Migrating Applications into Clouds: Components Factoring, Workload Scheduling and Incentives

Xuanjia Qiu
Department of Computer Science
The University of Hong Kong

Probation Report
for the degree of PhD

Oct. 2011

Signature from the supervisors:

1. Dr. Chuan Wu:

2. Prof. Francis C.M. Lau:

Abstract

The emerging of cloud-computing paradigm is rapidly gaining momentum due to its advantage on elasticity and cost-efficiency. In this report, we first survey the progress in the industry and academia on the field of cloud computing. Then, as a first-step investigation in the cloud computing field, we explore the problem of migrating content distribution serivce into geo-distributed hybrid clouds. In this problem, two key tasks are involved for such a move: to migrate their contents to cloud storage, and to distribute their web service load to cloud-based web services. The main challenge is to make the best use of the cloud as well as their existing on-premise server infrastructure, to serve volatile content requests with service response time guarantee at all times, with minimum operational cost incurred. Employing Lyapunov optimization techniques, an optimization framework is presented for dynamic, cost-minimizing migration of content distribution services into a hybrid cloud infrastructure that spans geographically distributed data centers. A dynamic control algorithm is designed, which optimally places contents and dispatches requests to different data centers to minimize overall operational cost over time, subject to service response time constraints. Rigorous analysis shows that the algorithm nicely bounds the response times within the preset QoS target in cases of arbitrary request arrival patterns, and guarantees that the overall cost is within a small constant gap from the optimum achieved by a T-slot lookahead mechanism with known information into the future. We verify the performance of our dynamic algorithm with extensive simulations under realistic settings. Finally, we identify a number of open problems which may be our research directions in the coming years.

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Chapter 1

Background

1.1 Introduction to Cloud Computing

Cloud Computing is a new term for a long-held dream of computing as a utility (**?** **?**), which has recently emerged as a commercial reality. According to (**?**), the following features consist of the key to the definition of cloud computing:

- Providing the illusion of infinite computing resources available on demand.
- Eliminating an up-front commitment by cloud users.
- Pay for use (or, called as "usage-based", "pay-as-you-go") of computing resources on a short-term basis as needed.

Cloud computing enables individuals or companies with market domain expertise to build and run a SaaS company with minimal effort developing software and without managing an hardware operations, which may grow from inception to a massive scale at incredible rates(**?**). The users pay no premium for scaling up and suffer from no penalty for scaling down.

Some pioneered researchers of grids have tried to extend the concept of grid to close to what cloud computing is like. However, in grids, many difficulties arise due to lack of a single owner of the whole system, while clouds are always assumed to be operated by single companies, even though we envision federated clouds facing similar problems as grids (**?**), so many new problem arise in cloud computing compared with grid computing.

1. BACKGROUND

Like the programming languages, different utility computing offerings will be distinguished based on the level of abstraction presented to the programmers and the level of management of the resources. These different abstractions would coexist for a long time. EC2 (**?**) by Amazon is a form of IaaS (Infrastructure as a Service), which provides abstraction close to a physical machine and lies at one end of the spectrum. AppEngine (**?**) by Google and Azure by Microsoft (**?**) are a form of PaaS (Platform as a Service). AppEngine lies at the other end of the spectrum which provides platform support for large-scale computation. Azure lies in the middle of the spectrum, which provides language-based abstraction.

There are three key roles in cloud computing:

- Cloud provider: a company which provides PaaS or IaaS.
- Cloud user (i.e., application provider, companies providing SaaS): the entity which uses PaaS or IaaS to deploy a SaaS, that is provided to end users or revoked by other SaaSes.
- End user: the individuals who use SaaS.

In the current cloud market, there exist three basic categories of instances of virtual machines (**?**):

- *On-demand instances*: On-Demand instances charge the user for compute capacity by the hour with no long-term commitments. This frees the user from the costs and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into much smaller variable costs. On-Demand instances also remove the need to buy safety net capacity to handle periodic traffic spikes (**?**).
- *Reserved instances*: *Reserved instances* enable users to maintain the benefits of elastic computing while lowering costs and reserving capacity. The user pays a one-time fee and in turn receives a significant discount on the hourly usage charge for that instance. *Reserved instances* can provide substantial savings over On-Demand instances when the utilization of the instance is high enough, as well as help assure that the capacity is available when required (**?**).

- *Spot instances*: In December 2009 Amazon EC2 launched spot instance service to sell its unused computing capacity with a market-driven resource allocation mechanism and received considerable attention (**? ? ? ?**). *Spot instances* are sold at dynamic spot prices. To use *spot instances*, a user places a *spot instance* request, specifying the instance type, the availability zone desired, the number of *spot instances* the user wants, and the maximum price the user are willing to pay per instance hour. The cloud provider sets the spot price periodically based on supply and demand, and customers whose bids exceed it gain access to the available *spot instances*. Generally the charge for a *spot instance* is lower than on-demand instances or *reserved instances*. However, although users run *spot instances* for as long as their bids exceed the current spot price, if their pre-set maximum acceptable prices no longer exceed the current spot price, the running *spot instances* will be terminated. Therefore, *spot instances* provide cloud users an option at lower prices without providing reliability on services.

1.2 State-of-Art of Cloud Computing

1.2.1 Content Clouds

Content distribution services are a major category of popular Internet applications. A growing number of content providers are contemplating a switch to cloud-based services, for better scalability and lower cost. As an important category of popular Internet services, content distribution applications, e.g., video streaming, web hosting and file sharing, feature large volumes of content and demands that are highly dynamic in the temporal domain.

MetaCDN (**?**) is a content distribution system based on content clouds that has many advantages over traditional CDN, e.g., no long-term contract, and lower charge rate per upload byte.

Bjorkqvist *et al.* (**?**) study the problem of minimizing the retrieval latency considering both caching and retrieval capacity of the edge nodes and server simultaneously. Their analytical models evaluate the content retrieval latency under different source selection strategies and different caching policies.

1.2.2 Capacity Provisioning of Resource and Schedule of Tasks

Although cloud computing eliminates the requirement for users to plan ahead for provisioning when the application is setup, the application provider still needs to predict the demand and compute the capacity provisioning for each component of the system, in order to minimize operational cost and comply with SLA (or to increase performance under the given budget constraint). This problem arises because the system is now elastic to scale up and down.

Urgaonkar *et al.* (**?** **?**) adopt a queueing model to compute the delay of a multi-tier system when workload and system configurations are given, and the bottleneck of the system can be identified. For a given total resource budget of an N-tier web application that can meet the end-to-end performance threshold, there exist degrees of freedom to partition the resource budget among the individual tiers. Well-designed partitioning scheme can achieve the best performance for the given total resource budget, or minimize the total resource cost without compromising the end-to-end performance. Based on (**?** **?**), *KingFisher* (**?**) develops a generalized provisioning framework for supporting elasticity in the cloud, which exploits the pricing differentials of various server configurations and chooses the most appropriate scaling mechanism to minimize costs. It assumes a multi-tier application whose workload demand change over time. *Kingfisher* can compute a new hardware configuration given a current configuration, that specifies how many cloud servers and of what types to use for each tier to sustain the new peak workloads at each tier. The optimization objectives are rental cost and transition cost. It outputs a cost-optimized configuration for the desired capacity as well as a plan for transitioning the application from its current setup to its new configuration. Xiong *et al.* (**?**) adopt a multi-level approach to address these the problem through the combination of the resource controllers on both application and container levels. On the application level, a decision maker determines the total budget of the resources that are required for the application to meet SLA requirements as the workload varies. On the container level, a second controller partitions the total resource budget among the components of the applications to optimize the application performance (i.e., to minimize the round trip time).

1.2.3 Categories of Virtual Machine Instances

Market resource allocation: Unused capacity is in a shared resource pool, but there are disjoint spot markets. Zhang *et al.* (**?**) discuss the problem of dynamically allocating data center resources to each spot market to maximize cloud provider's total revenue.

1.2.4 Geo-distributed Data Centers in a Cloud

Geo-diversity lowers latency to users and increases reliability in the presence of an outage taking out an entire site (**?**). Increment of delay in the scale of $100-500$ msecs would cause $1-20\%$ revenue loss according to reports from Google and Amazon, which creates a strong motivation for geo-distributed data centres round the world to reduce speed-of-light delays (**?**).

From the perspective of cloud providers, geo-diversity of data centers opens a door to the following challenges: determining where to place data centres; how big to make them; how to use the geo-diversity of data cetnres as a source of redundancy to improve system availability (**?**).

From the perspective of cloud users, similar problems arise: determining where to place data; how to schedule tasks to be executed. DONAR (**?**) is a distributed system for geo-distributed services that provides an effective way to direct client requests to a particular location, based on performance, load, and cost. It solves an optimization problem that jointly considers both client performance and server load.

Plume (**?**) builds a novel quorum ring overlay to implement fast message dissemination in large-scale geo-distributed clouds.

Automated mechanisms to place application data across geo-distributed datacenters is an increasingly urgent need. The placement must deal with WAN bandwidth costs and datacenter capacity limits, while minimizing user-perceived latency. The task of placement is further complicated by the issues of shared data, data inter-dependencies, and user mobility (**?**). Volley (**?**) is a system that computes the optimal data placement and migration by continousely analysing user request logs.

Large scale OSN (Online Social Network) is an example of data inter-dependencies. In large scale OSN, data should be stored on multiple VMs (**?**), which are possibly geo-distributed. The data is accessed following some specific pattern, frequency, and updated according the social relationship. The challenge is to partition (and replicate)

users' data and assign them into a number of geo-distributed VMs. Proposed partition algorithms make use of social relationship (**?**) and user access pattern (**?**) to achieve data locality while minimizing replication.

1.2.5 Collaboration in Federated Clouds

Although cloud computing provides an illusion of unlimited resource pool, inherently limited scalability of single-provider clouds still exists. If a cloud provider deploys too much resource, the resource would be underutilized; otherwise, saturated. Just as in other industries where no single provider serves all customers at all times, the next-generation cloud-computing infrastructure should support a model that enable multiple independent providers to cooperate seamlessly to meet a spike in demand and maximize their mutual benefit (**?**).

Reservoir project (**?**) is motivated by the vision of implementing an architecture that would enable providers of cloud infrastructure to dynamically partner with each other to create a seemingly infinite pool of IT resources while preserving their individual autonomy in making technological and business management decisions. The goal is to facilitate an open service-based online economy in which resources and services are transparently provisioned and managed across clouds on an on-demand basis at competitive costs with high-quality service.

1.2.6 Hybrid Clouds

Because of the "multi-tenant" properties, cloud users concern about the privacy and security of their data. To deal with that, various schemes are proposed, e.g., computation based on encryption (**?** **?**). Hybrid cloud architecture partially solves that problem by separating the application into two parts and placing the unimportant parts in the cloud while the important parts on-premise.

Hybrid clouds are generally defined as the federation of data centres provided by public clouds and other computing resource outside the cloud, provided by either on-premise server cluster or user's client devices. Specifically, there are multiple forms of hybrid clouds, including hybrid of cloud and on-premise server cluster, hybrid of cloud and PCs, hybrid of cloud and handsets. A task may be executed on the cloud, local device, or devices belonging to other users.

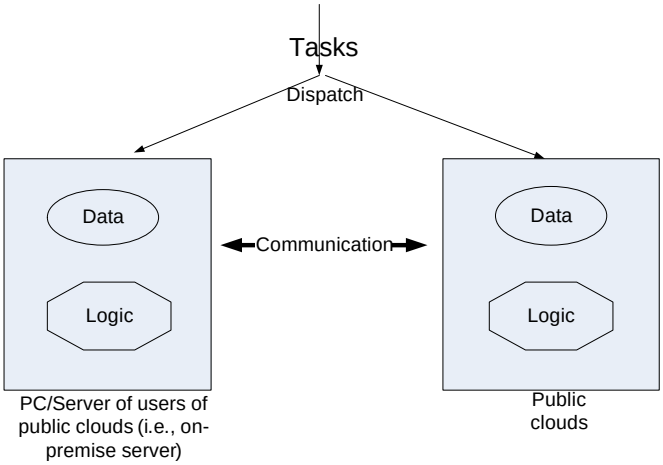

Figure 1.1: Hybrid cloud architecture

The hybrid cloud architecture not only benefits users in terms of protection of privacy and security, but also prevents existing private enterprise servers from being wasted once the application is completely migrated to the public clouds. When migrating an application into a hybrid cloud, the application may be deployed partially on the on-premise servers, and partially on the public clouds, as illustrated in Fig. 1.1

Research projects that explore the migration of services to hybrid clouds have been emerging in recent years. Hajjat *et al.* (**?**) develop an optimization model for migrating enterprise IT applications onto a hybrid cloud. Their model takes into account enterprise-specific constraints, such as cost savings, increased transaction delays, and wide-area communication costs. Zhang *et al.* (**?**) propose an intelligent algorithm to factor workload and dynamically determine the service placement across the cloud and the on-premise server. Their focus is on designing an algorithm for distinguishing base workload and trespassing workload. Cheng *et al.* (**?**) study the partition of social media contents and assignment of them onto a number of cloud servers, when migrating the social media distribution service to clouds. It focuses on load balance in terms of accesses by preserving social relationship and considering user access behaviour. Li *et al.* (**?**) propose cost saving by partial migration of VoD services to content clouds. Heuristic strategies are proposed to decide the update of cloud content and verified by trace-driven evaluations.CloudFlex (**?**) leverages a feedback control system that monitors system performance, and distributes load onto a heterogeneous

set of resources, which transparently taps cloud resources to serve application requests that exceed capacity of the on-premise server cluster.

1.2.7 Incoming and Outgoing Bandwidth Capacity of Clouds

Although clouds are powerful in provisioning large storage capacity and a huge number of CPUs, their provisioning Internet bandwidth is still very limited. We can foresee a trend that the cost of intra-cloud bandwidth consumption will keep decreasing faster than WAN bandwidth consumption in many coming years, due to different development paces of their underlying technologies. To save bandwidth cost in the WAN communication, mechanisms of predicting traffic is proposed (**?**). Besides traffic prediction, efficient caching mechanisms, implementing at either client side or cloud side, are yet to be developed to alleviate the high cost of WAN communication.

Chapter 2

Migrating Content Distribution Service to Geo-distributed Hybrid Content Clouds

Cloud computing technologies have enabled rapid provisioning and release of server utilities (CPU, storage, bandwidth) to users anywhere, anytime. To exploit diversity of electricity cost and to provide service proximity to users in different geographic regions, a cloud service often spans multiple data centers over the globe, *e.g.*, Amazon CloudFront (**?**), Microsoft Azure (**?**), Google App Engine (**?**). The elastic and on-demand nature of resource provisioning has made cloud computing attractive to providers of various applications. More and more new applications are being created on the cloud platform (**?**)(**?**)(**?**), while many existing applications are also considering the cloud-ward move (**?**)(**?**), including content distribution applications (**?**)(**?**).

As an important category of popular Internet services, content distribution applications, *e.g.*, video streaming, web hosting and file sharing, feature large volumes of content and demands that are highly dynamic in the temporal domain. A cloud platform with multiple, distributed data centers is ideal to host such a service, with substantial advantages over a traditional private or public content distribution network (CDN) based solution, in terms of more agility and significant cost reduction with respect to machines, bandwidth, and management. In this way, the application providers can focus their business more on content provision, rather than IT infrastructure maintenance.

2. MIGRATING CONTENT DISTRIBUTION SERVICE TO GEO-DISTRIBUTED HYBRID CONTENT CLOUDS

Two major components are found in a typical content distribution application, namely back-end storage for keeping the contents, and front-end web service to serve the requests. Both can be migrated to the cloud: contents can be stored in storage servers in the cloud, and requests can be distributed to cloud-based web services. Therefore, the key challenge for cloud-ward move of a content distribution application is how to efficiently replicate contents and dispatch requests across multiple cloud data centers and the provider's existing on-premise servers, for the modest operational expenditure and providing good service response time guarantee at all times.

It may not be too hard to design a simple heuristic for dynamic content placement and load distribution in the hybrid cloud; however, proposing one with cost optimality over a long run of the system rest assured, is an intriguing yet intimating challenge, especially when arbitrary arrival rates of requests are considered.

In this paper, we present an optimization framework for dynamic, cost-minimizing migration of content distribution services into a hybrid cloud, and design a joint content placement and load distribution algorithm that minimizes overall operational cost over time, subject to service response time constraints. Our design is rooted in Lyapunov optimization theory (**?**)(**?**), where cost minimization and response time guarantee are achieved simultaneously by efficient scheduling of content migration and request dispatching among data centers. Lyapunov optimization provides a framework for designing efficient algorithms that achieve arbitrarily close to optimal system performance over the long run, without a need for any information from the future. It has been extensively used in routing and channel allocation in wireless networks (**?**)(**?**)(**?**), and has only recently been introduced to address a few resource allocation scenarios in other types of networks (**?**)(**?**). We tailor Lyapunov optimization techniques in the hybrid cloud computing setting, to dynamically and jointly resolve the optimal content replication and load distribution problems.

The contribution of this work can be summarized as follows:

- ▷ To our best knowledge, this work is the first to propose a generic optimization framework based on Lyapunov optimization theory, for dynamic, optimal migration of a content distribution service to a hybrid cloud consisting of private on-premise servers and public geo-distributed cloud services.

- ▷ We design a joint content placement and load distribution algorithm for dynamic content distribution service deployment in the hybrid cloud. Providers of content distribution services can practically apply it to guide their service migration, with cost minimization and performance guarantee rest assured, regardless of the request arrival pattern.

- ▷ We demonstrate optimality of our algorithm with rigorous theoretical analysis. The algorithm nicely bounds the response times (including queueing and round-trip delays) within the preset QoS target in cases of arbitrary request arrivals, and guarantees that the overall cost is within a small constant gap from the optimum achieved by a T-slot lookahead mechanism with information from the future.

The rest of the chapter is organized as follows. We present the optimization model in Chapter 2.1. The joint content placement and load distribution algorithm and is designed and analyzed in Chapter 2.2 and Chapter 2.3, respectively. The performance of the algorithm is evaluated in Chapter 2.4. Finally, we conclude the paper in Chapter 2.5.

2.1 The Service Migration Problem

2.1.1 System Model

We consider a typical content distribution application providing a collection of contents (files), denoted as set $\mathcal{M}$, to users spanning multiple geographical regions. Let $v^{(m)}$ be the size (in bytes) of file $m \in \mathcal{M}$. There is an on-premise server cluster (or private cloud) owned by the provider of the content distribution application, which stores and serves the contents to users in a traditional fashion.

There is a public cloud that includes data centers located in a number of geographical regions, denoted as set $\mathcal{N}$. One data center resides in each region. Each data center has two types of servers: storage servers for data storage, and computing servers that support the running and provisioning of virtual machines (VMs). Servers inside the same data center can access each other via high-speed Ethernet switches and LAN buses.

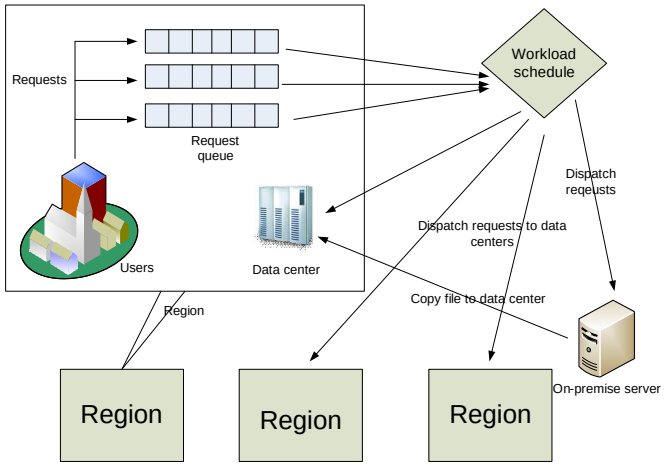

Figure 2.1: Illustration of the system model

The content distribution application provider (*application provider*) wishes to provision its service by exploiting a *hybrid cloud* architecture, which includes the geo-distributed public cloud and its on-premise server cluster. Specifically, the major components of a content distribution application are: (i) back-end storage of the contents and (ii) front-end web service that serve user requests for contents. The application provider may migrate both service components into the public cloud: contents can be replicated in storage servers in the cloud, while requests can be dispatched to web services installed on VMs on the computing servers. Fig. 2.1 gives an illustration of the system architecture.

Our objective in this paper is to design a dynamic, optimal algorithm for the application provider to strategically make the following decisions for service migration into the hybrid cloud architecture: (i) *content replication*: which content should be replicated in which data center at each time? (ii) *request distribution*: How many requests for a content should be directed to the on-premise server cluster and to each of the data centers that store this content at the time? The goal is to pursue the minimum operational cost for the application provider over time, while ensuring the service quality of content distribution.

We next develop an optimization framework to characterize the optimal content distribution service migration problem. Important notations are summarized in Table 2.1.

2.1.2 Cost-Minimization Service Migration Problem

Suppose the system executes in a time-slotted fashion. Each time slot is one unit time that is enough for uploading any file $m \in \mathcal{M}$ with size v_m at the unit bandwidth. In time slot t, $a_j^{(m)}(t)$ requests are generated for downloading file $m \in \mathcal{M}$, from users residing in region j. We assume that the request generation is an arbitrary random process over time, with A_{max} being the upper bound of the number of request arising from region j for file m in each time slot.

Without loss of generality, we use one server to represent the on-premise server cluster, which always stores an original copy of all contents in the content distribution system. The on-premise server has an overall upload bandwidth of b units, for serving contents to users. The cost of uploading a byte from the server is h.

At each data center i of the public cloud, we assume the storage capacity is sufficient for storing contents from this content distribution application. The charge of storage is p_i per byte per time slot. g_i and o_i per byte are charged for uploading from and downloading onto the data center, respectively. The cost for renting a VM instance in data center i is f_i per unit time. These charges follow the charging model of leading commercial cloud providers such as Amazon EC2 (**?**) and S3(**?**). We also suppose that each request is served by one unit bandwidth, and the number of requests a VM in data center i can serve per time slot is r_i.

Optimization variables. Given the above inputs, the decision variables in our model are formulated as follows:

(i) For *content replication*, we use binary variable $y_i^{(m)}(t)$ to indicate whether file m is stored in data center i in time slot t or not.

In each time slot, file m can be (a) migrated into data center i, *i.e.*, $y_i^{(m)}(t-1) = 0$ and $y_i^{(m)}(t) = 1$, (b) removed, *i.e.*, $y_i^{(m)}(t-1) = 1$ and $y_i^{(m)}(t) = 0$, or (c) keep its previous state, *i.e.*, $y_i^{(m)}(t-1) = y_i^{(m)}(t) = 0$ or $y_i^{(m)}(t-1) = y_i^{(m)}(t) = 1$. In case of migration, we assume that the video is always copied from the on-premise server to the destination data center.

(ii) For *dispatching of requests* for file m originated from region j, let $s_j^{(m)}(t)$ be the number of requests to be served by the on-premise server in time slot t, and $c_{ji}^{(m)}(t)$ denote the number dispatched to data center i in time slot t. Let μ^{max} be the maximum number of requests dispatched from each request queue to a data center in a time slot.

2. MIGRATING CONTENT DISTRIBUTION SERVICE TO GEO-DISTRIBUTED HYBRID CONTENT CLOUDS

Not all requests arising in one time slot are distributed in the same time slot, subject to capacity constraints. In particular, a queue $Q_j^{(m)}$ is maintained to buffer requests for file m generated from users in region j over time, $\forall j \in \mathcal{N}, \forall m \in \mathcal{M}$. The backlog size of queue $Q_j^{(m)}$ at time t, *i.e.*, the number of requests generated in region j for file m but not dispatched yet by t, is denoted by $Q_j^{(m)}(t)$. The update of the request queue size is given as the following queueing law (**?**):

$$Q_j^{(m)}(t+1) = \max[Q_j^{(m)}(t) - s_j^{(m)}(t) - \sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t), 0] + a_j^{(m)}(t). \tag{2.1}$$

Requests for file m can only be dispatched to data center i when i is storing the file, *i.e.*, $c_{ji}^{(m)}(t) > 0$ only if $y_i^{(m)}(t) = 1$. In case that file m is migrated into data center i in time slot t, we assume that data center i can serve this file to users in the same time slot. This assumption is reasonable in that though copying a file from the on-premise server to another data center takes time, replicating the file and uploading it to users can be implemented in parallel: after receiving a small portion of the file, a data center can already start to serve the received chunks of the file to users. We can assume that more bandwidth is reserved for replicating files to data centers at the on-premise server (this bandwidth is not counted in b), than that used at data centers to upload to users.

Service quality. The QoS experienced by users is evaluated by service response delays. From the time when a request is generated by a user to the time the user starts to receive the file, such delay consists of two major components: queueing delay in the respective request queue, and round-trip delay from when the request is dispatched from the queue to the time the first byte of the file is received. We ignore processing delays inside a data center, due to the high inter-connection bandwidth and CPU capacities inside a data center. Let d_j denote the round-trip delay between region j and the on-premise server, and e_{ji} be the round-trip delay between region j and data center i, reflecting proximity between the two regions. Let α be the upper-bound of average round-trip delay per request, which the application provider wishes to guarantee in this content distribution application. We reasonably assume $\alpha > e_{ii}, \forall i \in \mathcal{N}$, *i.e.*, this bound is larger than the round-trip delay between a user and the data center in the same region. We will show that our dynamic optimal service migration algorithm can bound both the average round-trip delay and queueing delay experienced by users.

Operational cost. Our algorithm focuses on minimizing recurring operational cost of

the content distribution system, not one-time costs such as the purchase of on-premise machines and contents. The recurring costs in each time slot t include the following categories:

i) Bandwidth charge at the on-premise server for uploading contents to users, at the total amount of $\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} v^{(m)}s_j^{(m)}(t)h$.

ii) Storage cost at data center $i, \forall i \in \mathcal{N}$, for caching replicated contents, at the total amount of
$\sum_{m\in\mathcal{M}} v^{(m)}y_i^{(m)}(t)p_i$.

iii) Request service cost at data center i for uploading replicated files to users. The cost for serving file m includes VM rental cost $\frac{\sum_{j\in\mathcal{N}} c_{ji}^{(m)}(t)}{r_i} \times f_i$ and upload bandwidth cost $\sum_{j\in\mathcal{N}} v^{(m)}c_{ji}^{(m)}(t)g_i$. Let $q_i^{(m)} = \frac{f_i}{r_i} + v^{(m)}g_i$ denote the unit cost to serve each request for file m on data center i. The total cost of serving requests at data center i is $\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} c_{ji}^{(m)}(t)q_i^{(m)}$.

iv) Migration cost for copying files from the on-premise server to data center i. Let $w_i^{(m)}$ denote the migration cost to copy file m into data center i, which includes costs of upload and download bandwidths from the on-premise server to data center i, *i.e.*, $w_i^{(m)} = v^{(m)}(h + o_i)$. The total migration cost incurred at data center i is $\sum_{m\in\mathcal{M}}[y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+ w_i^{(m)}$, where notation $[x]^+ = x$ if $x \geq 0$ and $[x]^+ = 0$ if $x < 0$.

We will not consider any recurring storage cost on the on-premise server (the purchase of storage disks by the application provider is an one-time investment). In addition, the removal of contents from a data center is cost-free.

Therefore, the overall operational cost to the application provider in time slot t is

$$\begin{aligned} M(t) \;=\; & \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} v^{(m)}s_j^{(m)}(t)h + \sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}} v^{(m)}y_i^{(m)}(t)p_i \\ & + \sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} c_{ji}^{(m)}(t)q_i^{(m)} + \sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}} [y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+ w_i^{(m)}. \end{aligned} \tag{2.2}$$

Optimization formulation. The optimization pursued by our dynamic algorithm is formulated as follows, which minimizes the time-averaged operational cost while guaranteeing service stability and quality. Let $\overline{x(t)} = \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} x(t)$ represent the time-averaged value of $x(t)$.

$$\min \overline{M(t)} \tag{2.3}$$

subject to:

$$\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} s_j^{(m)}(t) \le b, \forall t, \tag{2.4}$$

$$0 \le c_{ji}^{(m)}(t) \le \mu_{max} y_i^{(m)}(t), \forall j \in \mathcal{N}, \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t, \tag{2.5}$$

$$s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t) \le Q_j^{(m)}(t), \forall m \in \mathcal{M}, \forall j \in \mathcal{N}, \forall t, \tag{2.6}$$

$$\overline{a_j^{(m)}(t)} \le \overline{s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)}, \forall m \in \mathcal{M}, \forall j \in \mathcal{N}, \tag{2.7}$$

$$\overline{\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}} (s_j^{(m)}(t)d_j + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)e_{ji})} < \alpha \overline{\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}} (s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t))},$$

$$s_j^{(m)}(t) \ge 0, \forall j \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t, \tag{2.8}$$

$$y_i^{(m)}(t) \in \{0,1\}, \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t. \tag{2.9}$$

Recall that each request is served by a unit bandwidth. (2.4) are upload bandwidth constraints at the on-premise server. (2.5) states that requests for a file are only dispatched to data centers storing its content at the time, and in a time slot the max number of requests dispatched from each request queue to a data center is no larger than μ^{max}. (2.6) states that the number of requests dispatched from queue $Q_j^{(m)}$ cannot be larger than the current queue size. (2.7) guarantees the strong stability of queue $Q_j^{(m)}$, *i.e.*, the average number of backlogged requests in the queue at each time will not grow unbounded, based on the following theorem from (**?**):

Theorem 1 ((**?**)). *For a queue Q with the queuing law $Q(t+1) = Q(t) - b(t) + a(t)$, where $a(t)$ and $b(t)$ are the queue incoming rate and outgoing rate at time slot t, respectively. If the average incoming rate $\bar{a} = \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}(a(\tau))$ is strictly smaller than the average outgoing rate $\bar{b} = \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}(b(\tau))$,* i.e.*, $\bar{a} < \bar{b}$, then queue Q is strongly stable.*

In (2.8), $\Gamma_1 = \overline{\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t))}$ is the average number of overall requests in the system per unit time, and $\Gamma_2 = \overline{\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)d_j + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)e_{ji})}$ is the total round-trip delay experienced by requests in the system per unit time. Therefore, this constraint specifies that the average round-trip delay per request, $\frac{\Gamma_2}{\Gamma_1}$, should be bounded by α. Although we only model round-trip delay bound in the constraints, we will show in Chapter 2.3 that our algorithm to solve this optimization can guarantee a constant queueing delay bound in each request queue $Q_j^{(m)}$ as well.

2.2 Dynamic Migration Algorithm

In this chapter, we design a dynamic control algorithm using Lyapunov optimization techniques, which solves the optimal migration problem in (2.3), and discuss its practical implementation.

2.2.1 Introducing Virtual Queues

The optimization problem in (2.3) includes constraints on time-averaged variable values, *i.e.*, inequalities (2.7) and (2.8). Our dynamic algorithm will only be able to adjust variables in each time slot. How can we guarantee these inequalities by controlling the variable values over time?

Constraint (2.7) is satisfied if we guarantee the stability of each request queue Q_j^m, $\forall j \in \mathcal{N}, m \in \mathcal{M}$, according to Theorem 1. To satisfy constraint (2.8), we resort to the virtual queue techniques in Lyapunov optimization.

We introduce a virtual queue G, with the arrival rate of $\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)d_j + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)e_{ji})$, *i.e.*, the overall round-trip delay experienced by all requests in t, and departure rate of $\alpha\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t))$, *i.e.*, the total number of requests in t multiplied by the upper bound of round-trip delay per request (α). G is updated as follows:

$$G(t+1) = \max[G(t)+\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)d_j+\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)e_{ji})-\alpha\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(s_j^{(m)}(t)+\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)), 0]. \tag{2.10}$$

Based on Theorem 1, if queue G is stable, then its time-averaged arrival rate should not exceed its time-averaged departure rate, *i.e.*, constraint (2.8) is satisfied. Therefore, we can adjust load distribution strategies $s_j^{(m)}(t)$'s and $c_{ji}^{(m)}(t)$'s in each time slot t to guarantee that the virtual queue is always stable, in order to satisfy constraint (2.8). Intuitively, when the size of G is large, *i.e.*, a risk arises for constraint (2.8) to be violated, requests should be dispatched more to the on-premise server or data centers with small round-trip delay from users; when the queue size is small, requests can be distributed based more on cost considerations.

Recall that our dynamic migration algorithm also seeks to bound queueing delays in the request queues Q_j^m, $\forall j \in \mathcal{N}, m \in \mathcal{M}$. A technique, *$\epsilon$-persistent service* queue (**?**), can be applied, to bound the worst-case queueing delay of each request in a queue within a threshold. In particular, we associate each request queue $Q_j^{(m)}$ with a virtual queue $Z_j^{(m)}$, updated by:

$$Z_j^{(m)}(t+1) = \max[Z_j^{(m)}(t) + 1_{\{Q_j^{(m)}(t)>0\}}(\epsilon_j^{(m)} - s_j^{(m)}(t) - \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)) - 1_{\{Q_j^{(m)}(t)=0\}}\mu_{max}, 0], \tag{2.11}$$

where $\epsilon_j^{(m)} > 0$ is a pre-specified constant that can be gauged to control the queueing delay bound. Its value may further render a tradeoff between the queueing delay bound and the optimality of operational cost achieved by the dynamic algorithm. More discussions are included in Sec. 2.3. The rationale behind (2.11) can be explained intuitively: In each time slot t, if request queue $Q_j^{(m)}$ is not empty, then there is a constant rate of arrivals $\epsilon_j^{(m)}$ to virtual queue $Z_j^{(m)}$, making the length of the virtual queue grow (even though queue size of request queue $Q_j^{(m)}$ may not increase in t), and the departure rates from $Z_j^{(m)}$ and $Q_j^{(m)}$ are the same. If request queue $Q_j^{(m)}$ is empty, length of virtual queue $Z_j^{(m)}$ decreases by rate μ_{max}. The arrival of $\epsilon_j^{(m)}$ in each time slot in virtual queue $Z_j^{(m)}$ pressures the departure from request queue $Q_j^{(m)}$, *i.e.*, to guarantee stability of $Z_j^{(m)}$, request departure rates $s_j^{(m)}(t)$ and $c_{ji}^{(m)}(t)$ should be carefully decided, such that a proper length of the request queue is retained (and thus limited queueing delay per request results). Detailed analysis is in Theorem 4 in Sec. 2.3.

2.2.2 Dynamic Algorithm Design via Drift-Plus-Penalty Minimization Method

In summary, in our dynamic algorithm for the cost minimizing problem, three types of queues are needed, *i.e.*, $Q_j^{(m)}$ $(\forall j \in \mathcal{N}, \forall m \in \mathcal{M})$, G, and $Z_j^{(m)}$ $(\forall j \in \mathcal{N}, \forall m \in \mathcal{M})$. Let $\mathbf{\Theta}(t) = [\mathbf{Q}(t), \mathbf{G}(t), \mathbf{Z}(t)]$ be the vector of all queues in the system. Define our Lyapunov function as

$$L(\mathbf{\Theta}(t)) = \frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}(Q_j^{(m)}(t))^2 + \frac{1}{2}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}(Z_j^{(m)}(t))^2 + \frac{1}{2}(G(t))^2.$$

The one-slot conditional Lyapunov drift is

$$\Delta(\mathbf{\Theta}(t)) = \mathbb{E}\{L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t))|\mathbf{\Theta}(t)\}.$$

According to the *drift-plus-penalty* framework in Lyapunov optimization (**?**), an upper bound for the following expression should be minimized in each time slot, with the observation of the queue states $\mathbf{\Theta}(t)$, and data arrival rates $a_j^{(m)}(t), \forall j \in \mathcal{N}, \forall m \in \mathcal{M}$, such that an upper bound for $\overline{M(t)}$ is minimized (see Chapter 5 in (**?**)):

$$\Delta(\mathbf{\Theta}(t)) + VM(t).$$

Here, V is a non-negative parameter chosen by the application provider to control the tradeoff between operational cost and service response delays. Squaring the queueing laws (2.1), (2.10) and (2.11), we derive the following inequality :

$$\begin{aligned}
&\Delta(\mathbf{\Theta}(t)) + VM(t) \\
&\leq B + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} Q_j^{(m)}(t)[a_j^{(m)}(t) - s_j^{(m)}(t) - \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)] \\
&+ G(t)[\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)e_{ji} + s_j^{(m)}(t)d_j) - \alpha\sum_{j\in\mathcal{N}}\sum_{m\in\mathcal{M}}(\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t) + s_j^{(m)}(t))] \\
&+ \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} Z_j^{(m)}(t)[1_{\{Q_j^{(m)}(t)>0\}}(\epsilon_j^{(m)} - s_j^{(m)}(t) - \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)) - 1_{\{Q_j^{(m)}=0\}}\mu_{max}] \\
&+ V[\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)q_i^{(m)} + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} v^{(m)}s_j^{(m)}(t)h + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} v^{(m)}y_i^{(m)}(t)p_i + \sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{N}}[y_i^{(m)}(t) - y_i^{(m)}(t -
\end{aligned} \tag{2.12}$$

Rearranging the terms in the right-hand-side, we get:

$$\begin{aligned}
&\Delta(\mathbf{\Theta}(t)) + VM(t) \\
&\leq B - \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} s_j^{(m)}(t)[Q_j^{(m)}(t) + (\alpha - d_j)G(t) + 1_{\{Q_j^{(m)}(t)>0\}}Z_j^{(m)}(t) - v^{(m)}Vh] \\
&- \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)[Q_j^{(m)}(t) + (\alpha - e_{ji})G(t) + 1_{\{Q_j^{(m)}(t)>0\}}Z_j^{(m)}(t) - Vq_i^{(m)}] \\
&+ V\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{N}}[v^{(m)}y_i^{(m)}(t)p_i + [y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+ w_i^{(m)}] \\
&+ \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} Q_j^{(m)}(t)a_j^{(m)}(t) + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} Z_j^{(m)}(t)[1_{\{Q_j^{(m)}(t)>0\}}\epsilon_j^{(m)} - 1_{\{Q_j^{(m)}=0\}}\mu_{max}],
\end{aligned} \tag{2.13}$$

where $B = \frac{1}{2}|\mathcal{M}||\mathcal{N}|[A_{max}^2 + \epsilon_{max}^2 + 2(b + N\mu_{max})^2] + + \frac{1}{2}(|\mathcal{M}||\mathcal{N}|^2\mu_{max}e_{max} + bd_{max})^2 + \frac{1}{2}\alpha^2(|\mathcal{M}||\mathcal{N}|^2\mu_{max} + b)^2$ is a constant value, where $d_{max} = \max\{d_j | j \in \mathcal{N}\}, e_{max} = \max\{e_{ji} | j \in \mathcal{N}, i \in \mathcal{N}\}, \epsilon_{max} = \max\{\epsilon_j^{(m)} | j \in \mathcal{N}, m \in \mathcal{M}\}$.

Our algorithm seeks to minimize the right-hand-side of inequality (2.13), in order to minimize the upper bound for $\Delta(\mathbf{\Theta}(t)) + VM(t)$, and thus the upper bound of $\overline{M(t)}$. To minimize the right-hand-side of inequality (2.13), the algorithm observes the queues $Q_j^{(m)}(t)$, $G(t)$ and $Z_j^{(m)}(t)$ in each time slot t, and decides optimal values of $s_j^{(m)}(t)$ and $c_{ji}^{(m)}(t)$, $\forall j \in \mathcal{N}, i \in \mathcal{N}, m \in \mathcal{M}$.

We simplify the notation by defining

$$\gamma_j^{(m)}(t) = Q_j^{(m)}(t) + 1_{\{Q_j^{(m)}(t)>0\}}Z_j^{(m)}(t) - Vv^{(m)}h + (\alpha - d_j)G(t)$$

which is a constant in time slot t, and

$$\eta_{ji}^{(m)}(t) = Q_j^{(m)}(t) + 1_{\{Q_j^{(m)}(t)>0\}} Z_j^{(m)}(t) - Vq_i^{(m)} + (\alpha - e_{ji})G(t)$$

which is also a constant at t. In addition, we notice that when $y_i^{(m)}(t-1) = 0$, $[y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+ = y_i^{(m)}(t)$, and when $y_i^{(m)}(t-1) = 1$, $[y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+ = 0$. Therefore, we can simplify $V \sum_{m\in\mathcal{M}} \sum_{i\in\mathcal{N}} [v^{(m)} y_i^{(m)}(t) p_i + [y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+ w_i^{(m)}]$ to

$$\sum_{m\in\mathcal{M}} \sum_{i\in\mathcal{N}} \phi_i^{(m)}(t) y_i^{(m)}(t),$$

where

$$\phi_i^{(m)}(t) = \begin{cases} Vv^{(m)}p_i, & if \quad y_i^{(m)}(t-1) = 1, \\ V(v^{(m)}p_i + w_i^{(m)}), & if \quad y_i^{(m)}(t-1) = 0, \end{cases}$$

$\forall i \in \mathcal{N}, \forall m \in \mathcal{M}$. Given $y_i^{(m)}(t-1)$, $\phi_i^{(m)}(t)$ is a constant in time slot t.

Therefore, minimizing the right-hand-side of (2.13) is equivalent to:

$$\max \quad F(t) = \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{N}} s_j^{(m)}(t)\gamma_j^{(m)}(t) + \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{N}} \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)\eta_{ji}^{(m)}(t) - \sum_{m\in\mathcal{M}} \sum_{i\in\mathcal{N}} \phi_i^{(m)}(t) y_i^{(m)}(t) \tag{2.14}$$

$$\text{subject to: constraints (2.4) (2.6) (2.5) (2.8) (2.9).}$$

This problem is an integer linear program (ILP). For efficient and timely solutions, we design a simple heuristic in Algorithm 1, which turns out to be fast and approaches the optimum well, based on our evaluations in Sec. 2.4.

The idea of the heuristic is as follows. In time slot t, given the content placement solutions in $t-1$, *i.e.*, $y_i^{(m)}(t-1), \forall i \in \mathcal{N}, m \in \mathcal{M}$, we probe to see if altering each placement decision $y_i^{(m)}$ from 0 (or 1) in $t-1$ to 1 (or 0) in t will lead to increase of the objective function in (2.14). Such probing is conducted in an iterative fashion for at most H rounds. In each round, we further iterate through each $y_i^{(m)}$: we set $y_i^{(m)}(t) = 1 - y_i^{(m)}(t-1)$, while retaining all other placement decisions as those resumed in $t-1$; then based on the changed content placement situation $y_i^{(m)}(t), \forall i \in \mathcal{N}, m \in \mathcal{M}$, we solve optimization (2.14) for the optimal load distribution strategies, $s_j^{(m)}(t)$ and $c_{ji}^{(m)}(t)$, $\forall j, i \in \mathcal{N}, m \in \mathcal{M}$; we then find out the difference between the new objective function value in t and the previous value in $t-1$, which could be an increase or a decrease. In each round, $y_i^{(m)}$ whose change from $t-1$ to t yields the largest increase

in objective function value is identified, and we will indeed make $1 - y_i^{(m)}(t-1)$ the value for $y_i^{(m)}(t)$ when this round ends, but will keep other placement decisions intact.

Algorithm 1 iterates for at most H rounds in each execution, where H is a parameter set by the application provider to control the tradeoff between running time and optimality of the output.

In the above steps, we need to solve optimization (2.14) for the optimal load distribution strategies, with given content placement situation $y_i^{(m)}(t)$'s. With fixed $y_i^{(m)}(t)$'s, the problem (2.14) an be solved using an efficient heuristic:

Associate variable $c_{ji}^{(m)}$ with weight $\eta_{ji}^{(m)}$ and $s_j^{(m)}$ with weight $\gamma_j^{(m)}$, respectively, $\forall j, i \in \mathcal{N}, m \in \mathcal{M}$. If the associated weight is non-positive (*i.e.*, the request queue $Q_j^{(m)}$ is not long enough, the queueing delay of the requests in $Q_j^{(m)}$ is not large, or the corresponding round trip delay is too large), the value of the variable is set to 0 (*i.e.*, no request for content m will be dispatched to region i from j). Sort all remaining variables (with positive weights) in decreasing order of their weights. Starting with the variable with the largest weight, we allocate the largest possible value to the variables in the ordered list, as long as constraints (2.4)(2.6)(2.5) on request handling capacities of the on-premise server and data centers are not violated.

2.2.3 Discussions on Practical Implementation

Our dynamic algorithm is to be deployed by the application provider to optimally distribute their content distribution service onto the hybrid cloud. The application provider deploys one or multiple web servers providing portal service of the content distribution application, in a centralized or distributed fashion. The portal aggregates user requests and sends collection request information to a *control center*, which executes our algorithm periodically.

The control center maintains a content placement table with entries $y_i^{(m)}, \forall j \in \mathcal{N}, m \in \mathcal{M}$, indicating whether file m is currently replicated on data center i. The entries are initialized to be 0 at the system initialization stage. In each time slot, received requests for file m originated from region j are placed in request queue $Q_j^{(m)}$. Virtual queues $Z_j^{(m)}$ and G are maintained simply as counters. The control center observes lengths of the queues and request arrival rates, and calculates the optimal content placement and load distribution strategies using Algorithm 1.

Algorithm 1: Heuristic to Solve Optimization (2.14)

Input: $\gamma_j^{(m)}(t), \eta_j^{(m)}(t), \phi_i^{(m)}(t), Q_j^{(m)}(t), Z_j^{(m)}(t), G(t)$, $\forall i, j \in \mathcal{N}, m \in \mathcal{M}$

Output: $s_j^{(m)}(t), c_{ji}^{(m)}(t), y_i^{(m)}(t), \forall i, j \in \mathcal{N}, m \in \mathcal{M}$

set $y_i^{(m)}(t) = y_i^{(m)}(t-1), \forall i \in \mathcal{N}, m \in \mathcal{M}$;
$c_{max} = F^*(t-1)$ as the objective function value in last time slot;
for $round \leftarrow 1$ **to** H **do**
    $i_{max} = -1, m_{max} = -1$;
    foreach $m \in \mathcal{M}, i \in \mathcal{N}$ **do**
        set $y_i^{(m)}(t) = 1 - y_i^{(m)}(t-1)$;
        calculate $F^*(t)$ as the objective function value by solving optimization (2.14) using the greedy heuristic, with the above $y_j^{(m)}(t)$, $\forall j \in \mathcal{N}$, $m \in \mathcal{M}$, given ;
        recover $y_i^{(m)}(t) = 1 - y_i^{(m)}(t-1)$;
        if $F^*(t) > c_{max}$ **then**
            % record the best $y_i^{(m)}$%;
            $c_{max} = F^*(t)$;
            $i_{max} = i$;
            $m_{max} = m$;
    if $i_{max} == -1$ **then**
        % optimal solution achieve %;
        Break ;
    $y_{i_{max}}^{(m_{max})}(t) = 1 - y_{i_{max}}^{(m_{max})}(t-1)$;
output $y_i^{(m)}(t), \forall i \in \mathcal{N}, m \in \mathcal{M}$;
output $s_j^{(m)}(t), c_{ji}^{(m)}(t), \forall i, j \in \mathcal{N}, m \in \mathcal{M}$ by calling the greedy heuristic based on the fixed $y_i^{(m)}(t), \forall i \in \mathcal{N}, m \in \mathcal{M}$;

Based on the derived content placement strategies, it updates the placement table, and compares the optimal solution $y_i^{(m)}(t)$ against the current value of $y_i^{(m)}(t-1)$ in the table, $\forall j \in \mathcal{N}, m \in \mathcal{M}$: If $y_i^{(m)}(t-1) = 0$ and $y_i^{(m)}(t) = 1$, the control center instructs the on-premise server to send a copy of file m to data center i; if $y_i^{(m)}(t-1) = 1$ and $y_i^{(m)}(t) = 0$, it signals data center i to remove file m from its storage.

Based on the request distribution decisions, the control center dispatches $s_j^{(m)}(t)$ requests from queue $Q_j^{(m)}$ to the on-premise server, and $c_{ji}^{(m)}(t)$ requests from the queue $Q_j^{(m)}$ to data center i, $\forall j, i \in \mathcal{N}, m \in \mathcal{M}$. Virtual queue $Z_j^{(m)}$ and G are updated accordingly.

The sketch of our complete dynamic, joint content placement and load distribution algorithm is presented in Algorithm 2.

As an engineering parameter, the length of intervals between two executions of

Algorithm 2: Dynamic Control Algorithm on the Control Center

Initialization:
Set up request queue $Q_j^{(m)}$, virtual queues G and $Z_j^{(m)}, \forall j \in \mathcal{N}, m \in \mathcal{M}$, and initialize their backlogs to 0;

In every time slot t:

1. Enqueue received requests to request queues ($Q_j^{(m)}$'s);
2. Call Algorithm 1 to obtain optimal content placement and load distribution strategies $c_{ji}^{(m)}(t)$, $s_j^{(m)}(t)$, $y_i^{(m)}(t)$, $\forall j, i \in \mathcal{N}, m \in \mathcal{M}$;
3. Update content placement table with $y_i^{(m)}(t)$'s, and migrate files as follows:
for $i \in \mathcal{N}, m \in \mathcal{M}$ **do**
    if $y_j^{(m)}(t-1) = 0$ *and* $y_j^{(m)}(t) = 1$ **then**
        instruct on-premise server to upload file m to data center i;
    if $y_j^{(m)}(t-1) = 1$ *and* $y_j^{(m)}(t) = 0$ **then**
        signal data center i to remove file m;
4. Dispatch $s_j^{(m)}(t)$ requests from queue $Q_j^{(m)}$ to on-premise server, $c_{ji}^{(m)}(t)$ requests to data center i, $\forall j, i \in \mathcal{N}, m \in \mathcal{M}$;
5. Update virtual queue $Z_{ji}^{(m)}$ and G according to Eqn. (2.11) and (2.10);

the algorithm can be set by the application provider, based on update frequencies of contents, sizes of the files, as well as their targeted system performance optimality.

2.3 Performance Analysis

We next analyze the performance guarantee provided by our dynamic algorithm, with respect to bounded queueing delay and optimality in cost minimization.

2.3.1 Bound of Request Queue Length

Lemma 2. *At any given time t, for some $j \in \mathcal{N}$, $i \in \mathcal{N}$, and $m \in \mathcal{M}$, if $Q_j^{(m)}(t) + 1_{\{Q_j^{(m)}(t)>0\}} Z_j^{(m)}(t) + (\alpha - e_{ji})G(t) > V(v^{(m)} p_i + w_i^{(m)} + q_i^{(m)})$, then $s_j^{(m)}(t) + \sum_{i \in \mathcal{N}} c_{ji}^{(m)}(t) = Q_j^{(m)}(t)$ or $c_{ji}^{(m)}(t) = \mu_{max}$.*

This lemma tells us that if the request queue $Q_j^{(m)}(t)$ or its corresponding virtual queue $Z_j^{(m)}(t)$ is long enough in time slot t, this request queue will be cleared in t, *i.e.*, the number of departures from the queue equals the current queue length, or be processed to the point that the request handling capacity of data center i is saturated ($c_{ji}^{(m)}(t) = u_{max}$), where the round-trip delay between data center i and region j is smaller than the preset upper bound. The detailed proof can be found in Appendix

2.6.1. Based on Lemma 2, we can prove the following theorem on the length of request queues.

Theorem 3. *(Bound of Queue Length) Define*

$$Q_j^{(m)^{max}} = V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)}) + A_{max} \tag{2.15}$$

where

$$i = argmin_{\tilde{i}}\{v^{(m)}p_{\tilde{i}} + w_{\tilde{i}}^{(m)} + q_{\tilde{i}}^{(m)} | \alpha - e_{j\tilde{i}} > 0, \forall \tilde{i} \in \mathcal{N}\}. \tag{2.16}$$

Then $Q_j^{(m)^{max}}$ *is the maximum size of queue* $Q_j^{(m)}$ *at any time* t, i.e., $Q_j^{(m)}(t) \leq Q_j^{(m)^{max}}$, $\forall j \in \mathcal{N}, \forall m \in \mathcal{M}$.

The proof is given in Appendix 2.6.2.

2.3.2 Bound of Queueing Delay

The following theorem shows that the queueing delay experienced by each request in each request queue is bounded.

Theorem 4. *(Bounded Queueing Delay): For each request queue* $Q_j^{(m)}$, $\forall j \in \mathcal{N}, \forall m \in \mathcal{M}$, *define*

$$W_j^{(m)} = \lceil \frac{V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)}) + Q_j^{(m)^{max}}}{\epsilon_m} \rceil$$

where i *is defined as in (2.16). The queueing delay of each request in* $Q_j^{(m)}$ *is bounded by* $W_j^{(m)}$.

The proof in Appendix 2.6.3 is based on the following idea: in the worst case, there are $Q_j^{(m)^{max}}$ requests in queue $Q_j^{(m)}$; in the subsequent several time slots, there are no requests arriving at $Q_j^{(m)}$. It takes at most $\lceil \frac{V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)})}{\epsilon_m} \rceil$ time slots for $Z_j^{(m)}$ to become saturated and starts to "press" the requests in $Q_j^{(m)}$ to departures. After that, it takes at most $\lceil \frac{Q_j^{(m)^{max}}}{\epsilon_m} \rceil$ time slots for $Q_j^{(m)}$ to be cleared.

2.3.3 Optimality against the T-Slot Lookahead Mechanism

Since request arrival rates are arbitrary in our system, it is difficult to find the global cost optimum, against which the time-averaged cost $\overline{M(t)}$ achieved by our algorithm can compare. Therefore, we utilize a local optimum target, which is the optimal (objective function) value of a similar cost minimization problem within known information (*e.g.*,

request arrivals) for T time slots into the future, *i.e.*, a *T-slot lookahead mechanism* (**?**). We will show that the optimal value obtained by our algorithm is close to that of the T-slot lookahead mechanism, even if our algorithm does not assume any future information.

In the T-slot lookahead mechanism, time is divided into successive frames, each consisting of T time slots. Denote each frame as $F_k = \{kT, kT+1, \ldots, kT+T-1\}$, where $k = 0, 1, \ldots$. In each time frame, consider the following optimization problem over variables $c_{ji}^{(m)}(t), s_j^{(m)}(t), y_i^{(m)}(t)$, $\forall j, i \in \mathcal{N}, m \in \mathcal{M}$, $t \in F_k$:

$$\min \frac{1}{T} \sum_{t=kT}^{kT+T-1} M(t) \tag{2.17}$$

subject to:

$$\sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{N}} s_j^{(m)}(t) \le b, \forall t \in F_k,$$

$$0 \le c_{ji}^{(m)}(t) \le \mu_{max} y_i^{(m)}(t), \forall j \in \mathcal{N}, \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in F_k,$$

$$s_j^{(m)}(t) + \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t) \le Q_j^{(m)}(t), \forall m \in \mathcal{M}, \forall j \in \mathcal{N}, \forall t \in F_k,$$

$$\sum_{t=kT}^{kT+T-1} [a_j^{(m)}(t) - s_j^{(m)}(t) - \sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t)] \le 0, \forall j \in \mathcal{N}, \forall m \in \mathcal{M},$$

$$\sum_{t=kT}^{kT+T-1} \sum_{j\in\mathcal{N}} \sum_{m\in\mathcal{M}} (\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t) e_{ji} + s_j^{(m)}(t) d_j) < \alpha \sum_{t=kT}^{kT+T-1} \sum_{j\in\mathcal{N}} \sum_{m\in\mathcal{M}} (\sum_{i\in\mathcal{N}} c_{ji}^{(m)}(t) + s_j^{(m)}(t)),$$

$$s_j^{(m)}(t) \ge 0, \forall j \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in F_k,$$

$$y_i^{(m)}(t) \in \{0, 1\}, \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in F_k.$$

Assuming full knowledge of request arrivals in the T time slots in F_k, this optimization derives the optimal content placement and load distribution decisions in each of the T time slots, which minimize the average cost per time slot in the objective function. We show the time-averaged cost $\overline{M(t)}$ achieved by our algorithm is within a constant gap $\frac{BT}{V}$ from that achieved by solving the above optimization:

Theorem 5. *(Optimality of Cost Minimization) Let $\widehat{M_k}$ denote the optimal objective function value in the T-slot Lookahead problem (2.17) in time frame F_k. The minimum operational cost derived with our Algorithm 2 is $M(t)$ in time slot t. Suppose the time lasts for KT time slots, where K is a constant. We have*

$$\frac{1}{KT} \sum_{t=0}^{KT-1} M(t) \le \frac{1}{K} \sum_{k=0}^{K-1} \widehat{M_k} + \frac{BT}{V}, \tag{2.18}$$

i.e., *our algorithm achieves a time-averaged cost within constant gap* $\frac{BT}{V}$ *from that by assuming full knowledge in* T *slots in the future.*

Theorem 5 is proved in details in Appendix 2.6.4.

Theorems 4 and 5 show that when V increases, worst-case queueing delay $W_j^{(m)}$ increases, while the gap between the operational cost of our dynamic algorithm and that of the T-Slot lookahead mechanism shrinks. $\epsilon_j^{(m)}$ has a similar effect: When $\epsilon_j^{(m)}$ increases, the worst-case queueing delay $W_j^{(m)}$ decreases, and B increases such that the gap to optimality increases. We will investigate proper values to assign to these tradeoff control parameters in our simulations in Sec. 2.4.

2.4 Empirical studies

We evaluate the performance of the dynamic algorithm with discrete-event simulations under realistic setting. There are 50 regions (50 data centers) and 1000 different files. The duration of a time slot is 10 seconds. Request arrivals in each region follow a Poisson process. Average request arrival rate is the multiplication of the popularity of the file and the population of the region. The popularity of files follows a Power-law distribution with mean of 10^{-6} accesses per person per time slot. The population in regions follows a uniform distribution within range $[0.5 \times 10^6, 1.5 \times 10^6]$ (persons). The maximum number of request arrivals per time slot is 12 and the maximum number of requests dispatched from a queue to a data center per time slot, i.e., μ_{max}, is 24. The size of a file follows a uniform distribution within range $[0.5 \times 10^6, 1.5 \times 10^6]$ (bytes). The number of concurrent requests that the on-premise server can serve in a time slot is 500.

The charge by the public cloud service is extracted from real settings (**?**). The cost of renting a VM is \$0.7 per hour. The number of requests a VM can process concurrently follows a uniform distribution within range $[40, 60]$. The cost of uploading from the on-premise server is $\$1.2 \times 10^{-10}$ per byte, and the cost of uploading from a data center follows uniform distribution within range $[\$0.96 \times 10^{-10}, \$1.44 \times 10^{-10}]$ (per byte). According to the current business model (**?**), there is no charge for downloading data to the cloud, i.e., $o_i = 0, \forall i \in \mathcal{N}$.

We denote the mean of the parameter set $\{\epsilon_j^{(m)} | j \in \mathcal{N}, m \in \mathcal{M}\}$ as

$$\overline{\epsilon} = \frac{1}{|\mathcal{N}||\mathcal{M}|} \sum_{j \in \mathcal{N}, m \in \mathcal{M}} \epsilon_j^{(m)}$$

The default value of $\overline{\epsilon}$ is 1. $\epsilon_j^{(m)}$ is proportional to $V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)}) + {Q_j^{(m)}}^{max}$ where i and ${Q_j^{(m)}}^{max}$ is defined in (2.16) and (2.15) respectively, such that the worst-case delay of each queue would be similar to each other. The round-trip delay between users in a region and the on-premise server, or between users in a region and a data center, follows a uniform distribution within range $[0.005, 0.025]$ (seconds). Especially, the round-trip delay between a user and the data center in the same region is the smallest, i.e., $e_{jj} = 0.005$ seconds for all j. The average round-trip delay target set by the application provider is 0.015 seconds.

2.4.1 Cost with Heuristic 1 and Optimal Solution to (2.14)

We first verify the performance of our heuristic in Algorithm 1 to solve optimization (2.14), against the precise optimal solution to (2.14). To derive the precise optimal solution, we use an open source tool *GLPK* (**?**). Fig. 2.2 shows the overall cost incurred at each time slot when using each method to solve (2.14). We can see that our heuristic performs very well with costs within a gap of 3% to that achieved by the precise optimal solution. During our experiments, we have also observed that the heuristic algorithm runs much faster than the *GLPK* solver. This proves that our heuristic algorithm is more suitable to deploy in realistic environments.

2.4.2 Impact of Algorithm Parameters

2.4.2.1 Exploring V

Fig. 2.3 shows that when V increases, the overall operational cost becomes smaller. Fig. 2.4 reveals that the average service response delay per request (queueing delay+round-trip delay) increases with V's increase, while Fig. 2.5 shows the increase of the average of maximum request queue lengths (i.e., the average of the maximum lengths that each request queue has ever reached until each time slot) with V's increase as well. These figures clearly show a tradeoff in V's setting. Selecting $V = [10000, 50000]$ would be good trade-off between cost optimality and service quality.

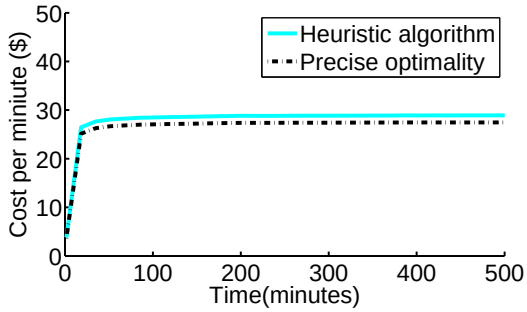

Figure 2.2: Cost comparison with our heuristic and precise optimal solution.

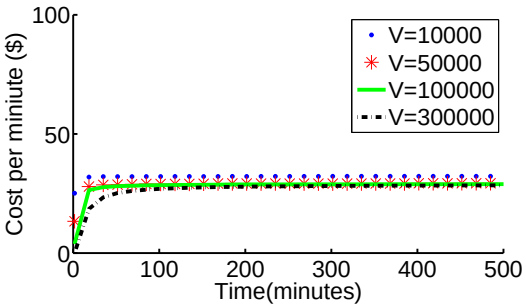

Figure 2.3: Cost with different V.

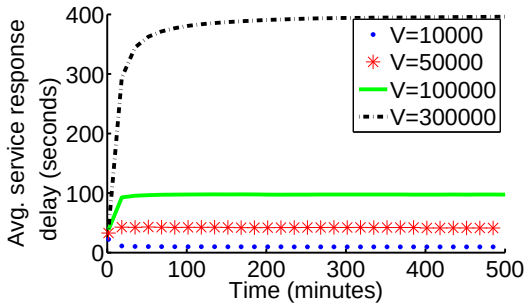

Figure 2.4: Average service response delay with different V.

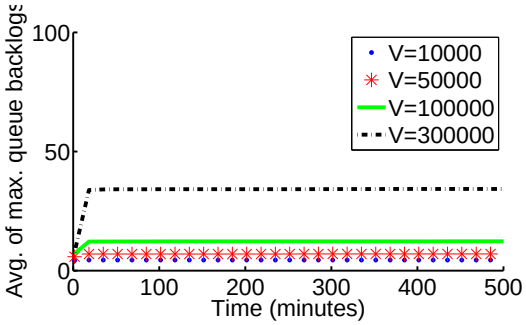

Figure 2.5: Average of max. queue lengths with different V.

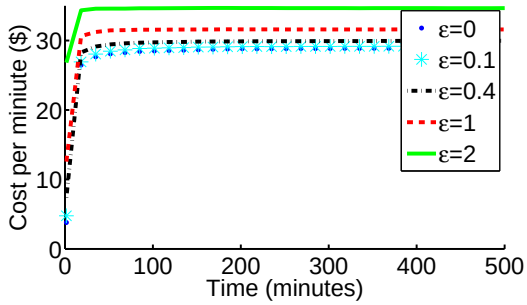

Figure 2.6: Cost with different $\bar{\epsilon}$.

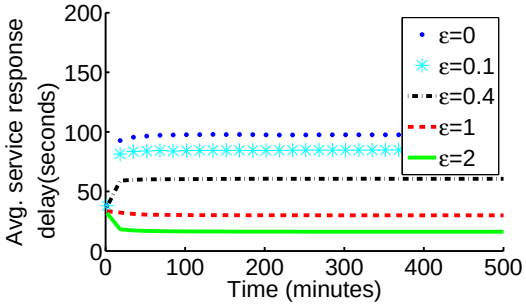

Figure 2.7: Average service response delay with different $\bar{\epsilon}$.

2.4.2.2 Exploring $\epsilon_j^{(m)}$

In Fig. 2.6 and 2.7, we observe that when $\bar{\epsilon}$ increases, the overall operational cost increases while the service response delay per request decreases. This shows that $\bar{\epsilon}$ also renders a tradeoff between cost optimality and service qualities. When $\bar{\epsilon}$ is larger than 1 the marginal decrease of average service response delay is small while the cost keeps increasing. Therefore $\bar{\epsilon} = [0.8, 1.2]$ would be a good option.

2.5 Summary

In this chapter, we study the problem of migration of a content distribution service to a hybrid cloud consisting of private on-premise servers and public geo-distributed cloud services. We propose a generic optimization framework based on Lyapunov optimization theory, to design a dynamic, joint content placement and load distribution algorithm, which minimizes the operational cost of application provider under the constraints of QoS. By rigorous theoretical analysis, we show that our algorithm approaches the optimality achieved by a mechanism with known information in the future T slots by a small constant gap, no matter what the request arrival pattern is.

2.6 Proofs

2.6.1 Proof of Lemma 2

Proof: In the following, j, i, m are fixed values unless stated otherwise.

(1) If $Q_j^{(m)}(t) = 0$, we have $s_j^{(m)}(t) + \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t) = 0 = Q_j^{(m)}(t)$.

(2) If $Q_j^{(m)}(t) \neq 0$, we prove by contradiction by assuming that $s_j^{(m)}(t)+\sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t) < Q_j^{(m)}(t)$ and $c_{ji}^{(m)}(t) < u_{max}$.

We first denote all the terms in $F(t)$ that contain the specific $c_{ji}^{(m)}(t)$ and $y_i^{(m)}(t)$ as

$$P(y_i^{(m)}(t), c_{ji}^{(m)}(t)) = c_{ji}^{(m)}(t)[Q_j^{(m)}(t) + 1_{\{Q_j^{(m)}(t)>0\}} Z_j^{(m)}(t) - Vq_i^{(m)} + (\alpha - e_{ji})G(t)] - V[v^{(m)} y_i^{(m)}(t) p_i + [y_i^{(m)}(t) - y_i^{(m)}(t-1)]^+ w_i^{(m)}].$$

We also denote the remaining terms in $F(t)$ excluding $P(y_i^{(m)}(t), c_{ji}^{(m)}(t))$ as

$$L(y_i^{(m)}(t), c_{ji}^{(m)}(t)) = F(y_i^{(m)}(t), c_{ji}^{(m)}(t)) - P(y_i^{(m)}(t), c_{ji}^{(m)}(t)).$$

First, we show that setting $y_i^{(m)}(t) = 0$ is no better than setting $y_i^{(m)}(t) = 1$, as follows:

When $y_i^{(m)}(t) = 0$, it must be $c_{ji}^{(m)}(t) = 0$ because of (2.5). So $P(y_i^{(m)}(t) = 0, c_{ji}^{(m)}(t) = 0) = 0$. When $y_i^{(m)}(t) = 1$, because $s_j^{(m)}(t) + \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t) < Q_j^{(m)}(t)$, we can set $c_{ji}^{(m)}(t)$ to be at least 1. Then, $P(y_i^{(m)}(t) = 1, c_{ji}^{(m)}(t) = 1) > 0 = P(y_i^{(m)}(t) = 0, c_{ji}^{(m)}(t) = 0)$. But $L(y_i^{(m)}(t) = 1, c_{ji}^{(m)}(t) = 1)$ should not be smaller than $L(y_i^{(m)}(t) = 0, c_{ji}^{(m)}(t) = 0)$. Therefore $F(y_i^{(m)}(t) = 1, c_{ji}^{(m)}(t) = 1) \geq F(y_i^{(m)}(t) = 0, c_{ji}^{(m)}(t) = 0)$.

Second, when $y_i^{(m)}(t) = 1$, there is room to increase $c_{ji}^{(m)}(t)$ without violating the constraints. We can increase $F(t)$ by increasing $c_{ji}^{(m)}(t)$ because the coefficient of $c_{ji}^{(m)}(t)$, *i.e.*, $Q_j^{(m)}(t) + 1_{\{Q_j^{(m)}(t)>0\}} Z_j^{(m)}(t) - V q_i^{(m)} + (\alpha - e_{ji})G(t)$, is larger than 0 according to our assumption. Therefore, the solution in which $s_j^{(m)}(t) + \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t) < Q_j^{(m)}(t)$ and $c_{ji}^{(m)}(t) < u_{max}$ is not optimal. Contradiction has occurred. □

2.6.2 Proof of Theorem 3

Proof: We prove the theorem by induction.

Induction Basis: According to the assumption of our model, we have $Q_j^{(m)}(0) = 0 \leq {Q_j^{(m)}}^{max}, \forall j \in \mathcal{N}, m \in \mathcal{M}$.

Induction steps: We assume that $Q_j^{(m)}(t) \leq {Q_j^{(m)}}^{max}$ and then we show that $Q_j^{(m)}(t+1) \leq {Q_j^{(m)}}^{max}$.

If $Q_j^{(m)}(t) \leq V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)})$, then $Q_j^{(m)}(t+1) \leq V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)}) + A_{max} = {Q_j^{(m)}}^{max}$.

If $Q_j^{(m)}(t) > V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)})$, then according to Lemma 2, it would be one of the following two cases, which we are going to discuss respectively.

Case (1) — $s_j^{(m)}(t) + \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t) = Q_j^{(m)}(t)$:
$Q_j^{(m)}(t+1) = \max[Q_j^{(m)}(t) - s_j^{(m)}(t) - \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t), 0] + a_j^{(m)}(t) = a_j^{(m)}(t) \leq A_{max} < {Q_j^{(m)}}^{max}$.

Case (2) — $s_j^{(m)}(t) + \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t) < Q_j^{(m)}(t)$ and $c_{ji}^{(m)}(t) = u_{max}$:
$Q_j^{(m)}(t+1) - Q_j^{(m)}(t) = \max[Q_j^{(m)}(t) - s_j^{(m)}(t) - \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t), 0] + a_j^{(m)} - Q_j^{(m)}(t) = -s_j^{(m)}(t) - \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t) + a_j^{(m)} \leq -u_{max} + A_{max} < 0$. This means that the length of $Q_j^{(m)}$ can not increase in time slot $t+1$. Therefore, $Q_j^{(m)}(t+1) < {Q_j^{(m)}}^{max}$. □

2.6.3 Proof of Theorem 4

Proof: Consider the requests in $Q_j^{(m)}$ at time slot t_0. According to Theorem 3, $Q_j^{(m)}(t_0) \leq {Q_j^{(m)}}^{max}$. We only need to show that in the following $W_j^{(m)}$ time slots, at least $Q_j^{(m)}(t_0)$ requests will depart from queue $Q_j^{(m)}$.

In the following $W_j^{(m)}$ time slots, if there exists $s_j^{(m)}(t) + \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t) = Q_j^{(m)}(t)$ for some t ($Q_j^{(m)}$ is cleared) , then the queueing delay is within $W_j^{(m)}$.

Otherwise, in the $W_j^{(m)}$ time slots, it remains that $Q_j^{(m)}(t) > 0$. So, $Z_j^{(m)}(t)$ keeps increasing by $\epsilon_j^{(m)}$ each time slot. After $\lceil \frac{V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)})}{\epsilon_j^{(m)}} \rceil$ time slots, $Z_j^{(m)}(t) > V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)})$. Because $\alpha - e_{ji} > 0$, $G(t) \geq 0$, $Q_j^{(m)}(t) \geq 0$, and $Q_j^{(m)}(t) + 1_{\{Q_j^{(m)}(t)>0\}} Z_j^{(m)} + (\alpha - e_{ji})G(t) > V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)})$, then according to Lemma 2, we have $s_j^{(m)}(t) + \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t) = Q_j^{(m)}(t)$ or $c_{ji}^{(m)}(t) = \mu_{max}$. Because we have excluded the case $s_j^{(m)}(t) + \sum_{\hat{i}\in\mathcal{N}} c_{j\hat{i}}^{(m)}(t) = Q_j^{(m)}(t)$, we have $c_{ji}^{(m)}(t) = \mu_{max}$, and $Z_j^{(m)}(t)$ is decreased by μ_{max}. According to the queueing laws (2.1) and (2.11), $Q_j^{(m)}$ has μ_{max} departures as $Z_j^{(m)}$ does. After every $\lceil \frac{\mu_{max}}{\epsilon_j^{(m)}} \rceil$ time slots, $Z_j^{(m)}(t)$ goes above $V(v^{(m)}p_i + w_i^{(m)} + q_i^{(m)})$ again, so that μ_{max} requests depart from $Z_j^{(m)}$ and $Q_j^{(m)}$ again. In total, after $\lceil \frac{Q_j^{(m)^{max}}}{\mu_{max}} \times \frac{\mu_{max}}{\epsilon_j^{(m)}} \rceil = \lceil \frac{Q_j^{(m)^{max}}}{\epsilon_j^{(m)}} \rceil$ time slots, $Q_j^{(m)^{max}}$ requests are processed.

Therefore, any request arrived at time t_0 will be processed before time slot $t_0 + W_j^{(m)}$.

□

2.6.4 Proof of Theorem 5

Define *T-slot Drift* as

$$\Delta_T(\Theta(t)) = L(\Theta(t+T)) - L(\Theta(t)).$$

Based on Lemma 4.11 in (**?**) , we have

Lemma 6. *(T-slot Drift) With our dynamic algorithm, for all t, all $\Theta(t)$, and for any integer $T > 0$ we have:*

$$
\begin{aligned}
&\Delta_T(\Theta(t)) + V \sum_{\tau=t}^{t+T-1} M(\tau) \\
&\leq BT^2 + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} Q_j^{(m)}(t) \sum_{\tau=t}^{t+T-1} (a_j^{(m)}(\tau) - {s_j^{(m)}}^*(\tau) - \sum_i {c_{ji}^{(m)}}^*(\tau)) \\
&\quad + G(t)(\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}} \sum_{\tau=t}^{t+T-1} ({c_{ji}^{(m)}}^*(\tau) e_{ji} + {s_j^{(m)}}^*(\tau) d_j) - \alpha \sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}}\sum_{m\in\mathcal{M}} ({c_{ji}^{(m)}}^*(\tau) + {s_j^{(m)}}^*(\tau))) \\
&\quad + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} Z_j^{(m)}(t) \sum_{\tau=t}^{t+T-1} (1_{\{Q_j^{(m)}(\tau)>0\}}(\epsilon_j^{(m)} - {s_j^{(m)}}^*(\tau) - \sum_{i\in\mathcal{N}} {c_{ji}^{(m)}}^*(\tau)) - 1_{\{Q_j^{(m)}(\tau)=0\}}\mu_{max}) \\
&\quad + V \sum_{\tau=t}^{t+T-1} (\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}} {c_{ji}^{(m)}}^*(\tau) q_i^{(m)} + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}}\sum_{i\in\mathcal{N}} v^{(m)} {s_j^{(m)}}^*(\tau) h + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} v^{(m)} {y_i^{(m)}}^*(\tau) p_j \\
&\qquad + \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{N}} [{y_i^{(m)}}^*(\tau) - {y_i^{(m)}}^*(\tau-1)]^+ w_j^{(m)}),
\end{aligned}
$$

where ${s_j^{(m)}}^*(\tau)$, ${c_{ji}^{(m)}}^*(\tau)$, *and* ${y_i^{(m)}}^*(\tau)$, $\forall i, j \in \mathcal{N}, m \in \mathcal{M}$, *are any alternative decisions that can be made in time slot* τ *within the feasible set.*

Proof of Theorem 5:

Because ${s_j^{(m)}}^*(\tau)$, ${c_{ji}^{(m)}}^*(\tau)$, and ${y_i^{(m)}}^*(\tau)$, $\forall i, j \in \mathcal{N}, m \in \mathcal{M}$, are any alternative decisions that can be made in time slot τ within the feasible set, apparently, ${s_j^{(m)}}^*(\tau)$, ${c_{ji}^{(m)}}^*(\tau)$, and ${y_i^{(m)}}^*(\tau)$, $\forall i, j \in \mathcal{N}, m \in \mathcal{M}$, can be the optimal solution of the problem with information of T time slots into future that minimizes Eqn. (2.17). Then, combining with Lemma 6, we derive

$$
\Delta_T(\Theta(t)) + V \sum_{\tau=t}^{t+T-1} M(\tau) \leq BT^2 + V \sum_{\tau=t}^{t+T-1} M^*(\tau).
$$

Considering the total K frames and summing the above over $k \in \{0, ..., K-1\}$ and then dividing the sum by VKT, we get

$$
\frac{L(\Theta(KT)) - L(\Theta(0))}{VKT} + \frac{1}{KT} \sum_{t=0}^{KT-1} M(t) \leq \frac{BT}{V} + \frac{1}{K} \sum_{k=0}^{K-1} \widehat{M}_k.
$$

Rearranging the terms in the above inequality, and noting that $L(\Theta(KT)) \geq 0$ and $L(\Theta(0)) = 0$, we derive

$$
\frac{1}{KT} \sum_{\tau=0}^{KT-1} M(t) \leq \frac{1}{K} \sum_{k=0}^{K-1} \widehat{M}_k + \frac{BT}{V}.
$$

□

Table 2.1: Notations

$\mathcal{M}$	File set	$\mathcal{N}$	Region set
$v^{(m)}$	Size of file m, in bytes		
$a_j^{(m)}(t)$	# of requests for file m from region j at time slot t		
$Q_j^{(m)}(t)$	Request queue for file m in region j		
A_{max}	Max # of requests for file m from region j in a time slot.		
$s_j^{(m)}(t)$	# of requests dispatched from $Q_j^{(m)}$ to on-premise server at t		
$c_{ji}^{(m)}(t)$	# of requests dispatched from $Q_j^{(m)}$ to data center i at t		
$y_i^{(m)}(t)$	Binary var: store file m on data center i or not at t.		
b	Max # of requests the on-premise server can process in a time slot		
μ^{max}	Max # of requests dispatched from each request queue to a data center in a time slot, i.e., $c_{ji}^{(m)}(t) \leq \mu^{max}$		
g_i	Charge for uploading a byte from the data center i		
o_i	Charge for downloading a byte onto the data center i		
f_i	Charge for renting one VM instance in data center i		
r_i	# of requests a VM in data center i can serve in a time slot.		
p_i	Charge for storing a byte on the data center i		
$q_i^{(m)}$	Charge for uploading file m from the data center i		
h	Time-averaged charge for uploading a byte from the on-premise server		
$w_i^{(m)}$	Charge for migrating file m from on-premise server to data center i.		
$W_j^{(m)}$	Bound of queueing delay of requests in queue $Q_j^{(m)}$		
$\epsilon_j^{(m)}$	Preset constant for controlling queueing delay in $Q_j^{(m)}$		
d_j	round-trip delay between region j and on-premise server		
e_{ji}	round-trip delay between region j and data center i.		
α	Bound of time-averaged round-trip delay		
$G(t)$	Virtual queue for bounding time-averaged round-trip delay		
$Z_j^{(m)}(t)$	Virtual queues for bounding the queueing delay for $Q_j^{(m)}$		

Chapter 3

Concluding Remarks

3.1 Conclusions

This report first investigates state-of-art of research on cloud computing, around the feilds which are related to my interested research topics. Then we study the problem of migration of a content distribution service to a hybrid cloud consisting of private on-premise servers and public geo-distributed cloud services, as a first step in our research on cloud computing.

3.2 Future Directions

3.2.1 Optimization Framework for Migration of Video-on-Demand to Hybrid Clouds

We may extend the current optimization framework for distributing general types of content in Chapter 2 to cater for specific content distribution services with detailed requirements, such as video-on-demand service. Video-on-demand service is more delay sensitive, and users follow certain access pattern that we can make use of.

3.2.2 Budget Allocation in Different Categories of VM Instances

Facing various categories of VM instances, the cloud user needs to model its own computing demand and determine the number of different categories of instances to meet the demand, while minimizing its total operational cost in the long run. When we use *spot instances*, we need to tackle the problem of reliability, i.e., *spot instances*

can benefit the user only when the user has flexibility in when their applications can run, or, can make proper replication in the instance running the same application. We may use replication, migration and checkpointing to develop stochastic scheduling algorithms and fault tolerant mechanisms which can provide satisfactory reliability and minimal operational cost.

Javadi *et al.* (**?**) model spot price dynamics by the mixture of Gaussian's distribution with three or four components based on historical price records. This inspires us to improve some existing systems by planning more jobs to be done when the price is low and fewer jobs when the price is high. For example, in a IPTV service, the application provider can schedule pushing more content to the set-top boxes' cache when the price is low. We may design stochastic online algorithms and use the theory of probability to analyze its performance.

3.2.3 Budget Allocation in Different Sub-type of VM Instances

When we look closer into each category of instances, there are many different sub-types of VM instances, with different prices and computing capacities. The relationship between computing capacity of a sub-type of instances and its price can be modelled as a convex or concave, increasing function for simplicity. The application provider may optimize the number of different sub-types of instances to lease, in order to minimize its operational cost under its SLA constraint.

3.2.4 Pricing of VM Instances

Currently, cloud providers, e.g., Amazon, use the standard uniform price auction mechanism to determine spot prices: the provider assigns resources to bidders in decreasing order of their bids until all available resources have been allocated or all resource requests have been satisfied. The selling price is the lowest winning bid (**?**). Such a pricing mechanism may favour the cloud users but may not maximize the revenue of the cloud provider. New pricing mechanisms are worth to be explored. When we incorporate the sub-contracting of tasks to peer cloud provider contributed instances or user contributed instances, the pricing problem becomes more complex.

3.2.5 Deployment of OSN on Geo-distributed Clouds

Online Social Networks (OSN) have the properties of large scale, distribution of users, skewed access patterns and mobility of users. Deploying OSN on multiple VMs that are possibly geo-distributed, to serve a large group of users who may travel around, is a very complex problem. We may compare different data placement strategies and request scheduling algorithms with queueing models, to show the trade-off between implementation complexity and performance improvement.

3.2.6 Online Task Schedule Algorithm in Federated Clouds

Because requests of computing resource could be served at either local data center, VMs contributed by user groups, or routed to other cloud providers, so there exists an optimal scheduling that maximizes the revenue of the community of cloud providers. We may first model the system as a queueing network where requests for executing tasks enter the queues and departure when there are proper VMs to process them. Then we could adopt Lyapunov optimization technique to develop algorithms which achieve performance with a small constant gap away from the optimality and comply with SLA in terms of queueing delay. After that, we will try to show that local revenue maximization meet optimal social welfare.

3.2.7 Tapping Any Cloud Users' Computing Resource to Any Other Users

If we allow the application provider to use on-premise server contributed by other individual or companies, we have one more dimension to explore. The reliability and pricing are key problems. How to schedule the task and make data placement decision are also interesting problems. To tackle the weakness of unreliability, the application may replicate VMs and data, and migrate VMs when other users send the leave signals. We may verify the feasibility of such an architecture under various user dynamics.

3.2.8 Bandwidth Allocation at Cloud Providers

Although storage and communication is isolated by virtual machines, network bandwidth is shared among all the users in a cloud provider, including intra-cloud network, and WAN. Sharing may bring a lot of problems, e.g., the Quality-of-Service is not

guaranteed. For the bandwidth-intensive and delay-sensitive applications , the cloud provider needs a mechanism to dynamically allocate amount network bandwidth to applications according to their neediness, so that the revenue of the cloud provider and the social welfare could be maximized. We may apply auction theory to design algorithm to deal with that.

Bibliography

[1] D. Parkhill, *The Challenge of the Computer Utility*. Addison-Wesley Educational Publishers Inc., US, 1966.

[2] R. B. et al., "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, pp. 599–616, June 2009.

[3] M. Armbrust and A. Fox and R. Griffith and A. D. Joseph and R. H. Katz and A. Konwinski and G. Lee and D. A. Patterson and A. Rabkin and I. Stoica and and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," in *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, Feb 2009.

[4] B. Rochwerger, D. Breitgand, E. Levy, a. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan, "The Reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 1–11, Jul. 2009.

[5] L. M. Vaquero, L. Rodero-merino, J. Caceres, and M. Lindner, "A Break in the Clouds : Towards a Cloud Definition," *Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2009.

[6] *Amazon Elastic Compute Cloud*, http://aws.amazon.com/ec2/.

[7] *Google App Engine*, http://code.google.com/appengine/.

[8] *Microsoft Azure*, http://www.microsoft.com/windowsazure/.

BIBLIOGRAPHY

[9] N. Chohan and C. Castillo and M. Spreitzer and M. Steinde and A. Tantawi and C. Krint, "See spot run: Using spot instances for mapreduce workflows," in *Proc. of USENIX HotCloud Workshop, 2010*, June 2010.

[10] Y. Sangho, D. Kondo, and A. Andrzejak, "Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud," in *Proc. of IEEE 3rd International Conference on Cloud Computing (CLOUD), 2010*, July 2010.

[11] Q. Zhang, E. Gurses, R. Boutaba, and J. Xiao, "Dynamic Resource Allocation for Spot Markets in Clouds," in *Proc. of Hot-ICE 2011*, March 2011.

[12] J. Broberg and R. Buyya and Z. Tari, "Metacdn: Harnessing 'storageclouds' for high performance content delivery," *Journal of Network and Computer Applications*, pp. 1012–1022, 2009.

[13] M. Bjorkqvist and C. Y. Lydia and V. Marko and X. Zhang , "Minimizing retrieval latency for content cloud," in *Proc. of IEEE INFOCOM 2011*, April 2011.

[14] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, p. 291, Jun. 2005.

[15] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier Internet applications," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 3, no. 1, pp. 1–39, Mar. 2008.

[16] U. Sharma and P. Shenoy and S. Sahu and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Proc. of 31st International Conference on Distributed Computing Systems (ICDCS)*, June 2011.

[17] P. Xiong, Z. Wang, S. Malkowski, Q. Wang, D. Jayasinghe, and C. Pu, "Economical and Robust Provisioning of N-Tier Cloud Workloads : A Multi-level Control Approach," in *Proc. of 31st International Conference on Distributed Computing Systems (ICDCS)*.

[18] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The Cost of a Cloud : Research Problems in Data Center Networks," *Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2009.

[19] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "DONAR : Decentralized Server Selection for Cloud Services," in *Proc. of ACM SIGCOMM 2010*, August 2010.

[20] C. Wu, J. Ho, and M. Chen, "Time-critical event dissemination in geographically distributed clouds," in *Proc. of IEEE Computer Communications Workshops (INFOCOM WKSHPS), 2011*, April 2011.

[21] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: automated data placement for geo-distributed cloud services," in *Proc. of the 7th USENIX conference on Networked systems design and implementation (NSDI)*, 2010.

[22] M. J. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez, "The Little Engine(s) That Could : Scaling Online Social Networks," in *Proc. of ACM SIGCOMM 2010*, August 2010, pp. 375–386.

[23] X. Cheng and J. Liu, "Load-Balanced Migration of Social Media to Content Clouds," in *Proc. of the 21st International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2011)*, June 2011.

[24] M. Li and S. Yu and N. Cao and W. Lou, "Authorized Private Keyword Search over Encrypted Personal Health Records in Cloud Computing," in *Proc. of 31st International Conference on Distributed Computing Systems (ICDCS)*, June 2011.

[25] C. Wang and K. Ren and J. Wang and K. M. R. Urs, "Harnessing the Cloud for Securely Solving Large System of Linear Equations," in *Proc. of 31st International Conference on Distributed Computing Systems (ICDCS)*, June 2011.

[26] M. Hajjat, X. Sun, Y. E. Sung, D. Maltz, and S. Rao, "Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud," in *Proc. of the SIGCOMM (SIGCOMM 2010)*, August 2010.

[27] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, "Intelligent Workload Factoring for a Hybrid Cloud Computing Model," in *Proc. of the International Workshop on Cloud Services (IWCS 2009)*, June 2009.

[28] H. Li, L. Zhong, J. Liu, B. Li, and K. Xu, "Cost-effective Partial Migration of VoD Services to Content Clouds," in *Proc. of the Fourth IEEE International Conference on Cloud Computing (CLOUD 2011)*, July 2011.

[29] T. Lam and T. Woo, "CloudFlex : Seamless Scaling of Enterprise Applications into the Cloud," in *Proc. of INFOCOM (Mini)*, April 2011.

[30] E. Zohar and I. Cidon, "The Power of Prediction : Cloud Bandwidth and Cost Reduction," in *Proc. of SIGCOMM 2011*, August 2011.

[31] *Amazon CloudFront*, http://aws.amazon.com/cloudfront/.

[32] *Dropbox*, http://www.dropbox.com/.

[33] *Microsoft Office Web Apps*, http://office.microsoft.com/en-us/web-apps/.

[34] *Google docs*, http://docs.google.com/.

[35] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1–149, 2006.

[36] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[37] H. Li, W. Huang, C. Wu, Z. Li and F. C.M. Lau, "Utility-Maximizing Data Dissemination in Socially Selfish Cognitive Radio Networks," in *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2011)*, Oct 2011.

[38] M. J. Neely, "Energy optimal control for time varying wireless networks," *IEEE Transactions on Information Theory*, no. 7, pp. 2915–2934, July 2006.

[39] M. M. Amble, P. Parag, S. Shakkottai, and L. Ying, "Content-Aware Caching and Traffic Management in Content Distribution Networks," in *Proc. of the INFOCOM (INFOCOM 2011)*, April 2011.

[40] M. J. Neely and L. Golubchik, "Utility Optimization for Dynamic Peer-to-Peer Networks with Tit-For-Tat Constraints," in *Proc. of the INFOCOM (INFOCOM 2011)*, April 2011.

[41] *Amazon Simple Storage Service*, http://aws.amazon.com/s3/.

[42] M. J. Neely, "Opportunistic Scheduling with Worst Case Delay Guarantees in Single and Multi-Hop Networks," in *Proc. of INFOCOM 2011*, April 2011.

[43] *GLPK (GNU Linear Programming Kit)*, http://www.gnu.org/s/glpk/.

[44] B. Javadi, R. K. Thulasiram, and R. Buyya, "Statistical Modeling of Spot Instance Prices in Public Cloud Environments," in *Proc. of 4th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2011)*, Dec. 2011.