{i=1}^{\infty} iF_\pi(p)^{i-1}(1 - F_\pi(p)) = \frac{t_k}{1 - F_\pi(p)} \tag{4.1}$$

In order to guarantee that the job that runs on spot instance can be finished, that is, the expected amount of time that a job will keep running must exceed its execution time, we need the following constraint:

$$(1 - q)t_e \leq \frac{t_k}{1 - F_\pi(p)}. \tag{4.2}$$

Now, we compute the expected time tn for a job to enter the system when the user bids p. Note that a job can only enter the system if the spot price is lower than the bid price. The random variable that the bid gets into the system follows a Geometric distribution. Thus we get the following term:

$$t_n = t_k \sum_{i=1}^{\infty} i(1 - F_\pi(p))^i F_\pi(p) = t_k(\frac{1}{F_\pi(p)} - 1) \tag{4.3}$$

From (cf. (4.3)), we notice that t_n monotonically decreases with p. Thus we would intuitively expect that in order to finish the job $(1 - q)t_e$ before deadline, the user should bid more to shorten the expected amount of time to enter the system t_n.

In order to finish the job that run on spot instance, the deadline need to be longer than the summation of the expected time to enter the system and the required execution time $(1 - q)t_e$, that is,

$$t_n + (1 - q)t_e \leq t_s. \tag{4.4}$$

Lemma 3 *[107] The expected price that a user must pay to use an instance in each time slot on spot instance, or the expected value of all possible spot prices that are no more than p is*

$$\mathbb{E}(\pi|\pi \leq p) = \frac{\int_{\underline{\pi}}^{p} x f_\pi(x) dx}{F_\pi(p)} \tag{4.5}$$

Lemma 4 *$\mathbb{E}(\pi|\pi \leq p)$ monotonically increases with p and not larger than $\frac{\bar{\pi}+\underline{\pi}}{2}$.*

Proof The proof is provided in Appendix A.5. ■

If the user's bid price is p, the user has to pay for the spot instance is the expected spot price $\mathbb{E}(\pi|\pi \leq p)$. The user puts $(1-q)$ fraction of the job on the spot market. Thus the user's expected cost for running the job on the spot instance is $(1-q)t_e\mathbb{E}(\pi|\pi \leq p)$. Recall that on the on-demand instance, a user has to pay the price $\bar{\pi}$. The user's cost for running q fraction of the job on the on-demand instance is $qt_e\bar{\pi}$. Thus the total expected cost of running the job is

$$qt_e\bar{\pi} + (1-q)t_e\mathbb{E}(\pi|\pi \leq p).$$

The user also has to make sure that its job is completed before the deadline. In other words, the expected time the job will take to finish must be smaller than the deadline t_s. The total time a job takes in the on-demand instance is qt_e and in the spot-instance is given by $t_n + (1-q)t_e$. Thus, the total time to complete the job is $\max\{qt_e, t_n + (1-q)t_e\}$. Hence, the user is solving the following problem:

$$\text{(P5)} \quad \text{minimize} \quad \Phi_1(p,q) = qt_e\bar{\pi} + \frac{(1-q)t_e\int_{\underline{\pi}}^{p} x f_\pi(x) dx}{F_\pi(p)} \tag{4.6}$$

$$\text{subject to} \quad (4.2), \quad (4.4)$$

$$qt_e \leq t_s \tag{4.7}$$

$$\underline{\pi} \leq p \leq \bar{\pi} \tag{4.8}$$

$$0 \leq q \leq 1 \tag{4.9}$$

The objective function (cf. 4.6)) aims to minimize the expected total cost running on on-demand and spot instance. In order to guarantee that the job can be completed before deadline, we include constraints (cf. (4.7)) and (cf. (4.4)), which represent that the both of maximum job completion time on each instance including the job execution time and time to enter the system (if any) should not exceed the deadline. The constraint in (cf. (4.8)) denotes the upper and lower bound of the bidding price in the spot instance.

Claim 1 *When* $t_s < t_e \leq 2t_s$, $q^* \leq \frac{1}{2} \leq \frac{t_s}{t_e}$ *and* $F_\pi(p^*) \geq \frac{1}{2}$.

Proof The proof is provided in Appendix A.6. ■

The above theorem shows that q^* is at most half. Thus, at most half of the job is put on the on-demand instance. Intuitively, on-demand price is larger than spot price, if the user wants to minimize his total cost, he will run as much job as possible on spot instance. However, if the deadline is smaller than the execution time, the user may have to opt for on-demand instance as the user has to complete the job before the deadline. The above claim shows that the fraction of the job that will be run on on-demand instance never exceeds half. The results show that the bidding price in the spot market has to be at least the median of the distribution.

Proposition 4.3.1 *When* $\frac{t_e}{2} < t_s < t_e$, *the optimal bid price for a one-time request is*

$$p^* = \max\{\psi_1^{-1}(t_k\bar{\pi}), \psi_2^{-1}(0)\}, \tag{4.10}$$

where $\psi_1^{-1}(.)$ *is the inverse function of*

$$\psi_1(p) = \frac{2t_k \int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)} + 2pt_sF_\pi(p) - pt_s - (t_s + t_k)\int_{\underline{\pi}}^{p} x f_\pi(x)dx \tag{4.11}$$

and $\psi_2^{-1}(.)$ *is the inverse function of*

$$\psi_2(p) = (t_s + t_k)F_\pi(p) - (t_s + t_k)F_\pi(p)^2 - t_k \tag{4.12}$$

with $F_\pi(p) \geq \frac{1}{2}$. *Further the optimal portion of the job to run on on-demand instance is*

$$q^* = 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}. \tag{4.13}$$

Proof The proof is provided in Appendix A.7. ■

Proposition 4.3.1 implies that the portion of job that runs on the on-demand instance q^* decreases as the deadline t_s increases. Intuitively, as the deadline increases, a user can be more likely to run the job in the spot instance as the job can be more likely to be finished using spot instances instead of the on-demand instance resulting into a lower cost. The above proposition also shows that the optimal bidding price p^* takes the maximum value of two functions. The intuition is that with certain portion of job running on spot instance, lower price can decrease the total cost, however, in order to guarantee that the spot instance can continue running without any interruption, the price cannot get too low.

Note that though the optimization problem is non-convex, we still obtain the optimal strategy. q^* is non-zero, however it is less than half.

Proposition 4.3.2 *When* $t_s > t_e$, *the optimal bid price for a one-time request is*

$$p^* = F_\pi^{-1}(1 - \frac{t_k}{t_e}). \tag{4.14}$$

Further, the optimal portion of the job to run on on-demand instance is

$$q^* = 0. \tag{4.15}$$

Proof The proof is provided in Appendix A.8. ■

We can observe from Proposition 4.3.2 that when $t_s > t_e$, all of the job will be run on spot instance, and the optimal bid price p^* does not depend on the deadline t_s, but instead *increases as the number of time slots that are needed to complete the job,* t_e/t_k *increase*. This increase in the bid price with the t_e/t_k is intuitive because more consecutive time slots are required to complete the job, and thus, a higher bid is needed.

4.3.2 OTR-P

In section 4.3.1, we consider that if the job is interrupted, it can not continue. However, there are some possibilities that the job will not get completed before deadline or get interrupted before completion. In this section, we consider the scenario where a user has to incur a penalty when the job is not completed before the deadline. Note that in the one-time request there can be two possible ways the job may not be completed before the deadline: i) *The job is incomplete*, and ii) *The job is late*. We now define each of them.

Definition 4.3.1 *Incomplete Job: the job is interrupted before its completion.*

Definition 4.3.2 *Late Job: the job is completed (i.e. it is never interrupted), however the total time it takes is greater than the deadline. For example, when the time to enter the system is long, the job may get completed beyond the deadline.*

In section 4.3.1, we put a constraint where we consider the expected time for completing the job is less than the deadline. However, as the spot price is random, the job that we run on the spot instance may not be completed (as it is one-time request) or may be completed after the deadline. In this section, we compute the optimal solution where we put penalty for the job which is incomplete or late.

If the job is not completed (i.e., the case (i) holds), there is a penalty c_I associated with the unfinished portion of the job[4]. If the job is not interrupted, however, it is completed after the deadline, there is another penalty c_s for the portion of the job that is completed after the deadline. We also assume $c_s \leq c_I$, which means the completed job will have a lower penalty than that of incomplete one. We denote the total number of slots needed to complete the job in the spot instance is $K(q) = \frac{(1-q)t_e}{t_k}$.

The user wants to minimize the expected cost which also consists of the expected penalty for incomplete jobs. We begin by finding the expected total penalty and then

[4]If the job is not complete, one may need on-demand instances or incur penalty for the unfinished job.

formulate the optimization problem before deriving the user's optimal bid price. We now compute the expressions.

Definition 4.3.3 *Let $L(p,q)$ be the expected time by which a completed job is late,* i.e.*, $L(p,q) = (t_c - t_s)^+$ where t_c is the time to complete the job when the user's strategy is (p,q).*

Lemma 5

$$L(p,q) = t_k(1-F_\pi(p))^{\frac{t_s}{t_k}-K(q)+1}F_\pi(p)^{K(q)-2} \tag{4.16}$$

Proof The proof is provided in Appendix A.9. ■

Recall that $F_\pi(p)$ represents the probability that $p \geq \pi$, that is, the request starts to run or continues running (we denote it as *"success"*); and $1-F_\pi(p)$ is the probability that the request fails or get terminated (we label it as *"failure"*). The intuition behind Lemma 5 is from when the user places the bid, a Bernoulli trial is "conducted" in each time slot. In order to guarantee the portion of job $(1-q)t_e$ can get completed on spot instance, a fixed number $K(q) = \frac{(1-q)t_e}{t_k}$ statistically independent Bernoulli trials' results need to be"success" successively, which happens with probability $F_\pi(p)^{K(q)-1}$. On the other hand, the random variable that the bid gets the first "success" follows a Geometric distribution. For example, when the bid dose not win until the Mth time slots, the probability is $(1-F_\pi(p))^{M-1}F_\pi(p)$. When the number of time slots that the bid spends without getting accepted is larger than $\frac{t_s}{t_k} - K(q)$, the job may be completed but late. Considering all the possibilities of the late but completed job, we have the expression in Lemma 5.

Definition 4.3.4 *Let $EC(p,q)$ be the portion of the job that is completed on the spot instance.*

Lemma 6 *The expected portion of the job that can be completed when the user bids the price p*

$$EC(p,q) = \frac{1-F_\pi(p)^{K(q)}}{1-F_\pi(p)}t_k \tag{4.17}$$

Proof The proof is provided in Appendix A.10. ■

The intuition behind Lemma 6 is in the Bernoulli process, the expected completed job is from the first "success" to the job interruption (the first "failure" from the first "success") or the job completion.

Definition 4.3.5 *Let $EI(p,q)$ be the portion of the job that is incomplete in the spot instance when the user bids the price p.*

$EI(p,q)$ is simply the difference between the total portion of job running on spot instance and the expected portion of job that can be completed with bid price p on spot instance, thus we can obtain

Lemma 7

$$EI(p,q) = (1-q)t_e - EC(p,q) = (1-q)t_e - \frac{1-F_\pi(p)^{K(q)}}{1-F_\pi(p)}t_k \tag{4.18}$$

Considering the penalty for incomplete job and completed but late job, the user solves the following optimization problem:

$$\text{(P6)} \quad \text{minimize} \quad \Phi_2(p,q) = qt_e\bar{\pi} + \frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)} EC(p,q) + c_I EI(p,q) + c_s L(p,q) \tag{4.19}$$

$$\text{subject to} \quad (4.7), \quad (4.8), \quad (4.9)$$

$$(1-q)t_e \leq t_s \tag{4.20}$$

Solution Method: Note that if c_I, c_s are large, q and the price should increase in order to avoid hefty penalty. Although the constraints in (P6) are linear, the objective function is non-convex. Thus problem (P6) is non-convex. Unlike the problem in the OTR-EG, we cannot have any closed form for (P6). We use the successive convex approximation based algorithm [128], which iteratively solves approximate convex relaxation of the problem. The algorithm is stated in Algorithm 1. Let $\tilde{U}$ as the

approximation of the objective function U, which is the first order approximation of U, that is,

$$\tilde{U}(\mathbf{x};\mathbf{y}) = \sum_{i=1}^{2}(\nabla_{x_i}U(\mathbf{y})^T(x_i - y(i)) + \frac{\tau}{2}(x_i - y_i)^2). \tag{4.21}$$

Instead of solving U, we solve $\tilde{U}$ iteratively, which is shown in Algorithm 1. When the difference between two successive objective values is smaller than $\epsilon = 10^{-5}$, the iteration stops.

Algorithm 1 Successive Convex Approximation Algorithm to solve (P6)

Input: $\nu = 0$, $k = 0$, $\gamma \in (0, 1]$, $\epsilon > 0$, $\mathbf{x}^0 = (q^0, p^0)$ such that $\mathbf{x}^0$ is the solution of OTR-EG.

Output: $\hat{\mathbf{x}}(\mathbf{x}^\nu)$

1: **while** obj(k) - obj(k-1) $\geq \epsilon$ **do**
2: // solve for $\mathbf{x}^{\nu+1}$ with given $\mathbf{x}^\nu$.
3: **Step 1:** Compute $\hat{\mathbf{x}}(\mathbf{x}^\nu)$, the solution of $\hat{\mathbf{x}}(\mathbf{x}^\nu) = \text{argmin}\tilde{U}(\mathbf{x}, \mathbf{x}^\nu)$, subject to (4.7), (4.8), (4.9), and (4.20), solved using CVX.
4: **Step 2:** $\mathbf{x}^{\nu+1} = \mathbf{x}^\nu + \gamma^\nu(\hat{\mathbf{x}}(\mathbf{x}^\nu) - \hat{\mathbf{x}}^\nu)$.
5: //update index
6: **Step 3:** $\nu \leftarrow \nu + 1$.
7: **end while**

4.3.3 PR

In section 4.3.1 and 4.3.2, we consider the one-time request job. We now consider a job that places a persistent spot instance request, where the job can be interrupted and recovered upon resuming when the bid price is above the spot price.

We, first, compute the total time T for completing a job in the PR in spot instance. The expected running time is $TF_\pi(p)$ with bidding price p, and the associated expected idle time is $(1 - F_\pi(p))T$. The expected number of idle-to-running

transitions in T/t_k time slots is $\frac{T}{t_k}F_\pi(p)(1-F_\pi(p))$. We incur a recovery time every time there is a transition from the ideal state to the running state. Thus, $TF_\pi(p) = (\frac{T}{t_k}F_\pi(p)(1-F_\pi(p)))t_r + (1-q)t_e$, we get $TF_\pi(p) = \frac{(1-q)t_e}{1-\frac{t_r}{t_k}(1-F_\pi(p))}$ [107].

Lemma 8 *The total time including the recovery, execution and idle time is*

$$T = \frac{(1-q)t_e}{1-\frac{t_r}{t_k}(1-F_\pi(p))}\frac{1}{F_\pi(p)} \tag{4.22}$$

Note that as $F_\pi(p)$ increases the time decreases.

$$\text{(P7)} \quad \text{minimize} \quad \Phi_3(p,q) = qt_e\bar{\pi} + \frac{(1-q)t_e}{1-\frac{t_r}{t_k}(1-F_\pi(p))}\frac{\int_{\underline{\pi}}^{p} xf_\pi(x)dx}{F_\pi(p)} \tag{4.23}$$

$$\text{subject to} \quad (4.7), \quad (4.8), \quad (4.9)$$

$$\frac{(1-q)t_e}{1-\frac{t_r}{t_k}(1-F_\pi(p))}\frac{1}{F_\pi(p)} \leq t_s \tag{4.24}$$

$$t_r < \frac{t_k}{2(1-F_\pi(p))} \tag{4.25}$$

The expected total time including the recovery, execution and idle time should be smaller than the deadline, so we get constraint ((cf. 4.24)). The constraint in (cf. (4.25)) guarantees that the recovery time is sufficiently small such that the job's running time is finite [107].

Claim 2 *In PR, when* $\frac{t_e}{2} < t_s < t_e$, $F_\pi(p^*) \geq \frac{1}{2}$.

Proof The proof is provided in Appendix A.11. ■

The above claim shows that in the spot instance a user's bid should exceed the median value of the distribution.

Proposition 4.3.3 *When* $\frac{t_e}{2} < t_s \leq t_e$, *the optimal bid price for a PR is*

$$p^* = \bar{\pi} \tag{4.26}$$

Further, the optimal portion of the job to run on on-demand instance is

$$q^* = 1 - \frac{t_s}{t_e} \tag{4.27}$$

Proof The proof is provided in Appendix A.12. ■

The above proposition entails that when $\frac{t_e}{2} < t_s \leq t_e$, the user bids the highest possible value in the spot instance. The user also runs a portion of the job in the on-demand instance. The portion of the job that is run on the on-demand instance is given by

$$q^* = 1 - \frac{t_s}{t_e}.$$

If t_e is large, q^* is higher. Proposition 4.3.3 thus implies that surprisingly, *the bidding price does not change as the deadline changes*, and the optimal portion of the job that run on on-demand instance is decreasing with the deadline. The bidding price at the spot instance is the maximum possible bidding price. This is intuitive, because when t_e is large, the bid price has to be large.

Proposition 4.3.4 *When $t_s > t_e$, the optimal bid price for a PR is*

$$p^* = \psi_3^{-1}(\frac{t_e}{t_s}). \tag{4.28}$$

where $\psi_3^{-1}(.)$ is the inverse function of

$$\psi_3(p) = F_\pi(p)[1 - \frac{t_r}{t_k}(1 - F_\pi(p))]$$

The optimal portion of the job to run on on-demand instance is

$$q^* = 0 \tag{4.29}$$

Proof The proof is provided in Appendix A.13. ■

This shows that similar to the one-time request, the portion of the job that is run on the on-demand instance is 0 when $t_s > t_e$. Thus, a job will be run on the on-demand only when $t_s \leq t_e$. Also note that $\psi_3(\cdot)$ is an increasing function. Hence, as the ratio $\frac{t_e}{t_s}$ increases the bidding price also increases.

Lemma 9 *When $t_s < t_e \leq 2t_s$, the difference of optimal portions of job to run on on-demand instance with OTR-EG and PR is bounded by $\frac{t_k}{t_e}$.*

Proof The proof is provided in Appendix A.14. ■

4.4 Numerical Studies

In this section, we present computational results that illustrate the sensitivity of the expected total cost, the corresponding bid price and the portion of job that run on on-demand instance in terms of the different parameters used in the model. We specifically focus on the impact of the deadline, penalty coefficient, and the recovery time. Note that we use closed-form solutions for our OTR-EG and PR, and convex approximation algorithm for the OTR-P.

4.4.1 Distribution of Spot Price

We, first, introduce the spot price probability density function. In many applications including the cloud spot market, the prices follow a Pareto distribution [107]. The PDF of the spot price is chosen to be

$$f_\pi(\pi) = \frac{\alpha(\frac{1}{\bar{\pi}-\underline{\pi}})^\alpha}{\theta(\frac{1}{\bar{\pi}-\pi})^{\alpha+1}} \tag{4.30}$$

which behaves like a Pareto Distribution, the random variable π is bounded by $\underline{\pi}$ and $\bar{\pi}$.

4.4.2 Simulation Set Up

In this section, we consider a job that needs one hour (i.e., $t_e = 1h = 3600s$) , the deadline t_s is $2000s$, the length of one time slot is 5 minutes (i.e., $t_k = 5min = 300s$), and the recovery time $t_r = 10s$. We assume that the PDF of the price in spot instances is drawn from the distribution shown in ((cf. 4.30)). We set $\alpha = 3$ and $\theta = 0.983$. The on-demand price $\bar{\pi}$ is 0.35 and the provider's marginal cost of running a spot instance $\underline{\pi}$ is 0.0321.

We use the above parameters to do the numerical evaluations and illustrate the tradeoff of different bidding strategies by comparing the bidding prices, portion of job that runs on on-demand instance, expected total cost and percentage of late job.

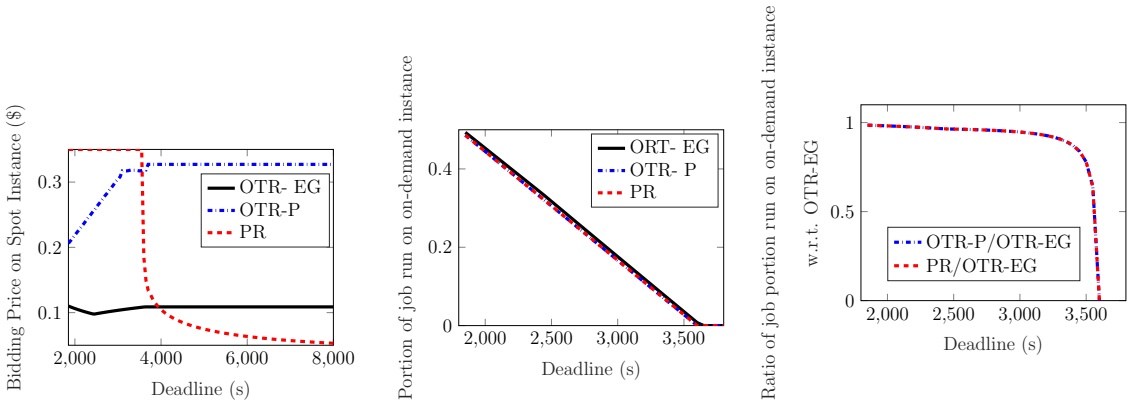
Recall that $x^* = (q^*, p^*)$ is the user's optimal decisions. We use closed-form solutions for our OTR-EG and PR, and convex approximation algorithm for the OTR-P for incomplete job and violating the deadline. we generate the random spot price π_t for each time slot according to the distribution of spot price we introduced, a one-hour count-down program will be run for each bidding strategy based on the associated optimal solution $x^* = (q^*, p^*)$, that is, q^* portion of the job will be run on on-demand instance, and the rest will be run on spot instance with bidding price p^*. Note that after the job starts running, if the random spot price π_t is higher than the bid price p^*, the job with one-time request will get interrupted and not resumed. However, the job with PR will get resumed, where a recovery time will be added, when the random spot price π_t is lower than its bid price p^* again.

When we consider the expected total cost, to be consistent and get insight, besides running cost, penalty for incomplete job and violating the deadline will be added for the strategies OTR-EG and OTR-P; penalty for violating the deadline will be added for PR. We run the simulation for 1000 times, and an average is taken to get the expected total cost for each bidding strategy.

4.4.3 Comparison Among Different Request Mechanisms

In this subsection, we compare the cost, bidding price in different scenarios for a fixed $t_e = 3600$s. We vary the deadline t_s, from 1850s to 8000s in the steps of 50s.

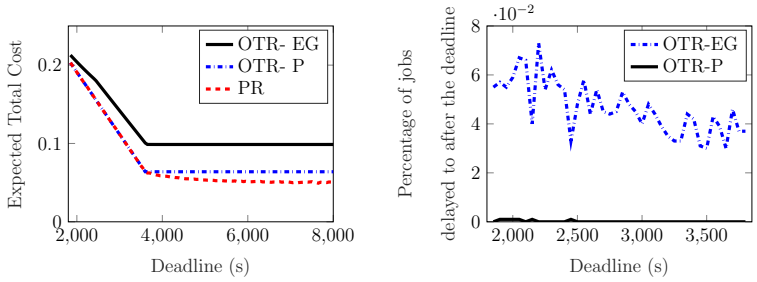
From Fig. 4.3(a), we can see that as the deadline increases, the bidding price in OTR-EG decreases first, and then increases. From Proposition 4.3.1 (cf. (4.10)) we can see that the optimal bid price is determined by the minimum of two terms: the first term (cf. (4.11)) is to get the trade-off between the bid price and portion of job running on on-demand instance, and the second term is to guarantee that the job can continue running without interruption, and finish before the deadline (cf. (4.12)). Thus, the bid price decreases for a while since lower price can lead to lower total cost while the job will not get any interruption. After that we see the bid price increases



(a) Impact of the deadline on bidding price

(b) Impact of the deadline on portion of job run on on-demand instance

(c) Impact of the deadline on ratio of job portion run on on-demand instance compared to OTR-EG



(d) Impact of the deadline on expected total cost

(e) Impact of the deadline on percentage of late job

Fig. 4.3.: Impact of Deadline with $t_e = 3600s$, $c_I = \bar{\pi}/3$, $c_s = \bar{\pi}/10$, and $t_r = 10s$

with the deadline. The reason is that with a longer deadline, the portion of job that runs on spot instance becomes bigger (see Fig. 4.3(b)). In order to guarantee that the job running on spot instance can continue running without any interruption, the bid price should be higher.

Fig. 4.3(a) also shows that when the deadline t_s is smaller than the execution time t_e, the user needs to bid with the upperbound of the spot price $\bar{\pi}$ for PR, and bid lowest for OTR-EG compared to PR and OTR-P. Note that with PR, if a job is interrupted, there will be a recovery time t_r before it get resumed. The user not only needs to pay for it, he also needs to allocate more t_r amount of job to on-demand

instance in order to finish the job. That is to say, although we do not put any penalty on our PR model, $(\pi+\bar{\pi})t_r$ will be added to the total cost for each interruption, which is even larger than the cost to run t_r on on-demand instance, i.e., $\bar{\pi}t_r$. Thus the users needs to bid the upper-bound of the spot price. In terms of OTR-P, the user needs to find a trade-off between the penalty and bid price, thus the bid price is higher than that of OTR-EG but lower than that of PR. When the deadline t_s is larger than the execution time t_e, the user's optimal strategy is to rely on spot instance: there will be no job running on the on-demand instance (see Fig. 4.3(b)) and the optimal bid price will not be impacted by the increase of deadline (cf. (4.14)).

Another interesting observation from Fig. 4.3(a) is that the bidding price is higher in PR for smaller deadline. It decreases and gets lower than that of OTR-EG and OTR-P when the deadline is longer than the execution time. This is because unlike one-time request, in PR, the interrupted job can get resumed in the spot instances and a recovery time will be included, which will induce more cost when the execution time is shorter than the deadline. However, when the execution time is longer than the deadline, with the increase of deadline, smaller bidding price can save the cost while guaranteeing the job can be completed before the deadline.

Fig. 4.3(b) shows that, as we would expect from Lemma 9, when the deadline is smaller than the execution time, the differences among the portions of job that runs on on-demand instance by using OTR-EG and PR are minimal, which is not larger than $\frac{t_k}{t_e} = \frac{300}{3600} \approx 0.0833$. We can also see that the user will put the same portion of the job $\frac{t_s}{t_e}$ running on the spot instance by using OTR-P and PR. The intuition is that because spot instance price is not higher than that of on-demand instance, the user tends to run as much job as possible on the spot instance, which is $\frac{t_s}{t_e}$.

In order to show the difference more clearly, we plot Fig. 4.3(c) to show the ratio of portion of job runs on on-demand instance compared to OTR-EG. The comparison shows that when the execution time is smaller than the deadline, the user put less job on on-demand instance by using OTR-P and PR compared to OTR-EG, and the

portion of the job put on on-demand instance decreases much faster for OTR-P and PR compared to OTR-EG.

Fig. 4.3(d) shows that when the deadline t_s is shorter than the execution time t_e, the expected total costs obtained with different strategies are decreasing. This is intuitive, as the deadline increases, the portions of the job will be run on on-demand instance, whose price is not less than that of spot instance, are getting smaller (which can be verified in Fig. 4.3(b)), thus the expected total costs are becoming lower.

However, when the deadline t_s is longer than the execution time t_e, the user's optimal strategy is to rely only on spot instance to finish the job and there will be no job running on on-demand instance (see Fig. 4.3(b), Proposition 4.3.2 and Proposition 4.3.4). Note that the bidding price obtained from PR decreases with the deadline and gets smaller than the bid prices in OTR-EG and OTR-P, which are not impacted by the increase of the deadline (see Fig. 4.3(a)). Therefore, the expected total costs obtained from PR is decreasing while that of OTR-EG and OTR-P do not change. Another interesting observation is that the expected total cost of PR is lower than that of the other two bidding strategies, indicating that users can further lower the total running cost by using PR.

In Fig. 4.3(e), we plot the percentage of jobs that are delayed to after the deadline for OTR-EG and OTR-P. We can see that with OTR-EG, the percentage of late jobs are decreasing with the increase of deadline and there are always some job being delayed after deadline. However, for OTR-P, almost all the jobs can be finished before deadline. Thus, adding penalty can reduce the fraction of jobs which are delayed.

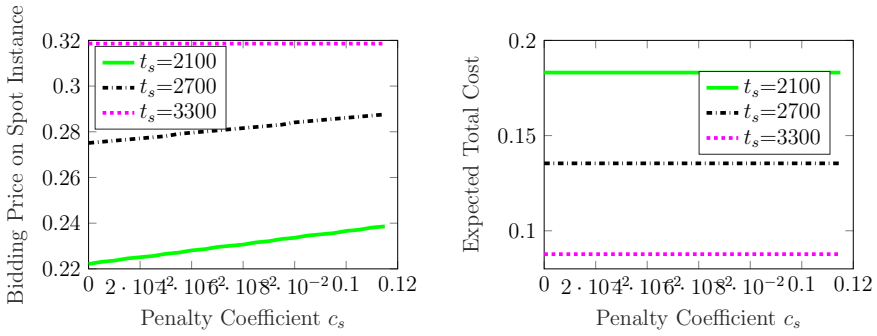
4.4.4 OTR-P: Impact of the Penalty Parameters c_s and c_I

Recall that the penalty corresponding to the incomplete job in the spot instance is c_I and the penalty corresponding to the portion of the job completed after the deadline is c_s. We, now, evaluate the impact of c_I and c_s on the bidding prices and the expected total cost (including penalty).

We assume that the execution time is 3600s. We consider three different scenarios of the deadlines: i) 2100s, ii) 2700s and iii) 3300s.

Impact of Penalty Parameter c_s

In order to see the impact of penalty coefficient for late but completed job c_s on the bidding price and expected total cost, we fix the execution time without any interruptions t_e as 3600s and penalty coefficient for incomplete job c_I as $\bar{\pi}/3$, and change c_s from 0 to $c_I = \bar{\pi}/3$ in the step of 0.005.



(a) Impact of the Penalty Coefficient c_s on Bidding Price

(b) Impact of the Penalty Coefficient c_s on Expected Total Cost

Fig. 4.4.: Impact of Penalty Coefficient c_s with $t_e = 3600s$ and $c_I = \bar{\pi}/3$

We observe for Fig. 4.4 that both the bid prices and expected total cost increase with the increase in the penalty coefficient c_s. The higher bidding price is due to the fact that there is a penalty due to the completed but late job. However, compared to the bid price, the expected total cost is relatively less sensitive to the change of penalty coefficient c_s (see Fig. 4.4(b)). Finally, from Fig. 4.4(b) we also observe that the total cost decreases with the deadline. This is because a smaller portion of the job that runs on the on-demand instance when the deadline is higher.

Impact of Penalty Parameter c_I

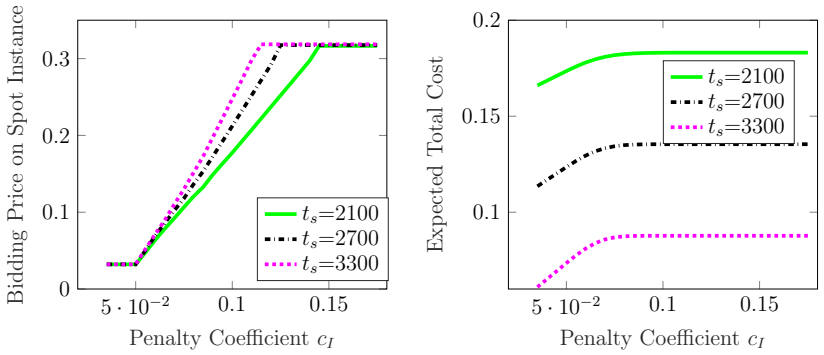
In order to investigate the impact of penalty coefficient for unfinished job c_I on the bidding price and expected total cost, we fix the execution time without any interruptions t_e as 3600s and penalty coefficient for late completed job c_s as $\bar{\pi}/10$, and vary c_I from $c_s = \bar{\pi}/10$ to $\bar{\pi}/2$ in the step of 0.005.

We note from Fig. 4.5(a) that the optimal bidding prices on spot instance increase with the increase in the penalty coefficient c_I. That is intuitive, since a larger penalty coefficient for unfinished job implies that a higher penalty for the unfinished job. Hence, the user has to bid a higher price for spot instance. However, very small or very large c_I will lead to a slower increase of bidding price. When c_I is small and below a certain threshold T_1, the bidding price does not increase much because of the lower penalty. On the other hand, when c_I exceeds a threshold T_2, the bid price becomes closer to the upper bound. Thus, increase in c_I does not increase the bid price as rapidly as c_I in between of T_1 and T_2. Fig. 4.5(a) also suggests that with longer deadline, the bid price is higher. This is because more portion of job will be run on spot instance with longer deadline, then in order to avoid the penalty for incomplete job or completed but late job, higher price needs to be set.

We notice that, in Fig. 4.5(b), the expected total cost is increasing with the penalty coefficient c_I for different deadlines. The total cost decreases with the deadline. Intuitively, longer deadline will introduce less portion of job running on the on-demand instance, the price of which is much higher than the spot price.

4.4.5 PR

In this subsection, we numerically evaluate the bidding prices, and the expected total cost for PR. We again set t_e at 3600s. We investigate the variation of the deadlines and the recovery time on the bidding prices in the spot instances, and the expected total cost. Fig. 4.6 suggests that the bid price on spot instance (Fig. 4.6(a)) and expected total cost (Fig. 4.6(b)) increases with the recovery time. Intuitively,

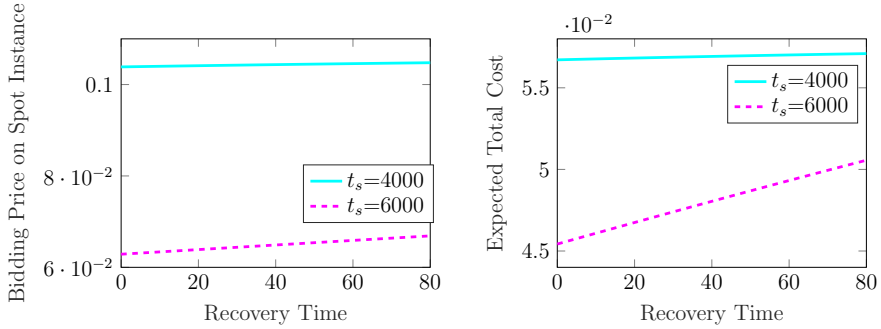


(a) Impact of the Penalty Coefficient c_I on Bidding Price

(b) Impact of the Penalty Coefficient c_I on Expected Total Cost

Fig. 4.5.: Impact of Penalty Coefficient c_I with $t_e = 3600s$ and $c_s = \bar{\pi}/10$

as the recovery time increases the more time is required to recover a job after it is interrupted. Thus, the bidding price is higher. Hence, the expected cost is also higher. Fig. 4.6(a) and Fig. 4.6(b) show that as the deadline increases, the bidding price and the expected cost decreases in the PR scenario. This is consistent with Proposition 4.3.3 which shows that bid price decreases as the deadline increases.



(a) Impact of the Recovery Time on Bidding Price

(b) Impact of the Recovery Time on Expected Total Cost

Fig. 4.6.: Impact of Recovery Time with $t_e = 3600s$.

4.5 Data-Driven Evaluation

To verify our models and comparing with baseline algorithms, we collect three sets of data for three instance types: r3.large, r4.16xlarge and d2.2xlarge in the US Eastern region, whose on-demand prices are \$0.1660, \$4.2560 and \$1.38 respectively [129]. We collect the Amazon EC2 spot price history for the three months from (July 9 - October 9, 2017). The empirical PDF and associated estimated PDF of these prices are shown by the black dots and blue line respectively in Fig. 4.7. We observe that they approximately follow exponential functions, which are consistent among different instance types, though the spot prices are different.

We consider a job that needs one hour (i.e., $t_e = 3600s$) to be executed without interruption. The one hour time periods are Oct.10, 3:00pm-4:00pm, Oct.12, 9:40am-10:40am, and Oct.14, 1:30pm-2:30pm for r3.large, r4.16xlarge, and d2.2xlarge, respectively. We first examine the optimal bid prices using three different bidding strategies (OTR-EG and with OTR-P ($t_s = \bar{\pi}/10$ and $t_I = \bar{\pi}/3$), and PR with recovery time 10s and 50s) that are derived in Section 4.3 on Amazon EC2 spot instances. *The strategies are summarized in Table 4.2 for* $t_s = 2000s$. We consider a model where the user is price taker, one single user's action does not impact the distribution of the spot price.

We show that our approach outperforms two baseline algorithms, where we use one single type of instance to finish the job. Specifically, for smaller value of deadlines, we split the job into two sub-jobs of equal size, and each corresponding to one instance request. For Baseline I, each of them will be run on one on-demand instance. For Baseline II, we consider they only use the spot market. We adopt the strategy that has been proposed in [107]. Since these two sub-jobs are requesting for the same types of spot instance, the bidding prices are same for both of them. For larger value of the deadlines, we run the whole job solely on one on-demand instance and one spot instance for Baseline I and Baseline II respectively.

Smaller Value of the Deadlines

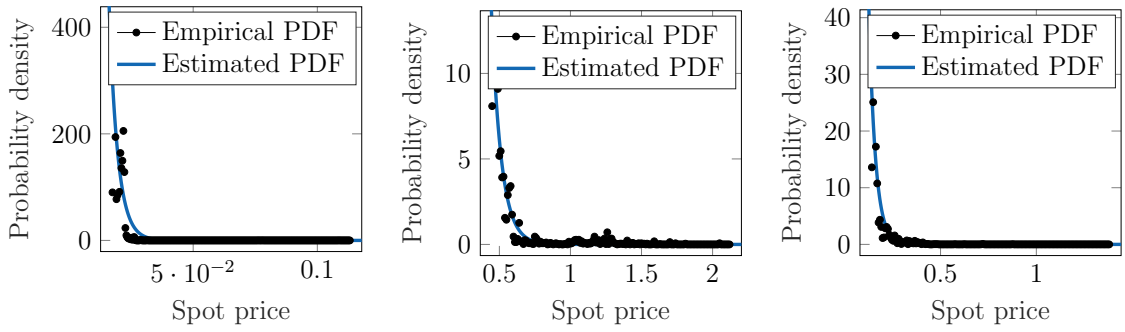
For shorter deadline with $t_s = 2000s$, we determine the optimal bid prices and optimal portion of job running on on-demand instance for different bidding strategies (Table 4.2) to the associated spot instance (r3.large, r4.16xlarge and d2.2xlarge). We set the deadline at $t_s = 2000s$ and compare our algorithm with two baseline algorithms. We use our proposed methods to calculate p and q. Specifically, from Table 4.2, we observe that the price of PR is the highest, the price for OTR-P is medium, and the price of OTR-EG is lowest, which are consistent among different instance types and consistent with our simulation results in Section 4.4.

Fig. 4.8 compares the job completion time, the completed job and total cost for different instance types with different bidding strategies and the baseline algorithms when the deadline t_s is $2000s$.

Fig. 4.8(a) shows that the total cost running with OTR-EG is almost equal to that of the bidding strategy OTR-P and with PR. Thus, the penalty does not increase the cost, yet, increases the portion of the completed jobs. Our numerical results show that *the penalty mechanism reduces the cost by almost* 50% *for all instances compared to the baseline I algorithm. And compared to other methods, the method where we put penalty for the incomplete job and violating the deadline can achieve the minimum cost but is able to get the job done before the deadline (see Fig. 4.8(b) and Fig. 4.8(c)).*

Fig. 4.8(b) shows that r3.large job is interrupted when using OTR-EG and baseline II. In contrast, *the r3.large job with penalty and PR bidding strategies are not interrupted.* However, for r4.16xlarge and d2.2xlarge instances, none of experiments are interrupted. Thus, *the penalty does not affect the rate of completion of jobs.* Since in the baseline I algorithm, all the jobs are put in the on-demand instance, thus, the job is never interrupted. *Note that our penalty based approach is able to achieve similar completion rate of the baseline algorithm I, however, at a smaller cost.which can be verified in Fig. 4.8(a).*

From Fig. 4.8(b), we observe that none of the job is interrupted by using different methods for job r4.16xlarge. We compare the job completion time of r4.16xlarge in Fig. 4.8(c). The results show *the job completion time by using our algorithms is longer than that of baseline I but not beyond the deadline (the red line), while the job completion time is beyond the deadline by using baseline II.* Recall for baseline I, we split the job to two sub-jobs with the same sizes and each of them will be run on on-demand instance without any interruption, which means the job completion time is 0.5 hour. There are two reasons that the job completion times are longer by using our proposed algorithms for job r4.16xlarge compared to baseline I: i) more than 50% of the job is allocated to spot instance, and ii) there may be some time to enter the system.



(a) r3.large (b) r4.16xlarge (c) d2.2xlarge

Fig. 4.7.: Fitting the probability density function of Amazon spot price in the US Eastern region, the best fits are exponential functions with equation $a\exp(bx)$. The fitted parameter values, which are with 95% confidence bounds, are (a, b) = (44350, -285.7), (8126, -14.39), and (1571, -28.84) for Fig. 4.7(a), Fig. 4.7(b), and Fig. 4.7(c), respectively.

Larger Value of the Deadlines

We now discuss our results when $t_s = 4000s$. When $t_s > 3600s$, from Section 4.3 (Proposition 4.3.2 and Proposition 4.3.4), we know that the whole one-hour job will be run on spot instance. The optimal bid prices are shown in Table 4.3.

Unlike the bidding prices with smaller value of the deadlines in Table 4.2, Table 4.3 shows that the bidding prices with PR are the lowest, while the bidding prices for OTR-P are the highest in different instance types. As we also expect from Section 4.3's analysis for PR, longer recovery times (50s) gives higher bidding prices than that of shorter recovery times (10s), which are also consistent with the findings in [107].

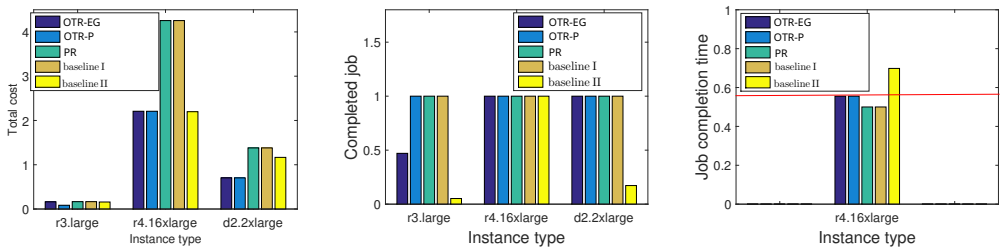
From Fig. 4.9(a) we can see that compared to the other two bidding strategies, OTR-P does not introduce more cost but can finish all the job in time without any interruption or penalty (see Fig. 4.9(b) and Fig. 4.9(c)), verifying the reliability of that bidding strategy.

Fig. 4.9(b) shows that with OTR-EG bidding strategy, it cannot finish more job compared to PR (e.g., r3.large), that is because the job may be interrupted after running some time, while the same job will be recovered from interruption with PR bidding strategy even with lower bidding prices (see Table 4.3). On the other hand, there may be a delay in finishing job in the PR. The reason behind that is the job may not be interrupted by using OTR-EG because of the higher bidding price, while PR job may be interrupted, which induces that the finished job can be more by using OTR-EG compared to PR strategy within a specific time period. Note that our proposed strategy outperforms the baseline II, as the baseline II strategy gives rise incomplete jobs for r3.large. Our proposed strategy for PR as well as for OTR-EG always results in completed jobs.

Note that the job completion time is the summation of the time to enter the system and running time. In Fig. 4.9(c), we can observe that for the jobs running on r3.large and d2.2xlarge, the completion time OTR-EG strategies is not longer than that of OTR-P strategy, that is because the bid price of OTR-EG is lower than that

of OTR-P (see Table 4.3), which means the job OTR-EG has higher probability to get interrupted, thus the running time is shorter. However, since the time to enter the system is longer with a lower bidding price, the job running on r4.16xlarge with OTR-EG bidding strategy has less job completion time compared to that of OTR-P strategy.

Another interesting result in Fig. 4.9(c), unlike our expectation (higher recovery time may induce higher completion time and higher total cost), is that within specific deadline, in insance d2.2xlarge, the PR with longer recovery time completes more job compared to that with shorter recovery time, that is because the longer recovery time yields higher bidding price (see Table 4.3), contributing to less interruption and less time to complete the job.



(a) Total cost for different instance types

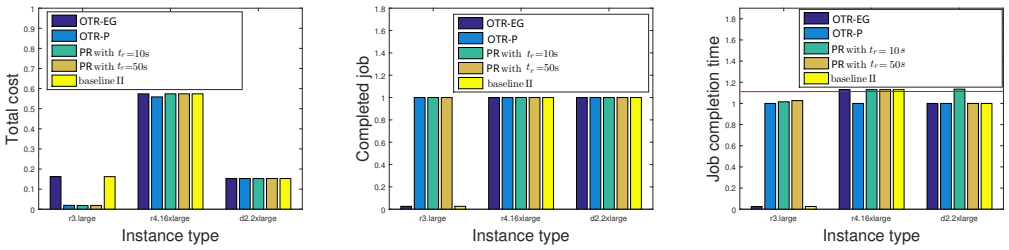
(b) Completed job for different instance types

(c) Job completion time for different instance types

Fig. 4.8.: Comparison among different bidding strategies for smaller value of deadlines

4.6 Summary and Extensions

In this paper, we develop optimization models to minimize the cost by randomizing between the on-demand, and spot instance. We also provide a mechanism to bid optimally in the spot instance. We consider three different strategies: one-time requests with expected guarantee, one-time requests with penalty, and persistent requests. We characterize the optimal portion of the job that should be run in the on-demand instance. Our analytical result shows that a user should never opt for



(a) Total cost for different instance types

(b) Completed job for different instance types

(c) Job completion time for different instance types

Fig. 4.9.: Comparison among different bidding strategies for smaller value of deadlines

on-demand instance if the execution time is smaller than the deadline. However, when the the execution time is larger than the deadline, the user should select the on-demand instance with a non-zero portion of the job. Additionally, the bid price also increases as the execution time increases. However, the portion to be run on the on-demand instance never exceeds half for one-time request without penalty and the persistent request scenario. Our numerical experiments shows the trade-off between higher prices to avoid interruptions (for one-time request without penalty), higher prices to avoid penalty (for one-time request with penalty) and lower prices to save money (for persistent request) on the condition that the deadline requirement is satisfied. It also shows that the user's cost is the lowest in the persistent request scenario. Finally, we use real world data to test our model and analytical results to verify our models and shows that the user can significantly reduces its cost both in the one-time and the persistent request scenario.

Some cloud service providers may give a two-minute warning to the user before the instance is revoked [130]. The user can use re-bidding strategy to avoid termination of the job. We do not consider such scenario in this paper, however, we believe that such analysis constitutes an interesting future direction. In order to prevent the job from revocation, some user may choose to run each spot request on multiple machines.

However, it is not clear how many machines the user should choose and how much to bid. Thus, it can be another interesting research direction.

Table 4.2.: Optimal bid prices for a single instance with $t_s = 2000s$

Instance Types	$\bar{\pi}$	$\underline{\pi}$	OTR-EG		OTR-P		PR with $t_r = 10s$		Baseline I		Baseline II	
			p	q	p	q	p	q	p	q	p	q
r3.large	\$0.1660	\$0.0173	\$0.04258	0.444505	\$0.08813	0.444444	\$0.166	0.444444	\$0.1660	0.5	\$0.02357	0.5
r4.16xlarge	\$4.2560	\$0.4343	\$1.0666	0.4445	\$1.8375	0.444444	\$4.2560	0.444444	\$4.2560	0.5	\$0.5588	0.5
d2.2xlarge	\$1.38	\$0.138	\$0.3538	0.44461	\$0.83833	0.444444	\$1.38	0.444444	\$1.38	0.5	\$0.2	0.5

Table 4.3.: Optimal bid prices for a single instance with $t_s = 4000s$

Instance Types	$\bar{\pi}$	$\underline{\pi}$	OTR-EG	OTR-P	PR with $t_r = 10s$	PR with $t_r = 50s$	Baseline II
r3.large	\$0.1660	\$0.0173	\$0.026	\$0.090299	\$0.025463	\$0.025855	\$0.026
r4.16xlarge	\$4.2560	\$0.4343	\$0.606983	\$1.8808	\$0.596374	\$ 0.604158	\$0.606983
d2.2xlarge	\$1.38	\$0.138	\$0.3538	\$0.83833	\$0.218868	\$0.222752	\$0.3538

5. CONCLUSIONS AND FUTURE WORK

5.1 Conclusions and Discussion

In Chapter 2, we provide an overview of the technical view of cloud, including the definition of cloud computing, cloud computing deployment models and service models, latency in the cloud, and pricing schemes in the cloud computing service market. This chapter helps us understand more of cloud computing definitions and why latency and pricing are important in the cloud market. The following two chapters study latency-aware pricing in cloud storage from a CSP's perspective and in cloud computing from a user's point of view respectively in the cloud market.

In Chapter 3, in order to store the files in the *cold storage* or *hot storage*, we propose a systematic framework for two-stage, latency-dependent bidding, which aims to maximize the cloud storage provider's net profit in tiered cloud storage systems where tenants may have different budgets, access patterns and performance requirements. In the proposed two-stage, latency-aware bidding mechanism, the users can bid for storage and access, in two separate stages, without knowing how the CSP stores the contents. The proposed optimization is modeled as a mixed-integer nonlinear program, for which an efficient heuristic is proposed. The numerical results demonstrate that the profits obtained from the proposed method are higher than those of other methods, and the access request acceptance rate also dominates that of other methods as the capacity of the cold storage or the service rate of hot storage increases.

In practice, it is possible that users may collude and modify their bidding behavior for maximizing own objective. The analysis of our two-stage pricing under strategic users and untruthful bids is an interesting future direction. Further, we consider a Poisson arrival process when users come to retrieve their data in the cloud. Analyzing

access latency under general service time distribution and its impact of pricing will also be considered in our future work.

Another interesting direction for the future is to extend the model for erasure coding storage system where multiple copies (n) of the files can be stored and a subset of those copies (k) are required to be fetched to get the original file. In that case, the CSP would need to select the n storage systems to store the file and among those k copies are needed to be fetched to get the original file. There are also other competing market places in the real world, the users may have other decision variables, such market selection. We do not provide any competitive analysis, however, we believe that such analysis constitutes an interesting future direction.

In Chapter 4, we develop optimization models to minimize the cost by randomizing between the on-demand, and spot instance. We also provide a mechanism to bid optimally in the spot instance. We consider three different strategies: one-time requests with expected guarantee, one-time requests with penalty, and persistent requests. We characterize the optimal portion of the job that should be run in the on-demand instance. Our analytical result shows that a user should never opt for on-demand instance if the execution time is smaller than the deadline. However, when the the execution time is larger than the deadline, the user should select the on-demand instance with a non-zero portion of the job. Additionally, the bid price also increases as the execution time increases. However, the portion to be run on the on-demand instance never exceeds half for one-time request without penalty and the persistent request scenario. Our numerical experiments shows the trade-off between higher prices to avoid interruptions (for one-time request without penalty), higher prices to avoid penalty (for one-time request with penalty) and lower prices to save money (for persistent request) on the condition that the deadline requirement is satisfied. It also shows that the user's cost is the lowest in the persistent request scenario. Finally, we use real world data to test our model and analytical results to verify our models and shows that the user can significantly reduces its cost both in the one-time and the persistent request scenario.

Some cloud service providers may give a two-minute warning to the user before the instance is revoked [130]. The user can use re-bidding strategy to avoid termination of the job. We do not consider such scenario in this paper, however, we believe that such analysis constitutes an interesting future direction. In order to prevent the job from revocation, some user may choose to run each spot request on multiple machines. However, it is not clear how many machines the user should choose and how much to bid. Thus, it can be another interesting research direction.

5.2 Future Work - Resource Allocation and Pricing in Fog Computing

Because of the proliferation of Internet of Things (IoT), the data generated by various sensors and applications has been increased enormously. Each company is utilizing the big data analytics tool to do the analysis of the collected data to extract useful insights in order to make essential decisions [131]. However, because of the requirements of the latency-aware computation for real-time application processing in IoT, the current cloud's capability is not that competitive. The new computing paradigm – fog computing, an extension of the cloud computing scheme from the core to the edge of network proposed by Cisco [132], can help overcome the above problem in the cloud by utilizing the idle resources of the devices in the neighborhood to support the utilization of storage, processing, and networking [133]. Fig.5.1 presents a basic model for fog computing, which has three layers: cloud, fog and IoT devices /end-users. The fog layer can have one or more fog domains, which include the fog devices with computing, storage and network connectivity, such as edge routers, switches, smartphones, etc. [134]. Please see [134] for more information about the fog system.

Technically, fog computing is similar to cloud computing since both of them provide on-demand provisioning of storage, processing and networking resources. However, compared to cloud computing, fog computing is geographically closer to the users. In order to show the characteristics of fog computing, we provide the com-

parison of fog and public cloud computing resources, which is summarized in Table 5.1. In terms of latency or the quality of service demanded by real-time applications requiring quick response, we can see that cloud computing has limitations because of the low network capacity and slow data travel time. Please see more detailed information in [135].

We have seen more computers than people in the world since 2015, but because of the development of cloud computing technology, a large amount of the computing power is untapped [136]. Hence the devices, like smartphones, switches, routers, and other devices having processing power and storage capacity, have the potential to serve as fog devices in the fog layer presented the Fig. 5.1 [137]. That provides an opportunity for the individual cloud service providers (ICSPs) to transfer their untapped computing power to money. As a consequence, there emerges a new marketplace where the ICSPs can provide their untapped computing resources and users pay for the resources to finish tasks. For example, ActiveAether, bringing the devices (such as phones, computers, servers, etc.) together to the cloud, aims to use latent computing power on the devices all around us and enable any computer to host software services and process data [136].

ActiveAether mainly focuses on the market that is composed of ICSPs and end users, which is circled in Fig 5.1. Because of the heterogeneous of the devices existing in the fog computing, one of the key challenges in that market is resource allocation and task scheduling. However, fog computing research is still in its infancy [137]. We propose several research directions related to the market that is composed of ICSPs and end users.

First we consider a static or single-period resource allocation in fog computing: ICSPs and users enter the market at the same time and a restricted time horizon is set to the market operations. Further, we can consider a dynamic or multiple-period resource allocation in fog computing: a market evolves dynamically where ICSPs and users arrive sequentially. Both of them can be resolved with the following centralized and decentralized approaches.

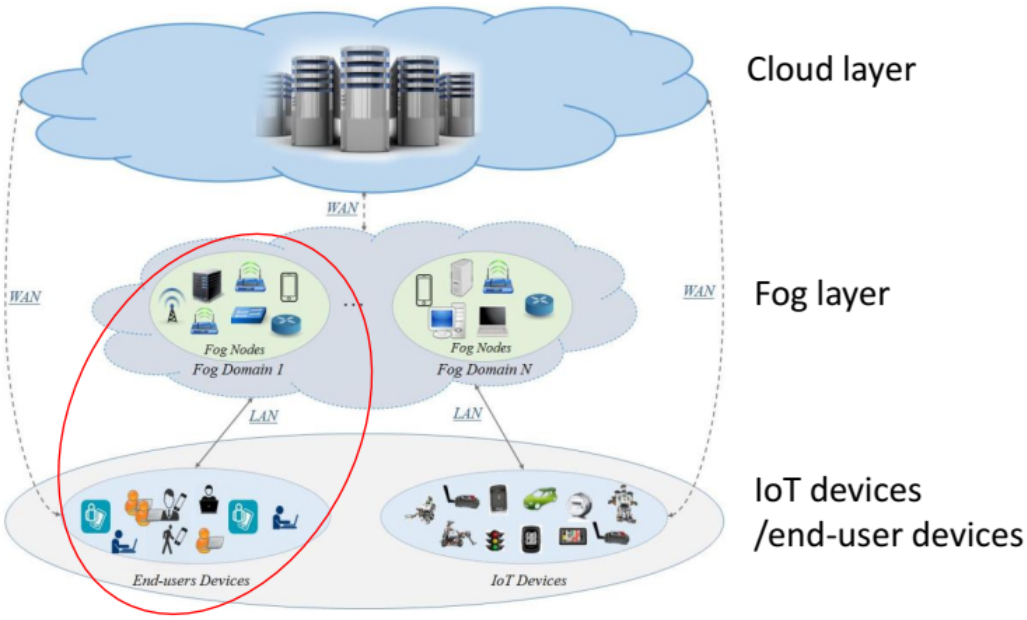


Fig. 5.1.: A model of fog computing [134]

Table 5.1.: Comparison of different cloud computing and fog computing [135]

	Cloud	Fog
Network Capacity	Low	High as possible
Data Travel Time	Slow	Fast as possible
Cost Structure	Measured on usage	Measured on usage
Scalability	As good as infinite	As good as infinite
Geographical Distribution	Centralized	Decentralized

5.2.1 Centralized Approach

In fog computing, we consider there is a third-party, which is a central controller between ICSPs and users. The exchange service is delivered through a platform or e-marketplace, where the ICSPs and users can share their bids, demands or capacities, and the third-party can service as a central controller to facilitate the resource allocation and pricing while meeting the users' requirements including latency.

5.2.2 Decentralized Approach

The development of blockchain technology, which is organized as a virtual Peer-to-Peer (P2P) network with maintained purely decentralized manner [138,139], has made it possible for transactions among individuals in the network even though there is no centralized controller. Gale and Shapley introduce the two-side matching market and a solution concept of *stability*. Using a simple algorithm *deferred acceptance algorithm*, they also show a stable matching always exists [140]. One of the future research directions can be developing efficient resource allocation scheme in a decentralized mode for the market based on the deferred acceptance algorithm, while the results from the the centralized approach can service as the upper bound for that of the decentralized approach.

REFERENCES

REFERENCES

[1] S. Conn, https://www.gartner.com/en/newsroom/press-releases/2018-09-12-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2019, 2018, gartner Forecasts Worldwide Public Cloud Revenue to Grow 17.3 Percent in 2019.

[2] F. Richter, https://www.statista.com/chart/9174/amazon-operating-profit/, March.2019, cloud Business Drives Amazon's Profits.

[3] R. Space, https://readyspace.com/.

[4] GoGrid, https://www.datapipe.com/gogrid.

[5] Amazon, https://aws.amazon.com, amazon Web Services.

[6] Microsoft, https://azure.microsoft.com/en-us/?v=17.36, microsoft windows azure.

[7] Google, https://cloud.google.com/storage/pricing, 2017, accessed 8th Feb,2017.

[8] R. Minnear, http://tomx.inf.elte.hu/twiki/pub/Team/CloudComputing/114241WP-IP-Achilles-Heel-Latency.pdf, 2011, latency The Achilles Heel of Cloud Computing.

[9] M. Al-Roomi, S. Al-Ebrahim, S. Buqrais, and I. Ahmad, "Cloud computing pricing models: a survey," *International Journal of Grid and Distributed Computing*, vol. 6, no. 5, pp. 93–106, 2013.

[10] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinl, W. Michalk, and J. Stoper, "Cloud computing – a classification, business models, and research directions," *Business and Information Systems Engineering*, vol. 1, no. 5, pp. 391–399, October 2009.

[11] AWS, "In shift, uber seeks bids for cloud business," *The information*, 2016. [Online]. Available: https://www.theinformation.com/articles/in-shift-uber-seeks-bids-for-cloud-business?token=de605b9fd69a938a04d3ca9331a496ea99a96151

[12] Y. Zhang, A. Ghosh, V. Aggarwal, and T. Lan, "Tiered cloud storage via two-stage, latency-aware bidding," *IEEE Transactions on Network and Service Management*, Oct. 2018.

[13] Y. Zhang, A. Ghosh, and V. Aggarwal, "Optimized portfolio contract for bidding the cloud." *IEEE Transactions on Cloud Computing*, Dec. 2018.

[14] M. Kader, "Securing the public & private cloud," *CISCO*, 2008. [Online]. Available: https://slideplayer.com/slide/5713468/

[15] P. Mell and T. Grance, https://csrc.nist.gov/projects/cloud-computing, 2009, draft NIST working definition of cloud computing.

[16] D. Milojicic, "Cloud computing: Interview with russ daniels and franco travostino," *IEEE Internet Computing*, vol. 5, pp. 7–9, September 2008.

[17] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared." Austin, TX,USA: Grid Computing Environments Workshop, 2008.

[18] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, January 2009.

[19] P. Gerald and G. Robert, "Formal requirements for virtualizable third generation architectures," *Communications of the ACM*, vol. 17, no. 7, pp. 412–421, 1974.

[20] AWS, "Launch a linux virtual machine with amazon ec2." [Online]. Available: https://aws.amazon.com/getting-started/tutorials/launch-a-virtual-machine/

[21] B. Sharma, R. K. Thulasiram, P. Thulasiraman, and S. K. Garg, "Pricing cloud compute commodities: A novel financial economic model," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. Washington, DC, USA: ACM, 2012, pp. 451–457.

[22] B. P. Rimal and E. Choi, "A conceptual approach for taxonomical spectrum of cloud computing," in *Proceedings of the 4th International Conference on Ubiquitous Information Technologies & Applications*. Fukuoka, Japan: IEEE, 2009, pp. 1–6.

[23] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *J Internet Serv Appl*, vol. 1, pp. 7–18, 2010.

[24] S. Goyal, "Public vs private vs hybrid vs community - cloud computing: A critical review," *I.J. Computer Network and Information Security*, vol. 3, pp. 20–19, 2014.

[25] O. Hamren, "Mobile phones and cloud computing," *M.S. Thesis*, 2012.

[26] W. Jansen and T. Grance, "Guidelines on security and privacy in public cloud computing," *NIST special publication 800-144*, 2011.

[27] Z. Zhou, H. Zhang, X. Du, P. Li, and X. Yu, "Prometheus: Privacy-aware data retrieval on hybrid cloud," in *In: Proceedings of IEEE INFOCOM*. IEEE, 2013, pp. 2643–2651.

[28] A. Tripathi and M. Jalil, "Data access and integrity with authentication in hybrid cloud," *Oriental International Journal of Innovative Engineering Research*, vol. 1, no. 1, 2013.

[29] Google Compute Engine, https://cloud.google.com/compute/.

[30] Windows Azure Virtual Machines, https://azure.microsoft.com/en-us/services/virtual-machines/.

[31] Amazon CloudFormation, https://aws.amazon.com/cloudformation/.

[32] Amazon Elastic Beanstalk, https://aws.amazon.com/elasticbeanstalk/.

[33] Google App Engine, https://cloud.google.com/appengine/.

[34] Windows Azure Compute, https://azure.microsoft.com/en-us/?v=17.44.

[35] L. L. Peterson and B. S. Davie, *Computer Networks, A Systems Approach*. Elsevier Science (USA), 2003.

[36] O2b Networks, "What is network latency and why dose it matter?" *White paper, O3b Networks*, 2008.

[37] Y. Xiang, T. Lan, V. Aggarwal, and Y. F. R. Chen, "Joint latency and cost optimization for erasurecoded data center storage," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 2, pp. 3–14, Sep. 2014. [Online]. Available: http://doi.acm.org/10.1145/2667522.2667524

[38] G.Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE Journal on Selected Areas in Communication*, 2014.

[39] Zona Research, "The need for speed 2," *AMCIS 2003 Proceedings*, 2001.

[40] R. B. Miller, "Response time in man-computer conversational transactions." San Francisco, California: AFIPS Proceedings of the fall joint computer conference, 1969, pp. 267–277.

[41] F. Nah, "A study on tolerable waiting time: How long are web users willing to wait?" *AMCIS 2003 Proceedings*, 2003.

[42] Forrester Consulting, "ecommerce web site performance today: An updated look at consumer reaction to a poor online shopping experience." *White paper, Akamai Technologies Inc.*, 2009.

[43] G.Linden, http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html, November, 2006, marissa Mayer at Web 2.0.

[44] G. Linden, https://www.gduchamp.com/media/StanfordDataMining.2006-11-28.pdf, November 2006, make your data useful, Amazon.

[45] D. Strom and J. F. van der Zwet, "Truth and lies about latency in the cloud." [Online]. Available: https://www.interxion.com/globalassets/-documents/whitepapers-and-pdfs/cloud/WP-TRUTHANDLIES-en-0715.pdf

[46] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 158 – 171, November 2013.

[47] C. Storage, https://searchstorage.techtarget.com/definition/cloud-storage, 2016.

[48] R. Bias, http://cloudscaling.com/assets/pdf/cloudscaling-whitepaper-tiered-storage-private-clouds.pdf, the Case for tiered storage in private clouds.

[49] gong zhang, lawrence chiu, and ling liu, "Adaptive data migration in multi-tiered storage based cloud environment," *2010 IEEE 3rd International Conference on Cloud Computing*, vol. IEEE comuter society, pp. 148–155.

[50] CloudBerry Lab Blog, https://www.cloudberrylab.com/blog/amazon-s3-azure-and-google-cloud-prices-compare/, August 2017.

[51] Google, https://cloud.google.com/, google Cloud Platform.

[52] G. C. pricing, https://cloud.google.com/pricing/list.

[53] Amazon, http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-requests.html, amazon EC2 spot requests.

[54] A. Luca and M. Bhide, *Storage virtualization for dummies, Hitachi Data Systems Edition*. John and Wiley Publishing, 2009.

[55] Y. Xiang, T. Lan, V. Aggarwal, and R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *IEEE/ACM Trans. Netw*, vol. 24, no. 4, pp. 2443–2457, 2016.

[56] J. Guerra, H. Pucha, W. J.Glider, and R. Rangaswami, "Cost effective storage using extent based dynamic tiering," in *In Proceedings of the 9th USENIX Conference on File and Stroage Technologies*. Usenix Association, 2011, pp. 20–20.

[57] H. Kim, S. Seshadri, C. Dickey, and L. Chiu, "Evaluating phase change memory for enterprise storage systems: study of caching and tiering approaches," in *In Proceedings of the 12th USENIX Conference on File and Storage Technologies*. Santa Clara, CA, USA: Usenix Association, 2014, pp. 33–45.

[58] Z. Li, A. Mukker, and E. Zadok, "On the importance of evaluating storage systems' $ costs," in *In Proceedings of the 6th USENIX Conference on Hot Topics in Storage and File Systems*. Philadelphia PA USA: Usenix Association, 2014, pp. 6–6.

[59] H. Wang and P. Varman, "Balancing fairness and efficiency in tiered storae systems with bottleneck-aware allocation," in *In Proceedings of the 12th USENIX Conference on File and Storage Technologies*. Santa Clara, CA, USA: Usenix Association, 2014, pp. 229–242.

[60] O. Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing amazon ec2 spot instance pricing," in *In Proceedings of the IEEE 3rd International Conference on Cloud Computing Technology and Science*. Athens, Greece: CloudCom 2011, 2011, pp. 304–311.

[61] I. Drago, M. Mellia, M. Munafó, A. Sperotto, R. Sadre, and A. Pras, "Inside dropbox: understanding personal cloud storage services," in *In Proceedings of the 12th ACM SIGCOMM Conference on Internet Measurement*. Boston, MA, USA: IMC 12, 2012, pp. 481–494.

[62] L. Youseff, M. Butrico, and D. D. Silva, "Toward a unified ontology of cloud computing," in *2008 Grid Computing Environments Workshop*, Nov 2008, pp. 1–10.

[63] M. Naldi and L. Mastroeni, "Cloud storage pricing: A comparison of current practices," in *Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services*, ser. HotTopiCS '13. New York, NY, USA: ACM, 2013, pp. 27–34. [Online]. Available: http://doi.acm.org/10.1145/2462307.2462315

[64] Amazon, https://aws.amazon.com/s3/pricing/, 2017, accessed 8th Feb,2017.

[65] Dropbox, https://www.dropbox.com/business/pricing, 2017, accessed 8th Feb,2017.

[66] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2348 – 2362, Aug. 2016.

[67] R. Zhou, Z. Li, and C. Wu, "An online procurement auction for power demand response in storage-assisted smart grids," in *IEEE INFOCOM 2015- IEEE Conference on Computer Communications*, April 2015, pp. 2641–2649.

[68] Q. Wu, M. Zhou, Q. Zhu, and Y. Xia, "Vcg auction-based dynamic pricing for multigranularity service composition," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 796–805, 2018.

[69] Q. Ma, Y.-F. Liu, and J. Huang, "Time and location aware mobile data pricing," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2599–2613, October 2016.

[70] H. R. Varian and C. Harris, "The VCG auction in theory and practice," *The American Economic Review*, vol. 104, no. 5, pp. 442–445, 2014.

[71] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. Incline Village NV USA: IEEE, 2010, pp. 1–10.

[72] D. Ongaro, S. M. Rumble, R. Stutsman, J. Ousterhout, and M. Rosenblum, "Fast crash recovery in ramcloud," in *SOSP 2011 Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, Cascais, Portugal, 2011, pp. 29–41.

[73] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *SOSP 2003 Proceedings of the 19th ACM Symposium on Operating Systems Principles*, New York, USA, 2003, pp. 29–43.

[74] D. Niu, Z. Liu, B. Li, and S. Zhao, "Demand forecast and performance prediction in peer-assisted on-demand streaming systems," in *2011 Proceedings IEEE INFOCOM Mini-Conference*. Shanghai, China: IEEE, 2011, pp. 421–425.

[75] G. G*ü*rsun, M. Crovella, and I. Matta, "Describing and forecasting video access patterns," in *2011 Proceedings IEEE INFOCOM Mini-Conference*. Shanghai, China: IEEE, 2011, pp. 16–20.

[76] G. Dan and N. Carlsson, "Dynamic content allocation for cloud-assisted service of periodic workloads," in *2014 Proceedings IEEE INFOCOM*. Toronto, ON, Canada: IEEE, 2014, pp. 853–861.

[77] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, E. Hyytiä, and A. Scheller-Wolf, "Reducing latency via redundant requests: Exact analysis," in *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Portland, OR, USA, 2015, pp. 347–360.

[78] A. G. Yue Cheng, M. Safdar lqbal and A. R. Butt, "Pricing games for hybrid object stores in the cloud: Provider vs. tenant," in *7th USENIX Workshop on Hot Topics in Cloud Computing*, Santa Clara, CA, 2015.

[79] V. Aggarwal, J. Fan, and T. Lan, "Taming tail latency for erasure-coded, distributed storage systems," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[80] V. Aggarwal, Y.-F. R. Chen, T. Lan, and Y. Xiang, "Sprout: A functional caching approach to minimize service latency in erasure-coded storage," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3683–3694, 2017.

[81] Y. Xiang, T. Lan, V. Aggarwal, and Y.-F. Chen, "Optimizing differentiated latency in multi-tenant, erasure-coded storage," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 204–216, 2017.

[82] A. O. Al-Abbasi and V. Aggarwal, "Video streaming in distributed erasure-coded storage systems: Stall duration analysis," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1921–1932, 2018.

[83] Y. Xiang, V. Aggarwal, Y.-F. Chen, and T. Lan, "Differentiated latency in data center networks with erasure coded files through traffic engineering," *IEEE Transactions on Cloud Computing*, 2017.

[84] B. Haeupler, V. S. Mirrokni, and M. Zadimoghaddam, "Online stochastic weighted matching: improved approximation algorithms," *WINE 2011: Internet and Network Economics*, pp. 170–181, 2011.

[85] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 4, pp. 546–559, May 2003.

[86] V. Aggarwal and T. Lan, "Tail index for a distributed storage system with pareto file size distribution," *CoRR*, vol. abs/1607.06044, 2016. [Online]. Available: http://arxiv.org/abs/1607.06044

[87] N. Shalom, "Amazon found every 100ms of latency cost them 1% in sales." http://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/, 2008, accessed 8th Feb. 2017.

[88] W. Chan, T.-C. Lu, and R.-J. Chen, "Pollaczek-khinchin formula for the m/g/1 queue in discrete time with vacations," *IEE Proceedings-Computers and Digital Techniques*, vol. 144, no. 4, pp. 222–226, 1997.

[89] R. J.-B. Wets, "Stochastic programs with fixed recourse: The equivalent deterministic program," *SIAM Review*, vol. 16, no. 3, pp. 309–339, 1974.

[90] E. M.B.Smith and C. C. Pantelides, "Global optimization of nonconvex minlps," *Computers & Chemical Engineering*, vol. 21, pp. S791–S796, May 1997.

[91] H. Ryoo and N. Sahinidis, "Global optimization of nonconvex nlps and minlps with applications in process design," *Computers & Chemical Engineering*, vol. 19, no. 5, pp. 551–566, May 1995.

[92] R. Karuppiah and I. Grossman, "A lagrangian based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures," *Computers & Chemical Engineering*, vol. 41, no. 2, pp. 163–186, July 2008.

[93] O. Exler, L. Antelo, J. Egea, A. Alonso, and J. Banga, "A lagrangian based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures," *Computers & Chemical Engineering*, vol. 32, pp. 1877–1891, 2008.

[94] C. Young, Y. Zheng, C. Yeh, and S.-S. Jang, "Information-guided genetic algorithm approach to the solution of minlp problems," *Industrial & Engineering Chemistry Research*, vol. 46, no. 5, pp. 1527 – 157, Feb. 2007.

[95] A. S. Drud, "Conopt: A large-scale grg code," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 207–216, 1994.

[96] A. J. and C. J., *Generalization of the Wolfe Reduced Gradient Method for Nonlinear Constraints*, e. Optimization (R. Fletcher, Ed. Academic Press, 1969, pp. 37-47.

[97] A. Drud, "A grg code for large sparse dynamic nonlinear optimization problems," *Mathematical Programming*, vol. 31, pp. 153–191, 1985.

[98] ——, "Conopt-a large scale grg code," *ORSA Journal on Computing*, vol. 6, pp. 207–216, 1992.

[99] W. Chen and L. Sha, "An energy-aware data-centric generic utility based approach in wireless sensor networks," in *In Proceedings of the third international symposium on Information processing in sensor networks*. Berkeley California USA: IPSN, 2004, pp. 215–224.

[100] L. Columbus, "Roundup of cloud computing forecasts," 2017.

[101] L. Zheng, C. J. Wong, C. G. Brinton, C. W. Tan, S. Ha, and M. Chiang, "On the viability of cloud virtual service provider," in *SIGMETRICS*, France, 2016.

[102] Amazon, https://aws.amazon.com/ec2/pricing/, amazon EC2 Pricing.

[103] H. Jin, X. Wang, S. Wu, S. Di, and X. Shi, "Towards optimized fine-grained pricing of iaas cloud platform," *IEEE Transactions on cloud computing*, vol. 3, no. 4, pp. 436–448, october 2015.

[104] G. Feng, S. Garg, R. Buyya, and W. Li, "Revenue maximization using adaptive resource provisioning in cloud computing environments." ACM/IEEE 13th International Conference on Grid Computing, 2012, pp. 192–200.

[105] P. Wang, Y. Qi, D. Hui, L. Rao, and X. Liu, "Present or future: optimal pricing for spot instances." IEEE 33rd International Conference on Distributed Computing Systems, 2013, pp. 410–419.

[106] B. Sharma, R. K. Thulasiram, P. Thulasiraman, and S. K. Garg, "Pricing cloud compute commodities: A novel financial economic model." Ottawa, ON, Canada: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2012, pp. 451–457.

[107] L. Zheng, C. J. Wong, C. W. Tan, M. Chiang, and X. Wang, "How to bid the cloud," in *SIGCOMM*, London, United Kingdom, 2015, pp. 71–84.

[108] D.Li, C. Chen, Y. Zhang, J. Zhu, and R.Yu, "Dcloud: Deadline-aware resource allocation for cloud computing jobs," pp. 2248–2260, 2016.

[109] B. M. E, "Famine early warning systems and remote sensing data," in *Springer*, 2011, p. 121.

[110] W. Wang, M. Barnard, and L. Ying, "Decentralized scheduling with data locality for data-parallel computation on peer-to-peer networks." Monticello, IL, USA: 2015 53rd Annual Allerton Conference on Communication, Control, and Computing, 2015, pp. 337–344.

[111] J. Tan, X. Meng, and L. Zhang, "Delay tails in mapreduce scheduling." London, England, UK: SIGMETRICS, 2012, pp. 5–16.

[112] Y. Chen, "Optimal dynamic auctions for display advertising," pp. 897–913, 2017.

[113] C. Bilington, "Hp cuts risk with portfolio approach," *Purchasing.com*, February 2002.

[114] M. de Albeniz and S.-L. D., "A portfolio approach to procurement contracts," *Production and Operations Management*, vol. 14, no. 1, pp. 90–114, 2005.

[115] Q. Fu, C.-Y. Lee, and C. piaw Teo, "Procurement management using option contracts: random spot price and the portfolio effect," *IIE Transactions*, vol. 42, no. 11, pp. 793–811, 2010.

[116] D. Li, C. Chen, Y. Zhang, J. Zhu, and R. Yu, "Dcloud: Deadline–aware resource allocation for cloud computing jobs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 8, pp. 2248–2260, August 2016.

[117] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222–235, 2014.

[118] R. N. Calheiros and R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1787–1796, September 2013.

[119] R. Abbott and H. Garcia-Molina, "Scheduling real-time transactions: a performance evaluation." Los Angeles, California, USA: Proceedings of the 14th VLDB Conference, 1988, pp. 1–12.

[120] R. Zhou, Z. Li, C. Wu, and Z. Huang, "An efficient cloud market mechanism for computing jobs with soft deadlines," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 793–805, November 2017.

[121] M. Hadji, W. Louati, and D. Zeghlache, "Constrained pricing for cloud resource allocation." Cambridge, MA, USA: Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on, 2011, pp. 359–365.

[122] A. A. Daoud, S. Agarwal, and T. Alpcan, "Brief announcement: Cloud computing games: Pricing services of large data centers." Berlin, Heidelberg: Distributed Computing, Spring 2009, pp. 309–310.

[123] D. Ardagna, B. Panicucci, and M. Passacantando, "A game theoretic formulation of the service provisioning problem in cloud systems." India: WWW 2011, 2011, pp. 177–186.

[124] ——, "Generalized nash equilibria for the service provisioning problem in cloud systems," *IEEE TRANSACTIONS ON SERVICES COMPUTING*, vol. 6, no. 4, pp. 429–442, October 2013.

[125] E. Kutanoglus and D. Wu, "On combinatorial auction and lagrangean relaxation for distributed resource scheduling," *IIE Transactions*, vol. 31, no. 9, pp. 813–826, 1999.

[126] K. Song, Y. Yao, and L. Golubchik, "Exploring the profit-reliability trade-off in amazon's spot instance market: A better pricing mechanism." Montreal, QC, Canada: Quality of Service (IWQoS), 2013 IEEE/ACM 21st International Symposium on, 2013.

[127] Q. Wang, K. Ren, and X. Meng, "When cloud meets ebay: Towards effective pricing for cloud computing." 2012 Proceedings IEEE INFOCOM, 2012, pp. 936–944.

[128] M. Kojima and A. Takeda, "Complexity analysis of successive convex relaxation methods for nonconvex sets," *Mathematics of Operations Research*, vol. 26, no. 3, pp. 519–542, 2001.

[129] Amazon, "Easy amazon ec2 instance comparison," *https://www.ec2instances.info/*.

[130] C. Wang, B. Urgaonkar, A. Gupta, G. Kesidis, and Q. Liang, "Exploiting spot and burstable instances for improving the cost-efficacy of in-memory caches on the public cloud." Belgrade, Serbia: In EuroSys'17, 2017, pp. 620–634.

[131] J. Chen, Y. Chen, X. Du, C. Li, J. Lu, S. Zhao, and X. Zhou, "Big data challenge: a data management perspective," *Frontiers of Computer Science*, vol. 2, pp. 157–164, 2013.

[132] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," *in Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM*, pp. 13–16, 2012.

[133] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: Principles, architectures, and applications," *Internet of Things: Principle & Paradigms, M. Kaufmann, Ed., USA*, 2016.

[134] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: Stateof-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2018.

[135] M. Firdhous, O. Ghazali, and S. Hassa, "Fog computing: Will it be the future of cloud computing?" in *Proceedings of the Third International Conference on Informatics & Applications*, Kuala Terengganu, Malaysia, 2014, pp. 8–15.

[136] R. Maclnnis, A. Boyd, and S. Cody, "Fogcoin: Enabling a global market for computing power."

[137] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47 980–48 009, 2018.

[138] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[139] W. Wang, D. T. Hoang, Z. Xiong, D. Niyato, P. Wang, P. Hu, and Y. Wen, "A survey on consensus mechanisms and mining management in blockchain networks," 2018. [Online]. Available: arXiv preprint arXiv:1805.02707

[140] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly*, vol. 69, pp. 9–15, 1962.

APPENDICES

A. PROOFS

A.1 Moments of the service time of a file request

Proof In this Appendix, we will derive the first and second moments of the service time of a file request at storage j in k-th scenario, which will be used further to prove Theorem 3.4.1. More precisely, we will show the following result.

Lemma 10 *X_j^k, the service time of a file request at storage j in k-th scenario,has a distribution with mean*

$$\mathrm{E}[X_j^k] = \frac{\sum_i \lambda_i^k \pi_{i,j}^k S_i}{\mu_j \sum_i \lambda_i^k \pi_{i,j}^k} \tag{A.1}$$

and second moment

$$\mathrm{E}[(X_j^k)^2] = \frac{2\sum_i \lambda_i^k \pi_{i,j}^k S_i^2}{\mu_j^2 \sum_i \lambda_i^k \pi_{i,j}^k}. \tag{A.2}$$

The rest of the Section proves this result.

It is easy to verify that under our model, the arrival of file requests at storage j in k-th scenario forms a Poisson Process with rate $\Lambda_j^k = \sum_i \lambda_i^k \pi_{i,j}^k$, which is the superposition of I Poisson Processes each with rate $\lambda_i^k \pi_{i,j}^k$.

Let S^r be the (random) requested file size at storage j, which is a discrete random variable such that the probability of $S^r = S_i$ is $\frac{\lambda_i^k \pi_{i,j}^k}{\sum_i \lambda_i^k \pi_{i,j}^k}$. Let V_j^t be the (random) service time of one MB at storage j, which is exponentially distributed with mean $\frac{1}{\mu_j}$. The the expectation of the service time of a file request at storage j is

$$
\begin{aligned}
\mathrm{E}[X_j^k] &= \mathrm{E}_{V_j^t}[\mathrm{E}_{S^r}[X_j|S^r = S_i]] \\
&= \sum_{S^r} \mathrm{E}_{V_j^t}[X_j|S^r = S_i]\mathrm{P}\{S^r = S_i\} \\
&= \sum_{S^r} \mathrm{E}_{V_j^t}[S^r V_j^t|S^r = S_i]\mathrm{P}\{S^r = S_i\} \\
&= \sum_i \frac{S_i}{\mu_j} \frac{\lambda_i^k \pi_{i,j}^k}{\sum_i \lambda_i^k \pi_{i,j}^k} \\
&= \frac{\sum_i \lambda_i^k \pi_{i,j}^k S_i}{\mu_j \sum_i \lambda_i^k \pi_{i,j}^k}
\end{aligned}
\tag{A.3}
$$

and the associated second moment is

$$
\begin{aligned}
\mathrm{E}[(X_j^k)^2] &= \mathrm{E}_{V_j^t}[\mathrm{E}_{S^r}[X_j^2|S^r = S_i]] \\
&= \sum_{S^r} \mathrm{E}_{V_j^t}[X_j^2|S^r = S_i]\mathrm{P}\{S^r = S_i\} \\
&= \sum_{S^r} \mathrm{E}_{V_j^t}[(S^r)^2 (V_j^t)^2|S^r = S_i]\mathrm{P}\{S^r = S_i\} \\
&= \sum_i S_i^2 \frac{2}{\mu_j^2} \frac{\lambda_i^k \pi_{i,j}^k}{\sum_i \lambda_i^k \pi_{i,j}^k} \\
&= \frac{2\sum_i \lambda_i^k \pi_{i,j}^k S_i^2}{\mu_j^2 \sum_i \lambda_i^k \pi_{i,j}^k}.
\end{aligned}
\tag{A.4}
$$

■

A.2 Proof of Theorem 3.5.1

Proof Consider a special case of problem (P1), where a special case where assume that there is a single scenario, and thus the bids for the second stage are perfectly known. Further, for this scenario, the optimal decisions of $\pi_{i,j}^k$ and H_i^k are assumed to be known. Since all this information is genie-aided, this only simplifies the problem. In this case, the problem (P1) reduces to the following.

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}(P_i - 2S_i c_1)A_i + \sum_i R_i S_i (c_1 - c_2) \\ \text{subject to} \quad & (3.1) - (3.3) \\ & A_i \in \{0,1\}, R_i \in \{0,1\} \end{aligned}$$

This problem is equivalent to a two-dimensional knapsack problem. Hence, the problem is NP-hard.

Now, we show that the second stage problem (P2) is also NP-hard. Assume that the optimal values of $\pi_{i,j}^k$ are known, which is a genie-aided information and can only simplify the problem. In this case, it can be seen that the second stage problem (P2) is equivalent to a Knapsack problem for the variables H_i^k, which is a NP-hard problem thus proving the NP-hardness of (P2).

A toy example will further demonstrate the complexity of (P1). Suppose we have 10 users, and 1 time period with 3 scenarios, so here we have 50 binary decision variables ($10A$, $10R$, $10H^1$, $10H^2$ and $10H^3$). The total number of branches in a decision tree is $(2^{10})^5 = 2^{50}$, which increases exponentially with the increase of number of users and scenarios. Thus, the problem scale is very large even with a small number of users, scenarios and time periods. In addition, the latency constraint is nonlinear, which makes our problem even harder. ■

A.3 Proof of Lemma 1

Proof

$$
\begin{aligned}
g_1(0) &= \frac{1}{1+1} - \frac{1}{1+\exp(-\alpha)} + \frac{1}{2} - \frac{1}{1+\exp(\alpha)} && \text{(A.5)}\\
&= \frac{\exp(-\alpha)}{1+\exp(-\alpha)} - \frac{1}{1+\exp(\alpha)} && \text{(A.6)}\\
&= 0. && \text{(A.7)}
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
g_1(1) &= \frac{1}{1+\exp(\alpha)} - \frac{1}{1+1} + \frac{1}{2} - \frac{1}{1+\exp(\alpha)} && \text{(A.8)}\\
&= 0. && \text{(A.9)}
\end{aligned}
$$

■

A.4 Proof of Proposition 3.5.2

Proof Let $A_i(C,\alpha), R_i(C,\alpha)$, and $H_i^k(C,\alpha)$ be the optimal solution of (P1) for given C, and α. Note that if we down-quantize all solutions to 0, it is easy to see that will be a feasible solution. We, thus, consider the situation where the optimal solution will be converted to 1. Note that for every C, the solution will be at most away from the integer solutions by β amount. Thus, if all the solutions are converted to 1, then the storage space is increased by

$$\sum_i S_i\beta \tag{A.10}$$

Hence, if $C_j' = C_j(1-\epsilon_1)$ where, $\epsilon_1 = \sum_i S_i\beta/C_j$, we would obtain a feasible solution of A_i, and R_i.

Now, we will proceed to find the amount by which the service time is increased for converting solutions H_i^k to 1. The additional amount of service time while converting solutions to 1, is

$$\max_k(\sum_i \lambda_i^k S_i\beta) \tag{A.11}$$

Hence, if $\mu_j' = \mu_j(1-\epsilon_2)$ where, $\epsilon_2 = \max_k(\sum_i \lambda_i^k S_i\beta)/\mu_j$ would give a feasible solution. Hence, the result follows. Note that β decreases as C and α increases, and eventually becomes zero as $C \to \infty$. ■

A.5 Proof of Lemma 4

Proof Suppose $g(p) = \mathbb{E}(\pi|\pi \leq p) = \frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)}$, we take the first-order derivative of $g(p)$ over p and get $\frac{\partial g(p)}{\partial p} = \frac{f_\pi(p)}{F_\pi(p)^2}(pF_\pi(p) - \int_{\underline{\pi}}^{p} x f_\pi(x)dx)$. Suppose $h(p) = pF_\pi(p) - \int_{\underline{\pi}}^{p} x f_\pi(x)dx$, we take the first derivative of $h(p)$ over p and get $\frac{\partial h(p)}{\partial p} = F_\pi(p) \geq 0$, which means $h(p)$ monotonically increases with p and the minimum value of $h(p)$ is $h_{min}(p) = h(\underline{\pi}) = 0$. Because $\frac{f_\pi(p)}{F_\pi(p)^2} \geq 0$, $\frac{\partial g(p)}{\partial p} \geq 0$. Therefore, $g(p)$ monotonically increase with p. The maximum value of $g(p)$ is $g_{\max}(p) = g(\bar{\pi}) = \frac{\int_{\underline{\pi}}^{\bar{\pi}} x f_\pi(x)dx}{F_\pi(\bar{\pi})} \leq \frac{\underline{\pi}+\bar{\pi}}{2}$. When $\underline{\pi}$ is closed to 0, $g_{max}(p) \leq \frac{\bar{\pi}}{2}$. ■

A.6 Proof of Claim 1

We will prove this result by contradiction. Suppose π_1 is the expected optimal bid price and $q_1 > \frac{1}{2}$ is the optimal portion of job running on on-demand instance, and the associated optimal objective value is

$$obj_1 = q_1 t_e \bar{\pi} + (1 - q_1) t_e E[\pi | \pi \leq \pi_1].$$

Now, we show that we can achieve a lower value by employing a strategy different to the above one. Note that since $q_1 > 1/2$. Thus, the value of obj_1 is at least $t_e\bar{\pi}/2$.

Thus, there exists a solution $q \leq 1/2$ such that $qt_e\bar{\pi} + (1-q)t_e\bar{\pi}/2 = obj_1$. Now, consider the strategy $q^* = q - \epsilon$, and the bidding price $p = \bar{\pi}$. Since $p = \bar{\pi}$, thus, $t_n = 0$. Now consider $q = q_1 - \dfrac{t_s}{t_e}$. Since $t_e/2 < t_s < t_e$, thus, $0 < q < 1/2$. The bid price be $\bar{\pi}$. The above strategy satisfies all the constraints. The objective value is thus at most $obj_2 = (q-\epsilon)t_e\bar{\pi} + (1-q+\epsilon)t_e\bar{\pi}/2$ which is less than obj_1. Hence, we obtain a lower value by employing a different strategy. Hence, the strategy is not optimal. Therefore, $q^* \leq \frac{1}{2}$.

Now, we show that $F_\pi(p) \geq 1/2$ for an optimal bidding. From constraint (4.2)

$$\begin{aligned} & (1-q^*)t_e \leq \frac{t_k}{1 - F_\pi(p^*)} \\ \Longleftrightarrow\ & (1-q^*)t_e F_\pi(p^*) \geq (1-q^*)t_e - t_k \\ \Longleftrightarrow\ & t_k(1 - F_\pi(p^*)) + (1-q^*)t_e F_\pi(p^*) \geq (1-q^*)t_e - t_k F_\pi(p^*) \end{aligned} \tag{A.12}$$

From constraint (4.4)

$$\begin{aligned} & t_k(\frac{1}{F_\pi(p^*)} - 1) + (1-q^*)t_e \leq t_s \\ \Longleftrightarrow\ & t_k(1 - F_\pi(p^*)) + (1-q^*)t_e F_\pi(p^*) \leq t_s F_\pi(p^*) \end{aligned} \tag{A.13}$$

According to (A.12) and (A.13), we can get that

$$(t_s + t_k)F_\pi(p^*) \geq (1-q^*)t_e \geq \frac{t_e}{2} \tag{A.14}$$

Because $t_s + t_k \leq t_e$, we can get $F_\pi(p^*) \geq \frac{1}{2}$.

A.7 Proof of Proposition 4.3.1

Proof By taking the first-order derivative of $\Phi_1(p,q)$ over q, we have

$$\frac{\partial \Phi(p,q)_1}{\partial q} = t_e(\bar{\pi} - \frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)}) \geq 0. \tag{A.15}$$

Therefore, $\Phi_1(p,q)$ increases monotonically with q, the user can minimize his expected total cot by choosing the smallest possible q in the feasible set. First we consider constraints (cf. (4.2)) and (cf. (4.4)), we get

$$q^* = \max\{1 - \frac{t_k}{t_e(1-F_\pi(p))}, 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}\},$$

then we will go back to check constraints (cf. (4.7)) and (cf. (4.9)).

When

$$1 - \frac{t_k}{t_e(1-F_\pi(p))} \leq 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e} \tag{A.16}$$

$q^* = 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}$.

Then we substitute q in (P5) with q^*, (P5) becomes

$$\text{(P5')} \quad \text{minimize} \quad G(p) = [t_s - t_k(\frac{1}{F_\pi(p)} - 1)][\frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)} - \bar{\pi}] + t_e\bar{\pi} \tag{A.17}$$

$$\text{subject to} \quad t_s - t_k(\frac{1}{F_\pi(p)} - 1) \leq \frac{t_k}{1-F_\pi(p)} \tag{A.18}$$

$$1 - \frac{t_s}{t_e} + \frac{t_k}{t_e}(\frac{1}{F_\pi(p)} - 1) \leq \frac{t_s}{t_e} \tag{A.19}$$

$$\underline{\pi} \leq p \leq \bar{\pi} \tag{A.20}$$

$$0 \leq 1 - \frac{t_s}{t_e} + \frac{t_k}{t_e}(\frac{1}{F_\pi(p)} - 1) \leq 1 \tag{A.21}$$

In terms of constraint (A.19),

$$\begin{aligned} & 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e} \leq \frac{t_s}{t_e} \\ \iff & \frac{t_k}{t_k - t_e + 2t_s} \leq F_\pi(p^*) \\ \iff & \frac{t_k}{2t_s - t_e + t_k} \leq \frac{t_k}{t_k + t_k} = \frac{1}{2} \leq F_\pi(p^*) \end{aligned} \tag{A.22}$$

which has been proved in Claim 1. Thus,

$$q^* = 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e} \leq \frac{t_s}{t_e} \leq 1.$$

Also

$$q^* = 1 - \frac{t_s}{t_e} + \frac{t_k}{t_e}(\frac{1}{F_\pi(p^*)} - 1) \geq 1 - \frac{t_s}{t_e} \geq 0$$

because $0 \leq F_\pi(p^*) \leq 1$.

Thus constraints (A.19) and (A.21) hold when the optimization model (P5') takes the optimal value.

Next, we will consider objective function $G(p)$ in (P5').

$$\begin{aligned}\frac{\partial G(p)}{\partial p} &= \frac{t_k f_\pi(p)}{F_\pi(p)^2}[\frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)} - \bar{\pi}] + \frac{f_\pi(p)[pF_\pi(p) - \int_{\underline{\pi}}^{p} x f_\pi(x)dx]}{F_\pi^2(p)}[t_s - t_k(\frac{1}{F_\pi(p)} - 1)] \\ &= \frac{f_\pi(p)}{F_\pi(p)^2}[\frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)} t_k - \bar{\pi}t_k + (pF_\pi(p) - \int_{\underline{\pi}}^{p} x f_\pi(x)dx)(t_s - t_k(\frac{1}{F_\pi(p)} - 1))]\end{aligned} \tag{A.23}$$

Suppose $g(p) = \frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)} t_k - \bar{\pi}t_k + (pF_\pi(p) - \int_{\underline{\pi}}^{p} x f_\pi(x)dx)(t_s - t_k(\frac{1}{F_\pi(p)} - 1))$

$$\frac{\partial g(p)}{\partial p} = \frac{2t_k f_\pi(p)}{F_\pi(p)}(p - \frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)}) + F_\pi(p)(t_s - t_k(\frac{1}{F_\pi(p)} - 1)) \tag{A.24}$$

It is clear that $p \geq \frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)}$, and since $F_\pi(p*) \geq \frac{1}{2}$, $t_s - t_k(\frac{1}{F_\pi(p)} - 1) \geq t_s - t_k \geq 0$, $\frac{\partial g(p)}{\partial p} \geq 0$. Thus $g(p)$ monotonically increases with p. Because $\lim_{p \to \underline{\pi}} g(p) = t_k\underline{\pi} - t_k\bar{\pi} < 0$, and $g(\bar{\pi}) = (\bar{\pi} - \int_{\underline{\pi}}^{\bar{\pi}} x f_\pi(x)dx)(t_s - t_k) > 0$, $g(p)$ increases monotonically from a negative value to a positive value. The term $\frac{t_k f_\pi(p)}{F_\pi(p)^2} > 0$ in $\frac{\partial G(p)}{\partial p}$, thus $\frac{\partial G(p)}{\partial p}$ also increases monotonically from a negative value to a positive value, i.e., $G(p)$ first decreases and then increases with p. Thus $G(p)$ is minimized when $\frac{\partial G(p)}{\partial p} = 0$. Letting $\frac{\partial G(p)}{\partial p} = 0$, we thus deduce

$$\begin{aligned}\psi_1(p) &= \frac{2t_k \int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)} + 2pt_s F_\pi(p) - pt_s - (t_s + t_k)\int_{\underline{\pi}}^{p} x f_\pi(x)dx \\ &= t_k\bar{\pi}\end{aligned} \tag{A.25}$$

Thus the objective function $G(p)$ takes the optimal value at

$$\hat{p_1}^* = \psi_1^{-1}(t_k\bar{\pi})$$

regardless of other constraints.

Further, we will analyze constraint (A.18), which is equivalent to

$$\psi_2(p) = (t_s + t_k)F_\pi(p) - (t_s + t_k)F_\pi(p)^2 - t_k \leq 0 \tag{A.26}$$

Because

$$\frac{\partial \psi_2(p)}{\partial p} = (t_s + t_k)f_\pi(p)(1 - 2F_\pi(p)) \leq 0 \tag{A.27}$$

$\psi_2(p)$ is monotone decreasing with p.

Also because $\lim_{p:F_\pi(p)\to\frac{1}{2}} \psi_2(p) \geq 0$ and $\psi_2(\bar{\pi}) \leq 0$, there exists one and only one $\hat{p_2}^*$ such that $\psi_2(\hat{p_2}^*) = 0$, so constraint (A.18) (or (A.36))is equivalent to

$$p \geq \hat{p_2}^*$$

In order to get the optimal solution for (P5'), we need to take the maximum of $\hat{p_1}^*$ and $\hat{p_2}^*$, i.e., $p^* = \max\{\hat{p_1}^*, \hat{p_2}^*\} = \max\{\psi_1^{-1}(t_k\bar{\pi}), \psi_2^{-1}(0)\}$.

Finally we will go back to check whether condition (A.28) can be satisfied with $p^* = \max\{\psi_1^{-1}(t_k\bar{\pi}), \psi_2^{-1}(0)\}$. We need to consider the following two cases:

Case 1: when $\psi_2^{-1}(0) \geq \psi_1^{-1}(t_k\bar{\pi})$, $p^* = \psi_2^{-1}(0)$, that is $(t_s + t_k)F_\pi(p^*) - (t_s + t_k)F_\pi(p^*)^2 - t_k = 0$, which is equivalent to $1 - \frac{t_k}{t_e(1-F_\pi(p))} = 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}$. Thus the condition (A.28) holds in Case 1.

Case 2: when $\psi_1^{-1}(t_k\bar{\pi}) \geq \psi_2^{-1}(0)$, $p^* = \psi_1^{-1}(t_k\bar{\pi})$. According to (A.27), we know that $\psi_2(p)$ is monotone decreasing with p. Thus $\psi_2(\psi_1^{-1}(t_k\bar{\pi})) < \psi_2(\psi_2^{-1}(0)) = 0$, which is equivalent to $1 - \frac{t_k}{t_e(1-F_\pi(p))} < 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}$. Therefore, the condition (A.28) also holds in Case 2.

Then we will consider when

$$1 - \frac{t_k}{t_e(1 - F_\pi(p))} \geq 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e} \tag{A.28}$$

$q^* = 1 - \frac{t_k}{t_e(1-F_\pi(p))}$

Then we substitute q in (P5) with q^*, (P5) becomes

$$\text{(P5")} \quad \text{minimize} \quad G_2(p) = (1 - \frac{t_k}{t_e(1-F_\pi(p))})t_e\bar{\pi} + \frac{t_k \int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)(1-F_\pi(p))} \tag{A.29}$$

$$\text{subject to} \quad 1 - \frac{t_k}{t_e(1-F_\pi(p))} \geq 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e} \tag{A.30}$$

$$1 - \frac{t_k}{t_e(1-F_\pi(p))} \leq \frac{t_s}{t_e} \tag{A.31}$$

$$\underline{\pi} \leq p \leq \bar{\pi} \tag{A.32}$$

$$0 \leq 1 - \frac{t_k}{t_e(1-F_\pi(p))} \leq 1 \tag{A.33}$$

Next, we take the first derivative of the objective function in (P5"),

$$\frac{\partial G_2(p)}{\partial p} = \frac{f_\pi(p)t_k[-\bar{\pi}F_\pi(p)^2 + pF_\pi(p)(1-F_\pi(p)) - (1-2F_\pi(p))\int_{\underline{\pi}}^{p} x f_\pi(x)dx]}{F_\pi(p)^2(1-F_\pi(p))^2} \tag{A.34}$$

Suppose $g_2(p) = -\bar{\pi}F_\pi(p)^2 + pF_\pi(p)(1-F_\pi(p)) - (1-2F_\pi(p))\int_{\underline{\pi}}^{p} x f_\pi(x)dx$, Now, we will prove $g_2(p) \leq 0$

$$\begin{aligned} & pF_\pi(p)(1-F_\pi(p)) + (2F_\pi(p)-1)\int_{\underline{\pi}}^{p} x f_\pi(x)dx \\ =& F_\pi(p)[p(1-F_\pi(p)) + (2F_\pi(p)-1)\frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)}] \\ \leq& F_\pi(p)[\bar{\pi}(1-F_\pi(p)) + \bar{\pi}(2F_\pi - 1)] \\ =& \bar{\pi}F_\pi(p)^2 \end{aligned} \tag{A.35}$$

Thus, $g_2(p) = pF_\pi(p)(1-F_\pi(p)) + (2F_\pi(p)-1)\int_{\underline{\pi}}^{p} x f_\pi(x)dx - \bar{\pi}F_\pi(p)^2 \leq 0$ Then we can get $\frac{\partial G_2(p)}{\partial p} \leq 0$, thus $G_2(p)$ decreases monotonically with p, and the optimal solution is the largest possible p in its feasible set.

Because $1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e} \geq 1 - \frac{t_s}{t_e} > 0$. Thus, if we can meet constraint (A.30), constraint (A.33) will be met.

As we analyzed in the last part, constraint (A.30) is equivalent to

$$\psi_2(p) = (t_s + t_k)F_\pi(p) - (t_s + t_k)F_\pi(p)^2 - t_k \leq 0 \tag{A.36}$$

which is monotone decreasing with p. Thus the largest feasible solution will be obtained when $\psi_2(p) = 0$. Therefore, the optimal solution $p^* = \psi_2^{-1}(0)$.

In summary,

- When $1 - \frac{t_k}{t_e(1-F_\pi(p))} \leq 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}$,
  $p^* = \max\{\psi_1^{-1}(t_k\bar{\pi}), \psi_2^{-1}(0)\}$, $q^* = 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}$;
- When $1 - \frac{t_k}{t_e(1-F_\pi(p))} \geq 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}$,
  $p^* = \psi_2^{-1}(0)$, $q^* = 1 - \frac{t_k}{t_e(1-F_\pi(p))} = 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}$

We combine the above two cases: $p^* = \max\{\psi_1^{-1}(t_k\bar{\pi}), \psi_2^{-1}(0)\}$, $q^* = 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}$. Thus Proposition 4.3.1 is proved. ■

A.8 Proof of Proposition 4.3.2

Proof Recalling (A.15) in Proposition 4.3.1, we know that $\Phi_1(p,q)$ increases monotonically with q, the user can minimize his expected total cost by choosing the smallest possible q in the feasible set. That is,

$$q^* = \max\{1 - \frac{t_k}{t_e(1-F_\pi(p))}, 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e}, 0\}.$$

When

$$1 - \frac{t_k}{t_e(1-F_\pi(p))} \leq 0 \tag{A.37}$$

and

$$1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e} \leq 0 \tag{A.38}$$

$q^* = 0$

When $q = 0$, the objective function of (P5) becomes

$$G(p) = \frac{t_e \int_{\underline{\pi}}^{p} x f_\pi(x) dx}{F_\pi(p)}$$

Then we take the first derivative of $G(p)$ and get

$$\frac{\partial G(p)}{\partial p} = t_e f_\pi(p)(\frac{p - \frac{\int_{\underline{\pi}}^{p} x f_\pi(x) dx}{F_\pi(p)}}{F_\pi(p)}) \geq 0,$$

which means the $G(p)$ monotonic increasing with p. Minimizing $G(p)$ is equivalent to finding the minimum p in its feasible set. Suppose $t_s - t_e \geq t_k$ and $t_e \geq 2t_k$, then $p^* = \max\{F_\pi^{-1}(1 - \frac{t_k}{t_e}), F_\pi^{-1}(\frac{t_k}{t_s - t_e + t_k}), \underline{\pi}\} = F_\pi^{-1}(1 - \frac{t_k}{t_e})$, which is equivalent to

$$1 - \frac{t_k}{t_e(1-F_\pi(p))} = 0,$$

thus condition (A.37) is satisfied.

Next we will check whether condition (A.38) will be satisfied.

$$\begin{aligned} 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e} &\leq 1 - \frac{t_e + t_k}{t_e} + \frac{t_k}{t_e}(\frac{t_k}{t_e} - 1) \\ &= \frac{t_k(2t_k - t_e)}{t_e(t_e - t_k)} \\ &\leq 0 \end{aligned}$$

Thus condition (A.38) is also satisfied. This proves the result as in the statement of Proposition 4.3.2. ■

A.9 Proof of Lemma 5

Proof When the part of job that is assigned to spot instance is finished before the deadline t_s, there will be no penalty for the late job. When the job is finished but late for t_k, then the associated penalty is

$$W_1(p,q) = (1 - F_\pi(p))^{\frac{t_s}{t_k} - K(q)+1} F_\pi(p)^{K(q)} t_k;$$

when the job is finished but late for it_k, the associated penalty is

$$W_i(p,q) = (1 - F_\pi(p))^{\frac{t_s}{t_k} - K(q)+i} F_\pi(p)^{K(q)} i t_k.$$

Considering all the possibilities when the job is finished but late for t_k, $2t_k$, ... we will get the total completed but late job is

$$\begin{aligned} L(p,q) &= t_k(1 - F_\pi(p))^{\frac{t_s}{t_k} - K(q)} F_\pi(p)^{K(q)} \sum_{i=1}^{\infty} i(1 - F_\pi(p))^i \\ &= t_k(1 - F_\pi(p))^{\frac{t_s}{t_k} - K(q)+1} F_\pi(p)^{K(q)-2} \end{aligned} \tag{A.39}$$

Thus Lemma 5 is proved.

■

A.10 Proof of Lemma 6

Proof First when the time to enter the system is 0, that is, the time from bid submission to the first win ($p \geq \pi(t)$) is 0, from first win, the probability for the first lose ($p < \pi(t)$) follows Geometric distribution, thus, the expected portion of the job that is completed is

$$\begin{aligned} EC_0(p,q) &= t_k \sum_{i=1}^{K(q)-1} iF_\pi(p)^i(1-F_\pi(p)) + K(q)t_kF_\pi(p)^{K(q)} \\ &= \frac{F_\pi(p) - F_\pi(p)^{K(q)+1}}{1-F_\pi(p)} t_k \end{aligned}$$

However, if the job has to wait t_k, $2t_k$,...it_k... amount before the bid is accepted in the spot instance, there is an extra probability $(1-F_\pi(p))^i$ before the first bid wins. Thus, the total time it takes for the expected portion of the job to be complete is given by–

$$\begin{aligned} EC(p,q) =& EC_0(p,q) \sum_{i=0}^{\infty} (1-F_\pi(p))^i \\ =& \frac{1-F_\pi(p)^{K(q)}}{1-F_\pi(p)} t_k \end{aligned}$$

Thus Lemma 6 is proved. ■

A.11 Proof of Claim 2

Proof Recall that with bid price p and deadline t_s, $F_\pi(p)$ denotes the probability that the bid price $p \geq \pi(t)$, the spot price, the job's expected running time on spot instance is $t_s F_\pi(p)$. In order to guarantee the job can be finished before deadline, $t_s F_\pi(p^*) \geq (1-q^*)t_e$,

$$\begin{aligned} & t_s F_\pi(p^*) \geq (1-q^*)t_e \\ \Longleftrightarrow\ & F_\pi(p^*) \geq (1-q^*)\frac{t_e}{t_s} \\ \Longleftrightarrow\ & F_\pi(p^*) \geq 1-q^* \\ \Longleftrightarrow\ & F_\pi(p^*) \geq \frac{1}{2} \end{aligned} \tag{A.40}$$

Thus Claim 2 is proved. ■

A.12 Proof of Proposition 4.3.3

Proof When $\frac{t_e}{2} < t_s < t_e$, we take the first-order derivative of $\Phi_3(p,q)$ over q and get

$$\begin{aligned}\frac{\partial \Phi_3(p,q)}{\partial q} &= t_e(\bar{\pi} - \frac{1}{1-\frac{t_r}{t_k}(1-F_\pi(p))}\frac{\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)}) \\ &\geq t_e(\bar{\pi} - \frac{2\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)})\end{aligned} \tag{A.41}$$

Suppose $g(p) = \bar{\pi} - \frac{2\int_{\underline{\pi}}^{p} x f_\pi(x)dx}{F_\pi(p)}$, and take the first-order of derivative of $g(p)$, we get

$$\frac{\partial g(p)}{\partial p} = -\frac{2f_\pi(p)}{F_\pi(p)^2}(pF_\pi(p) - \int_{\underline{\pi}}^{p} x f_\pi(x)dx) \leq 0$$

thus $g(p)$ monotonically decrease with p. The minimum value of $g(p)$ is $g_{min}(p) = g(\bar{\pi}) = \bar{\pi} - \frac{2\int_{\underline{\pi}}^{\bar{\pi}} x f_\pi(x)dx}{F_\pi(p)} \geq 0$. Then we can get

$$\frac{\partial \Phi_3(p,q)}{\partial q} \geq 0. \tag{A.42}$$

Therefore, $\Phi_3(p,q)$ monotonically increases with q. Minimizing $\Phi_3(p,q)$ is equivalent to finding the minimum q in its feasible set, that is,

$$q^* = \max\{1 - \frac{t_s F_\pi(p)(1-\frac{t_r}{t_k}(1-F_\pi(p)))}{t_e}, 0\}$$

Because

$$1 - \frac{t_s F_\pi(p)(1-\frac{t_r}{t_k}(1-F_\pi(p)))}{t_e} \geq 1 - \frac{t_s}{t_e} \geq 0, \tag{A.43}$$

$$q^* = 1 - \frac{t_s F_\pi(p)(1-\frac{t_r}{t_k}(1-F_\pi(p)))}{t_e} \tag{A.44}$$

Then substitute q using q^* in (P7), we will get a new optimization problem (P7')

$$\text{(P7')} \quad \text{minimize} \quad G(p) = t_e\bar{\pi} - t_s F_\pi(p)\bar{\pi}[1 - \frac{t_r}{t_k}(1 - F_\pi(p))] + t_s \int_{\underline{\pi}}^{p} f_\pi(x)dx \tag{A.45}$$

$$\text{subject to} \quad (1 - \frac{t_r + t_s F_\pi(p)(1 - \frac{t_r}{t_k}(1 - F_\pi(p)))}{t_e})t_e \leq t_s \tag{A.46}$$

$$t_r < \frac{t_k}{2(1 - F_\pi(p))} \tag{A.47}$$

$$\underline{\pi} \leq p \leq \bar{\pi} \tag{A.48}$$

$$1 - \frac{t_s F_\pi(p)(1 - \frac{t_r}{t_k}(1 - F_\pi(p)))}{t_e} \geq 0 \tag{A.49}$$

Take the first-order derivative of $G(p)$, we have

$$\frac{\partial G(p)}{\partial p} = (p - \bar{\pi})t_s f_\pi(p) + \frac{t_r t_s}{t_k}\bar{\pi} f_\pi(p)(1 - 2F_\pi(p)) < 0,$$

so $G(p)$ monotonic decreasing with p. Thus, the optimal solution is $p^* = \bar{\pi}$. In addition, the constraints (A.46) and (A.49) are satisfied at optimality.

Substitute p with $p^* = \bar{\pi}$ in (A.44), we will get $q^* = 1 - \frac{t_s}{t_e} \geq 0$. This proves the result given in the statement of Proposition 4.3.3. ■

A.13 Proof of Proposition 4.3.4

Proof From (A.42) we know that $\Phi_3(p,q)$ monotonically increases with q.

$$q^* = \max\{1 - \frac{t_s F_\pi(p)(1 - \frac{t_r}{t_k}(1 - F_\pi(p)))}{t_e}, 0\}$$

When

$$1 - \frac{t_s F_\pi(p)(1 - \frac{t_r}{t_k}(1 - F_\pi(p)))}{t_e} \leq 0, \tag{A.50}$$

$$q^* = 0,$$

then optimization problem (P7) will become

$$\text{(P7")} \quad \text{minimize} \quad \Phi(p) = \frac{t_e}{1 - \frac{t_r}{t_k}(1 - F_\pi(p))} \frac{\int_{\underline{\pi}}^{p} x f_\pi(x) dx}{F_\pi(p)} \tag{A.51}$$

$$\text{subject to} \frac{t_e}{1 - \frac{t_r}{t_k}(1 - F_\pi(p))} \frac{1}{F_\pi(p)} \leq t_s \tag{A.52}$$

$$\underline{\pi} \leq p \leq \bar{\pi} \tag{A.53}$$

By taking the first derivative of $\Phi(p)$ in (P7"), we will have

$$\frac{\partial \Phi(p)}{\partial p} = \frac{(t_e) f_\pi(p)(1 - \frac{t_r}{t_k} + 2\frac{t_r}{t_k} F_\pi(p))}{h(p)^2} g(p)$$

where

$$g(p) = -\int_{\underline{\pi}}^{p} x f_\pi(x) dx + p \frac{(1 - \frac{t_r}{t_k}) F_\pi(p) + \frac{t_r}{t_k}(F_\pi(p))^2}{1 - \frac{t_r}{t_k} + 2\frac{t_r}{t_k} F_\pi(p)}$$

and

$$h(p) = (1 - \frac{t_r}{t_k}) F_\pi(p) + \frac{t_r}{t_k}(F_\pi(p))^2$$

Because $\frac{(t_e) f_\pi(p)(1 - \frac{t_r}{t_k} + 2\frac{t_r}{t_k} F_\pi(p))}{h(p)^2} > 0$, in order to show the positivity of $\Phi(p)$, we take the first derivative of $g(p)$ and then we have

$$\frac{\partial g(p)}{\partial p} = \frac{1 - \frac{t_r}{t_k} + 2\frac{t_r}{t_k}(F_\pi(p) - p f_\pi(p))}{(1 - \frac{t_r}{t_k} + 2\frac{t_r}{t_k} F_\pi(p))^2} h(p)$$

Because $F_\pi(p)$ is concave and $F_\pi(p) - pf_\pi(p) \geq 0$, we have $\frac{\partial g(p)}{\partial p} \geq 0$. Thus, $g(p)$ monotonically increases with p. By the fact that $g(\underline{\pi}) = 0$, then $g(p) \geq 0$ and $\frac{\partial \Phi(p)}{\partial p} \geq 0$, i.e., $\Phi(p)$ increases monotonically with p.

In order to minimize the total cost, we just need to choose the lowest feasible bid price. Constraint (A.52) in (P7") is equivalent to

$$g(p) = \frac{t_r}{t_k} F_\pi(p)^2 + (1 - \frac{t_r}{t_k}) F_\pi(p) \geq \frac{t_e}{t_s}. \tag{A.54}$$

The axis of symmetry of $g(F_\pi(p))$ is $F_\pi(p) = -\frac{t_k - t_r}{2t_r} < 0$, thus $g(F_\pi(p))$ monotonically increases with $F_\pi(p)$ on the condition that $0 \leq F_\pi(p) \leq 1$ and $g(p)$ monotonically increases with p on the condition that $\underline{\pi} \leq p \leq \bar{\pi}$. So the minimum value to satisfy constraint (A.52) is $g(p^*) = \frac{t_e}{t_s}$.

Because $0 \leq \frac{t_e}{t_s} \leq 1$, the maximum and minimum value of $g(p)$ is $g_{max}(p) = g(\bar{\pi}) = 1$ and $g_{min}(p) = g(\underline{\pi}) = 0$ respectively. Therefore, constraint (A.53) is satisfied. $g(p^*) = \frac{t_e}{t_s}$ is equivalent to $1 - \frac{t_s F_\pi(p^*)(1 - \frac{t_r}{t_k}(1 - F_\pi(p^*)))}{t_e} = 0$, thus the condition (A.50) is satisfied. This proves the result as in the statement of Proposition 4.3.4. ■

A.14 Proof of Lemma 9

Proof Recall that when $t_s < t_e \leq 2t_s$, $F_\pi(p^*) \geq \frac{1}{2}$. The difference optimal portions of job to run on on-demand instance in Proposition 4.3.1 and Proposition 4.3.3 is

$$\begin{aligned} & 1 - \frac{t_s - t_k(\frac{1}{F_\pi(p^*)} - 1)}{t_e} - (1 - \frac{t_s}{t_e}) \\ =& \frac{t_k}{t_e}(\frac{1}{F_\pi(p^*)} - 1) \\ \leq& \frac{t_k}{t_e} \end{aligned} \tag{A.55}$$

Note the last step is because $F_\pi(p^*) \geq \frac{1}{2}$. Thus Lemma 9 is proved. ■

VITA

VITA

EDUCATION

Purdue University, West Lafayette, IN Aug. 2014 - May 2019
Ph.D. candidate in Operaions Research, School of Industrial Engineering
Advisor: Prof. Seokcheon Lee
Thesis: Latency-Aware Pricing in the Cloud Market

Southeast University, Nanjing, China Aug. 2010 - May 2014
B.S. in Information Management and Information Systems

WORKING EXPERIENCE

AT&T Labs - Research, Bedminster, NJ May 2018- July 2018
Student Intern - Technical II Mentors: Moo-Ryong Ra, Yu Xiang

- Analyzed the cache utilization in AT&T vCDN based on historical data.
- Implemented a segment tree based algorithm reducing the running time from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log(N))$.
- Gave suggestions for planning the edge placement in AT&T vCDN.

Tata Consultancy Services Innovation Labs, Cincinnati, OH June 2016 - Aug. 2016
Research and Development Intern Mentors: Devadatta M Kulkarni, Mark Zhuravel

- Designed a framework to support business decisions in product configuration planning.
- Developed a new hybrid Constraint Programming and Integer Programming approach.
- The simulation results proved its applicability and usefulness.

SELECTED PUBLICATIONS

1. **Y. Zhang**, A. Ghosh, V. Aggarwal, T. Lan, "Tiered cloud storage via two-stage, latency-aware bidding." IEEE Transactions on Network and Service Management, 16(1), Mar. 2019.
2. **Y. Zhang**, A. Ghosh, V. Aggarwal, "Optimized portfolio contracts for bidding the cloud." accepted in IEEE Transactions on Cloud Computing, Dec. 2018.

TECHNICAL STRENGTHS

Languages	Python, R, MATLAB, SQL	**Statistics**	Minitab
Optimization	GAMS, CPLEX, CVX	**Simulation**	Arena, AnyLogic
Data Analysis	NumPy, Pandas, VBA	**Machine Learning**	scikit-learn

SELECTED AWARDS AND HONORS

National Scholarship (the highest scholarship in China, among top 3%) Oct. 2012
Bosch Scholarship (awarded to only one student in the college each year) Oct. 2013
Excellent Student Leader Mar. 2012