SANDOOQ: IMPROVING THE COMMUNICATION COST AND SERVICE LATENCY FOR A MULTI-USER ERASURE-CODED GEO-DISTRIBUTED CLOUD ENVIRONMENT

BY

SHAYAN SAEED

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Adviser:

Professor Roy H. Campbell

ABSTRACT

Modern data centers have to accommodate the storage of an increasing amount of data with multiple users accessing that data from all over the world. Most of these data centers are geo-distributed to improve availability and protect against the loss of data in the case of outages and disasters. They are also increasingly using erasure codes to improve the reliability at a much lower storage cost. In addition to reliability, the clients and applications also demand storage solutions with better performance and cost-effectiveness. For a geo-distributed data center, a major part of the cost is associated with sending the data between the data centers. This paper builds on previous work to minimize the latency and cost in a data center and applies it to a multi-user geo-distributed environment. We develop a mathematical model for service latency and communication cost for a multi-user geo-distributed cloud environment. We also provide an algorithm to jointly optimize the service latency and communication cost by controlling the placement of the erasure-coded file chunks and scheduling the requests for these chunks. Through simulations, we show that our algorithm converges quickly and outperforms other heuristics in optimizing service latency and communication cost.

To my family, friends and colleagues for their love and support.

ACKNOWLEDGMENTS

I would like express the deepest appreciation to my adviser Professor Roy H. Campbell for his persistent support and expert guidance. I would also like to extend my gratitude to Dr. Vaneet Aggarwal and Dr Yih-Farn R. Chen for their insightful suggestions and advice. I am also grateful to Professor Indranil Gupta, Chris Cai and all the SRG group members who generously helped me and provided me counsel.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Data is being produced at an unprecedented rate these days, both by individuals and corporations. According to one estimate, Facebook generates 4 PB of data everyday [1]. Large-scale cloud systems and distributed storage systems are seeing an increasing demand both on enterprise and personal level. Data storage is not just being provided as part of the package along with computation facilities, but also as a separate on-demand virtual storage. These include the cloud storage services such as Amazon S3 [2], Microsoft Azure [3], Google Cloud [4] and personal storage services like Dropbox [5], Box [6], Apple's iCloud [7]. These have become an essential part of the personal computing experience for most of the users to deal with their increasing storage demands reliably.

For many applications such as social networking, e-commerce and web searching, sharing and access of data by multiple clients is necessary. The clients want sufficiently reliable storage, fast access times as well as lower costs associated with storing their data. Even delays of less than 400 ms can mean a measurable loss of users and revenue [8]. The cloud storage market is quite competitive because of high initial investments [9], so the vendors have to offer better pricing to gain customers. Reliable storage can be ensured by providing redundancy in the stored data through replication or erasure coding to protect against failures. Lower latency and cost are affected by several different parameters, which we will explore in detail subsequently.

In geo-distributed storage, while most of the previous work has focused on solving the problems related to reliability, availability and consistency; many ordinary users care far more about performance and cost. A major part of the cost involved in any geo-distributed system is the communication cost between the data centers since the WAN bandwidth is both limited and costly. The performance experienced by the users is highly influenced by the efficient load distribution in the system. Any attempt to reduce the bandwidth use and balance the load through data placement and request scheduling can play a major role to bring down the latency and costs. Xiang et al. [10] provides a latency model and perform a joint optimization for latency and storage cost for a multi-user data center. We use and extend the latency model to a multi-user geo-distributed cloud environment.

1.1 Thesis Contribution

We develop a mathematical model for service latency and communication cost in a cloud environment with data centers at multiple geographical places. The system model specifies data as being erasure-coded and distributed with many users storing and accessing the data from different locations. We also allow the capability to the users to specify a custom reliability guarantee for their data. Based on this model, we formulate a joint optimization problem for service latency and communication cost. We develop Sandooq algorithm to efficiently perform joint optimization for the formulated problem.

We show that Sandooq algorithm can converge quickly for a large number of users and files. We also demonstrate that Sandooq outperforms other heuristics in terms of optimizing service latency and communication cost. We further study and measure the effects of changing different parameters of the model on our optimization variables.

1.2 Thesis Organization

The rest of this work is organized as follows. We first explain the necessary background, motivation and related work for the problem in Chapter 2. We present the system model and develop the latency and cost model in Chapter 3. We formulate a joint optimization problem for service latency and communication cost and present the Sandooq algorithm to solve the problem in Chapter 4. In Chapter 5, we perform different experiments to evaluate and analyze Sandooq algorithm. We describe potential future work in Chapter 6 and present the conclusions in Chapter 7.

CHAPTER 2

BACKGROUND

2.1 What is Erasure Coding?

Erasure Coding is a technique to ensure reliability in a distributed storage system where a data block is broken down and encoded into several smaller fragments such that the original data can be recovered from a subset of the smaller fragments. Erasure coding has been studied widely for its use in distributed storage systems [11]. It ensures high reliability with lower storage overhead compared to replication, albeit with higher repair and reconstruction costs [12]. It has been widely used in industry by Windows Azure [13], Facebook [14], Ceph [15], QFS [16] etc.

In our work, we use a subtype of erasure codes called Maximum Distance Separable (MDS) codes. A data object O of size S is broken down into k equal sized chunks of size S/k. These k chunks are encoded using an (n,k) MDS code into a total of n chunks. The original k chunks are the *data chunks* while the rest of the n-k chunks are the *parity chunks*. A key property of MDS codes is that the original object O can be reconstructed from any of the k (data or parity) chunks. Thus, for an object to be recoverable, the system can tolerate corruption or loss of at most $n - k$ chunks. Therefore, erasure codes can have some overhead for encoding, decoding and repair but they save a lot on the storage cost by having much lower storage overhead compared to replication.

An example showing the encoding and reconstruction after two failures of a (6,4) MDS encoded file is shown in Figure 2.1. The file F is encoded into four data chunks D_1, D_2, D_3, D_4 and two parity chunks P_1, P_2. Each of the chunks is of size $S/4$. The total size of all the encoded chunks is $3S/2$. Since the original file size was S, this constitutes an overall storage overhead of 1.5x. This (6,4) encoded file can be reconstructed by decoding any of the

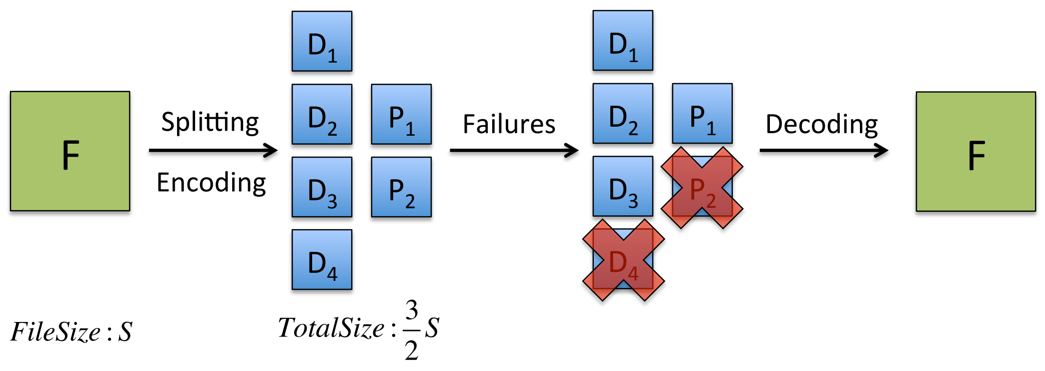

Figure 2.1: An example showing a file encoded with a (6,4) MDS erasure-code. It has 1.5x storage overhead and can tolerate up to 2 failures.

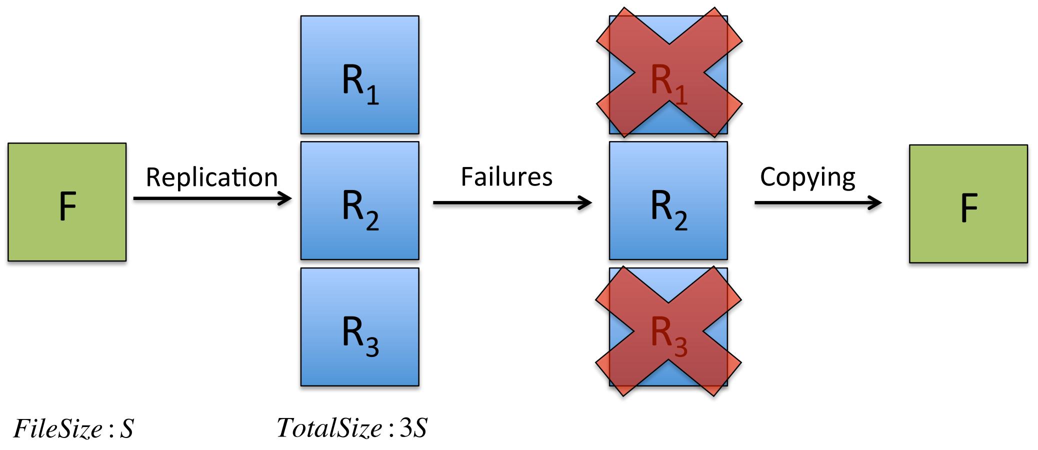

Figure 2.2: An example showing a file with 2 replicas. The storage overhead is 3x that is double compared to that for a (6,4) erasure code. It can also tolerate only up to 2 failures.

four chunks, so the system can tolerate two chunk losses. If any of the two chunks such as D_4 and P_2 fail, the system can still recover the original file by decoding the chunks.

For comparison, a replicated system with a similar fault tolerance is shown in Figure 2.2. The original file has been replicated twice with each replica having a size S. The total size of the original file and the replicas is $3S$, which is 3x overhead. The system can only recover from a loss of 2 of the replicas. Therefore, this replicated file system has twice as much storage overhead than the erasure-coded system described before, for the same fault tolerance. This makes it desirable for the cloud providers to use erasure coding in their data centers to save on the storage cost.

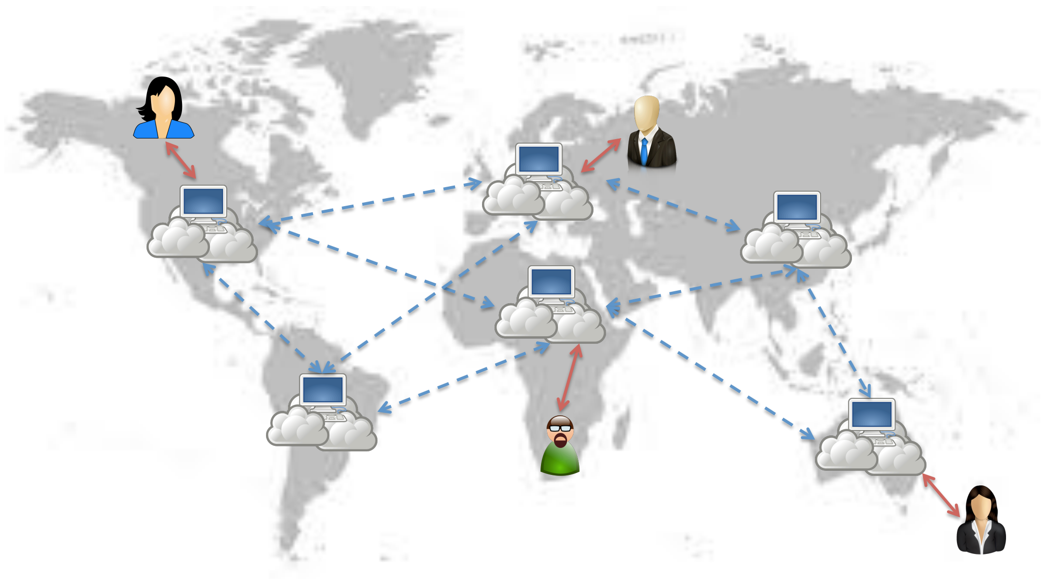
Figure 2.3: A sample multi-user geo-distributed file system. There are multiple data centers at different geographical locations and several users can access the files.

2.2 Geo-Distributed Storage

A geo-distributed storage system is a kind of cloud storage system in which data is stored across several different large data centers at different geographical locations. A simple geo-distributed storage system is shown in Figure 2.3. Each data center consists of local machines connected together with high bandwidth and low latency links. These data centers are connected to each other with WAN links that typically have higher latency than the local links. Such systems generally store a lot of data with many users accessing this data from all around the globe.

The need for distribution across multiple geographical regions arises because of several different design goals. The primary one is the need to scale for a huge storage system. As storage systems hosting the data grow at a rapid rate to millions of users, a single data center cannot scale out easily to cater to the storage needs of these users. Adding more and more storage units to a limited space becomes practically impossible. The only solution is to add another data center and connect them together to give the illusion of a single storage entity [17].

In addition, even a highly reliable data center is prone to occasional failures due to administrative faults, geographical catastrophes etc. [18]. To protect against data loss, it is desirable that the system has the resiliency to protect against the event of a data center failure. A geo-distributed storage system is a practical way to provide disaster tolerance [19]. It also has the potential to provide better availability. Even in the case of intermittent problems with a data center, users can be seamlessly served by data hosted on other data centers.

Another benefit is to provide better quality of service to the users. Latency can directly influence the user experience of using the cloud [20]. By geo-distributing data, you can place the data closer to the client. Whenever a user makes a request for data, that request can be served by the nearest data centers to improve latency [21]. Similarly geo-distribution can also take advantage of the regional differences in energy prices and schedule the requests accordingly to lower the costs [22] [23].

However, the prospect of geo-distributed storage also raises several problems and questions. Since data can be stored at and requested from any of the data centers, any request that is served from a data center to the other end of the globe can increase the cost and latency of retrieving the data significantly. Poor load balancing of the requests can lead to higher loads on one data center which can hurt performance [24]. Another challenge would be choosing the encoding format and scheduling strategy for retrieving the data in a way that provides the best cost-efficiency and maximizes the performance.

2.3 Motivation

We have seen that erasure coding has significantly lower storage overhead for the same reliability guarantees as replication. Using erasure coding in a geo-distributed storage setting can save a lot of money by decreasing the amount of storage needed. At the same time, it can cost more money by using up much of the costly bandwidth between the data centers to transmit the individual chunks. There is also a question of how to schedule the requests coming from multiple users throughout the world to minimize the average service latency for file access. These questions become much more relevant when the system has to provide certain reliability guarantees to the user to protect against the loss of data.

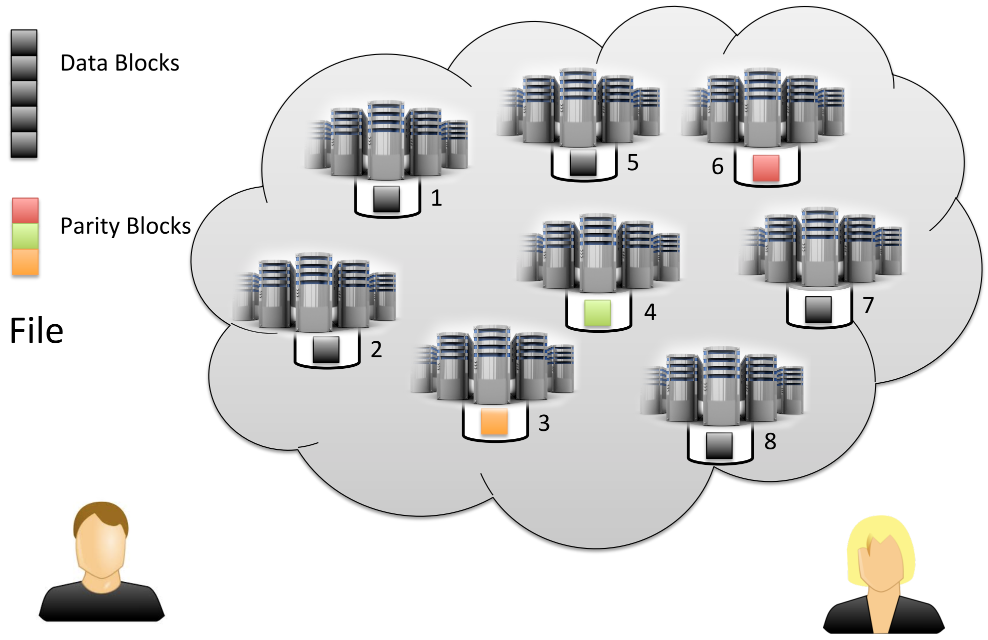

Figure 2.4: An example showing geo-distributed storage system having two users with one file stored with a (8,5) erasure code. Every chunk is placed on a different storage unit.

This can be illustrated by a motivating example shown in Figure 2.4. A file is being stored in the system that has been encoded using a (8,5) erasure code so it has 5 data chunks and 3 parity chunks. These chunks are stored on a geo-distributed storage system with 8 data centers. To maintain reliability guarantees, each data center has hosted only 1 chunk. Two different users are trying to access the same file from the storage system. The file can be reconstructed from any of the 5 chunks hosted on the storage system. We consider two simple heuristics to schedule requests to get the data. (i) The data centers that are nearest to the user will service the request. We call this *Prioritize-Nearest Heuristic*. This can be seen in Figure 2.5. Each of the user gets the chunks from 5 of the nearest data centers to reconstruct the file. (ii) As the requests arrive, they are served by the data centers hosting the data that have the least load. We call this *Balance-Load Heuristic*. This has been shown in Figure 2.6. The least loaded servers serve the user requests.

Prioritize-Nearest Heuristic tends to minimize the network transfer cost. Since the data is served by the nearest data centers, this decreases the distance over which the data has to be sent, which in turn brings down the cost. However, since this does not take into account the load on the data centers,

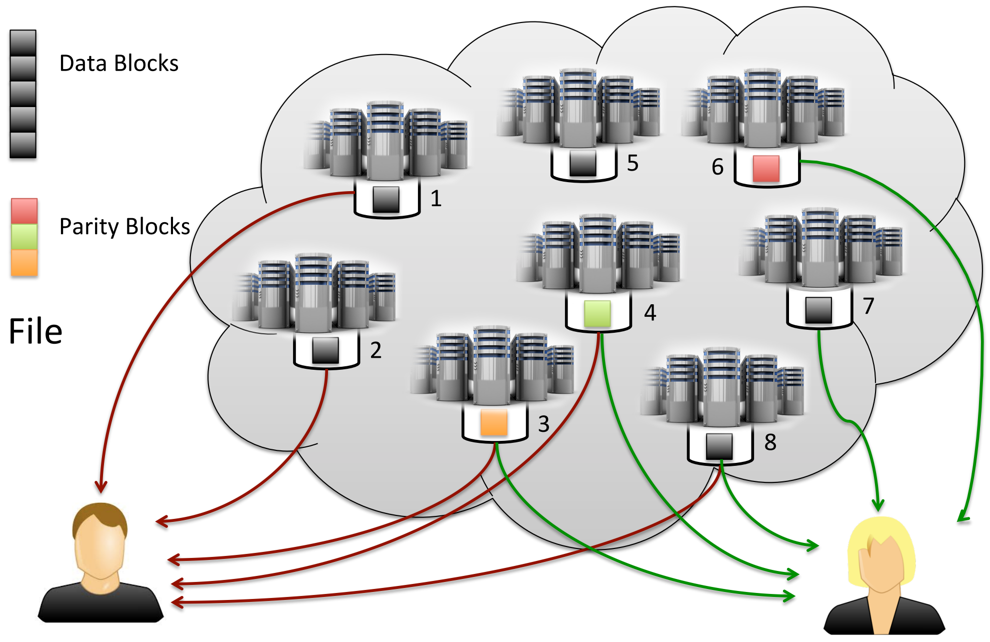

Figure 2.5: An example showing Prioritize-Nearest Heuristic in a geo-distributed cloud. The users get the erasure-coded file chunks from the storage units nearest to them. This reduces the communication cost.

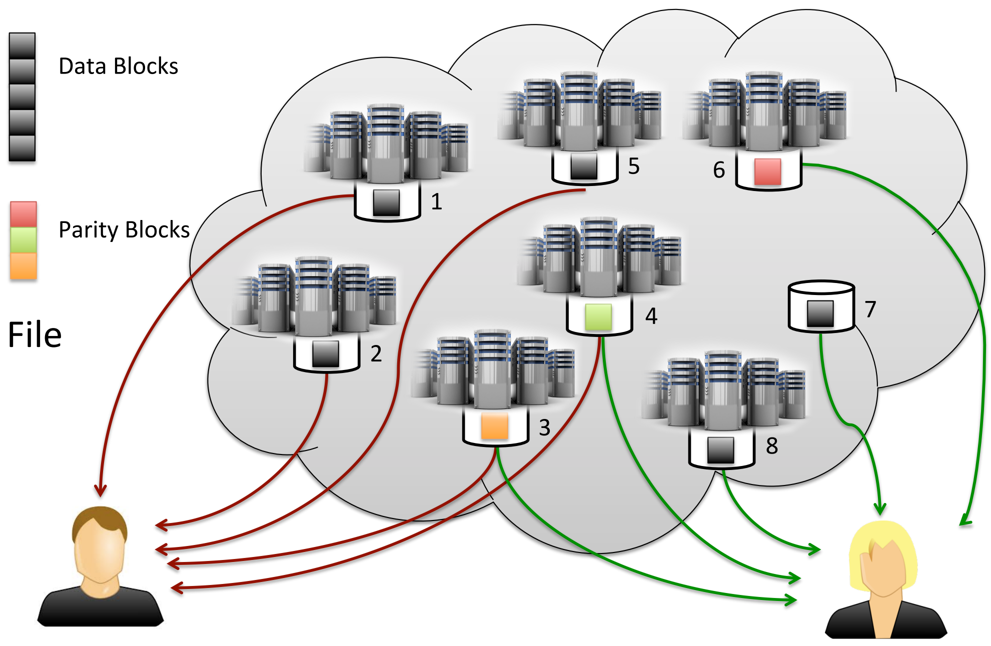

Figure 2.6: An example showing Balance-Load Heuristic in a geo-distributed cloud. The users retrieve the erasure-coded file chunks from the storage units with the least load. This reduces the service latency.

this can increase the service latency. If a lot of requests are coming from the same geographical area, they will be served by the same data centers, which will become loaded. The overloaded data centers will have higher queuing delay for the files that is undesirable for the end users. We can see this in Fig.2.5, that the data center 5 is staying idle while some of the other servers are overloaded.

On the other hand, *Balance-Load Heuristic* evenly distributes the load between the data centers. This makes sure that the service latency does not get very high. However, this can potentially increase the cost to transfer the data significantly since a user request from China can be sent to the US for processing. This causes the data to be transmitted over a lot of distance that is very costly.

In order to better balance out these competing attributes of network cost and service latency, our goal is to provide a model for service latency and communication cost for a multi-user erasure-coded geo-distributed system. Along with the model, we aim to provide an algorithm for the erasure-coded chunk placement and request scheduling to minimize both the service delay and communication cost for the end users.

2.4 Related Work

We divide the related work into different sections based on their focus and describe them briefly.

2.4.1 Minimizing Repair Bandwidth

Erasure coding has been mostly touted as a means to reduce the storage overhead with the same reliability as replication. Most of the research has gone towards reducing the high repair cost associated with erasure codes both in terms of computation and bandwidth [25], [26], [27]. More and more cloud storage companies have also shown a lot of interest in this to try to minimize the repair bandwidth and network I/O in the case of intermittent failures [14], [13]. Our work does not deal with minimizing repair bandwidth, but rather strives to minimize the communication cost for the simple case just by chunk placement and request scheduling. We also use only one type of erasure codes namely Reed-Solomon that is a type of MDS erasure code.

2.4.2 Coding Schemes for Latency Analysis and Improvement

There has been a variety of literature that presents a new erasure coding scheme for improving the reliability with less storage cost or repair bandwidth and also show latency improvements in the evaluation as a performance metric for different system implementations [28], [29], [30], [31], [32]. However, these schemes present latency improvement as just a side-benefit. Our work deals with latency as one of the main metrics. A lot of work has been done to perform a detailed analysis of the queuing latency, including the upper bounds on average latency, for an MDS erasure-coded file system [33], [34], [35], [36], [10]. We use and extend the latency model developed by [10] in our system model.

2.4.3 Costs in a Geo-Distributed Cloud

For a geo-distributed cloud, [37] has identified the major costs associated with a geo-distributed data center. Servers, infrastructure, power and network constitute a majority of the costs in the cloud. A variety of existing literature [38], [39], [40], [41] deals with optimizing the performance and power cost in a distributed data center. They vary various parameters or scheduling strategies to do a power-delay performance analysis. [10] tries to minimize the server cost without compromising the latency by doing a joint optimization of storage cost and service latency. Similar to this we do a joint optimization of communication cost and service latency and devise a placement and scheduling strategy to optimize these metrics.

CHAPTER 3

SYSTEM MODEL

In this section, we formulate a joint service latency communication-cost optimization problem for our system model with a given choice of erasure code with a scheduling strategy called probabilistic scheduling. The variables in this problem include the set of servers on which the chunks are to be placed and the set of servers that serve the requests. We modify the system model and latency model presented by [10] in our problem formulation and develop the communication cost model. The main addition in the model is the change of environment from a single data center to multiple geo-distributed data centers and replacing the storage cost model with the communication cost model. We also allow the users to specify a custom reliability guarantee for the files stored in the cloud. We shall describe the system model as well as the service latency and communication cost models in more concrete terms hereafter.

The system consists of a geo-distributed storage environment with m data centers, denoted by $\mathcal{M} = \{1, ..., \mathrm{m}\}$. Each of these data centers can contain multiple inter-connected storage devices but for the sake of our problem, we treat each data center as a single entity and we will refer to it as a *storage unit* throughout this paper. The whole system stores a set of r files. Each file i is divided into fixed sized k chunks which are further encoded by an (n, k) MDS erasure code into a total of n chunks of the same size. The file i can be reconstructed by getting any of the k chunks. Each of these chunks is placed onto a storage unit, such that file i is placed onto a set $\mathcal{S}_i \subseteq \mathcal{M}$. The reliability guarantee can be specified by the client that each storage unit can only host up to $c \leq k$ chunks of a file. This requirement can account for any failure or outage of a storage unit. This may be necessary if enough storage units are not available or the client does not want to account for the unlikely events that can bring a data center down.

A file can be requested by multiple clients from any of the storage units throughout the world. The probability that a file i will be requested from a storage unit l by any client is given by $P_{i,l}$. Each file i can be reconstructed by sending a request to any k of the n storage units hosting the file. In probabilistic scheduling as described in [10], k scheduling requests are randomly sent to get k chunks of the file. Every set $\mathcal{A}_i \subseteq \mathcal{S}_i$ has a probability $\mathcal{P}(\mathcal{A}_i)$ of being selected to serve the request for a file i.

Using this system model, the original latency and communication cost optimization problem can be described as two smaller subproblems.

Chunk Placement Subproblem

Given a choice of erasure code (n,k), find the set $\mathcal{S}_i \subseteq \mathcal{M}$ of storage units on which the chunks of a file i should be placed such that the latency and communication cost is minimized.

Request Scheduling Subproblem

Given a choice of erasure code (n,k) and chunk placement $\mathcal{S}_i$ for a file i, find the probability $\mathcal{P}(\mathcal{A}_i, \forall \mathcal{A}_i \subseteq \mathcal{S}_i$ that minimizes the average latency and communication cost.

It should be noted that both these subproblems are closely related. Optimal chunk placement, along with optimal request scheduling can minimize the latency and network cost. If chunk placement is arbitrarily chosen, solving for best request scheduling can only lead to sub optimal results. Similarly, for optimal chunk placement, arbitrarily scheduling the requests would not give the optimal result. We shall now develop and describe the mathematical models for service latency and communication cost for the aforementioned system model.

3.1 Latency Model

The access requests for files are independent of each other. Therefore, we assume that the arrival rate for the access requests for a file forms a Poisson process with rate λ_i. Let the probability that a storage unit j receives a request for v chunks of a file i from a storage unit l, if k chunk requests for the file i have gone out, be denoted as $\pi_{i,j}^{v}$. [10] has shown that the probabilistic scheduling policy would be feasible iff:

$$\sum_{j=1}^{m}\sum_{v=1}^{c} v\pi_{i,j}^{v} = k,\ \pi_{i,j} \in [0,1], \forall i, j \ and \ \pi_{i,j} = 0, \forall j \notin \mathcal{S}_i \tag{3.1}$$

Intuitively, this makes sense because $\sum_{v=1}^{c} v\pi_{i,j}^{v}$ is the average number of chunks retrieved from any storage unit j. A total of k chunks have to be retrieved from all the storage units collectively to reconstruct the file i.

Every storage unit maintains a single queue for all incoming requests for different hosted chunks. Thus, the arrival rate $\Lambda_{l,j}$ of chunk requests at a storage unit j from a storage unit l is also a Poisson process since it is formed by the superposition of r individual arrival requests, all of which are Poisson processes as specified by the model. Since the probability that a file i will be requested from a storage unit l is $P_{i,l}$ and chunk arrival request for a file i at the storage unit j would have a rate $\lambda_i \sum_{v=1}^{c} v\pi_{i,j}^{v}$, it can be seen that $\Lambda_{l,j}$ is given by:

$$\Lambda_{l,j} = \sum_{i=1}^{r}\sum_{v=1}^{c} \lambda_i P_{i,l} v\pi_{i,j}^{v} \tag{3.2}$$

Any incoming request to a storage unit j spends some time waiting in the local queue. Let $Q_{l,j,v}$ be the random wait time a request dispatched from storage unit i to storage unit j to retrieve v chunks of a file, spends in the queue of storage unit j. The queuing time is determined by the chunk request arrival rate at the storage units as well as the processing power of the storage units. We have already seen the expression for the chunk arrival request rate. The processing power of the storage units is characterized by the service time for a request on the storage unit. Let X_j be the random service time at the storage unit j with an arbitrary distribution satisfying finite mean $\mathbb{E}[X_j] = 1/\mu_j$, variance $\mathbb{E}[X_j^2] - \mathbb{E}[X_j]^2 = \sigma_j^2$, second moment $\mathbb{E}[X_j^2] = \Gamma_j^2$ and third moment $\mathbb{E}[X_j^3] = \hat{\Gamma_j}^3$. Similar to the model in [10], the expected latency of a file i is given by:

$$\begin{aligned}\bar{T}_i \leq \min_{z\in\mathbb{R}} \Bigg\{ z &+ \sum_{j\in\mathcal{S}_i} \sum_{\substack{l\in\mathcal{S}_i\\ l\neq j}} \sum_{v=1}^{c} \frac{\pi_{i,j}^{v}}{2} \left(\mathbb{E}[Q_{l,j,v}] - z\right) \\ &+ \sum_{j\in\mathcal{S}_i} \sum_{\substack{l\in\mathcal{S}_i\\ l\neq j}} \sum_{v=1}^{c} \frac{\pi_{i,j}^{v}}{2} \left[\sqrt{(\mathbb{E}[Q_{l,j,v}] - z)^2 + Var[Q_{l,j,v}]}\right] \Bigg\}\end{aligned} \tag{3.3}$$

where,

$$\mathbb{E}[Q_{l,j,v}] = \frac{1}{\mu_j} + \frac{\Lambda_{l,j}\Gamma_j^2}{2(1-\rho_{l,j})} \tag{3.4}$$

$$Var[Q_{l,j,v}] = \sigma_j^2 + \frac{\Lambda_{l,j}\hat{\Gamma_j}^3}{3(1-\rho_{l,j})} + \frac{\Lambda_{l,j}^2\Gamma_j^4}{4(1-\rho_{l,j})^2} \tag{3.5}$$

$$\rho_{l,j} = \Lambda_{l,j}/\mu_j < 1, \forall l, j \tag{3.6}$$

$\rho_{l,j}$ is the request intensity at storage unit j from l. The additional summations in Equation 3.3 are because of the multi-user environment with an additional condition that multiple chunks can be placed on the same storage unit.

An sample system with multiple users generating requests for file 1 is shown in Figure 3.1. It shows a heterogeneous system with $m = 6$ storage units having average service time $\mu_j, 1 \leq j \leq 6$ between 0.06 sec to 0.13 sec. An $(n, r) = (5, 3)$ erasure coded file has been stored with chunks placed on five of the storage units $(c = 1)$. $r = 3$ users generate requests for that file with the overall request arrival rate λ_1 for the file being 0.1 req/sec. Every storage unit has a given probability $P_{1,j}, 1 \leq j \leq 6$ for receiving the request to access the file with these probabilities adding up to 1. Every storage unit receiving a file request sends r chunk requests to the storage units hosting the chunks. The probability to send a chunk request to the storage unit is $\pi_{1,j}, 1 \leq j \leq 6$. These sum up to $k = 3$. $\pi_{1,6} = 0$ since storage unit 6 does not host any chunk for file 1.

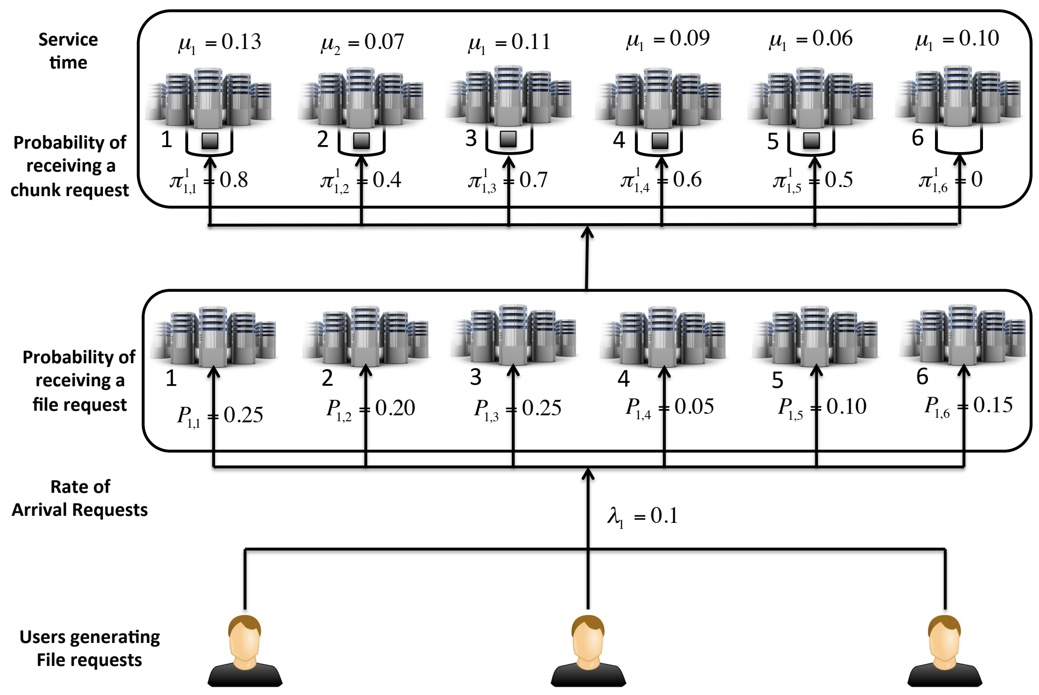

Figure 3.1: A sample multi-user geo-distributed storage system showing the processing of access requests for one file by multiple users.

3.2 Cost Model

We only consider communication cost in our analysis which is the cost to transmit a chunk from one storage unit to another. We assume there is uniform pricing and the communication cost is directly proportional to the amount of data sent as well as the distance across which the data was sent. Going by this assumption, greater amount of data sent directly corresponds to a higher cost. The cost is also higher for data transmitted over far-off distances. So the goal is to minimize the amount of traffic sent between the storage units as well as to minimize the data transfer between the far-off storage units.

We know that $P_{i,l}$ is the probability that a client requests file i from storage unit l. For this file i, v requests are dispatched to the storage unit j to retrieve them with conditional probability $\pi^v_{i,j}$ for $v = 1, ...c$. The average number of chunks retrieved from storage unit j would thus be given by $\sum_{v=1}^{c} v\pi^v_{i,j}$. Therefore, the total data transmitted from the storage unit j to l for a file i is:

$$D_{i,l,j} = C \sum_{v=1}^{c} P_{i,l} v \pi_{i,j}^{v} \tag{3.7}$$

where C is the given fixed size of an erasure coded chunk of any file i.

Under our cost model, the cost of retrieving a file i is the product of the traffic sent between the storage units and the distance between the storage units. This can be represented as:

$$C_i = \sum_{j \in \mathcal{S}_i} \sum_{\substack{l=1 \\ l \neq j}}^{m} D_{i,l,j} d_{l,j} \tag{3.8}$$

where $D_{i,l,j}$ represents the amount of data sent from storage unit j to l during the process of retrieval of file i whereas $d_{l,j}$ is the distance between the storage units l and j.

CHAPTER 4

JOINT LATENCY AND NETWORK COST OPTIMIZATION

4.1 Formulating a Joint Optimization

In this section, we will combine the latency and communication cost models that we developed in the previous section into a joint cost function. We will formulate a joint optimization problem using this cost function and solve it under the constraints specified by the system model. We know that λ_i represents the arrival rate of all the requests for a file i by all the clients in the system. Let $\hat{\lambda} = \sum_{i=1}^{r} \lambda_i$ be the aggregate arrival rate for all the access requests for all the files received by all the storage units in the system. The average latency to access all files would then be given by $\sum_{i=1}^{r}(\lambda_i/\hat{\lambda})T_i$ where T_i is the expected latency to access a file i. The joint objective function for minimization problem of latency-network cost would be given by:

$$\begin{aligned} & min \sum_{i=1}^{r} \frac{\lambda_i}{\hat{\lambda}} T_i + \theta \sum_{i=1}^{r} C_i \\ & s.t.\ (3.1), (3.2), (3.3), (3.4), (3.5), (3.8), (3.7) \\ & var.\ \mathcal{S}_i, \pi_{i,j}^{v}, \forall i, j, v \end{aligned} \tag{4.1}$$

where $\theta \in [0, \infty]$ is the tradeoff factor between the average latency and network cost for the file access. The solutions that give higher importance to minimizing latency will set this value to be much lower than the solutions which prioritize network cost minimization over latency.

This optimization problem has two variables: $\mathcal{S}_i$ that determines placement and $\pi_{i,j,v}$ that determines request-scheduling . This problem is also subject to various constraints that were derived in the previous chapter. Plugging the results from previous section into Equation 4.1, we can arrive at the following optimization problem:

$$\begin{aligned}
\min \quad & z+\sum_{j=1}^{m}\sum_{\substack{l=1\\ l\neq j}}^{m}\sum_{v=1}^{c}\frac{\hat{\lambda}_{l,j,v}}{2\hat{\lambda}}\left[X_{l,j,v}+\sqrt{X_{l,j,v}^2+Y_{l,j,v}}\right] \\
& +\theta\sum_{j=1}^{m}\sum_{\substack{l=1\\ l\neq j}}^{m}D_{l,j}d_{l,j} && (4.2)\\
s.t. \quad & X_{l,j,v}=\frac{v}{\mu_j}+\frac{\Lambda_{l,j}\Gamma_j^2}{2(1-\rho_{l,j})}-z, \forall l,j,v && (4.3)\\
& Y_{l,j,v}=v\sigma_j^2+\frac{\Lambda_{l,j}\hat{\Gamma}_j^3}{(1-\rho_{l,j})}+\frac{\Lambda_{l,j}^2\Gamma_j^4}{(1-\rho_{l,j})^2}, \forall l,j,v && (4.4)\\
& \rho_{l,j}=\Lambda_{l,j}/\mu_j<1, \forall l,j && (4.5)\\
& \hat{\lambda}_{l,j,v}=\sum_{i=1}^{r}\lambda_i P_{i,l}\pi_{i,j}^v, \forall l,j,v && (4.6)\\
& \Lambda_{l,j}=\sum_{i=1}^{r}\sum_{v=1}^{c}\lambda_i P_{i,l}v\pi_{i,j}^v, \forall l,j && (4.7)\\
& D_{l,j}=C\sum_{i=1}^{r}\sum_{v=1}^{c}P_{i,l}v\pi_{i,j}^v, \forall l,j,l\neq j && (4.8)\\
& \sum_{j=1}^{m}\sum_{v=1}^{c}v\pi_{i,j}^v=k;\ \pi_{i,j}^v\in[0,1];\ \pi_{i,j}^v=0,\forall j\notin\mathcal{S}_i && (4.9)\\
& |\mathcal{S}_i|\leq n \text{ and } \mathcal{S}_i\subseteq\mathcal{M},\forall i && (4.10)\\
& \sum_{l=1}^{m}P_{i,l}=1,\forall i && (4.11)\\
\text{var.} \quad & z,\ \mathcal{S}_i,\ \pi_{i,j}^v,\forall i,j,l. \\
\text{given} \quad & k,n,C,P_{i,l},\lambda_i,\mu_j,\Gamma_j^2,\hat{\Gamma}_j^3 && (4.12)
\end{aligned}$$

The summations in Equation (4.2) are changed from $j\in\mathcal{S}_i$ and $l\in\mathcal{S}_i$ to $j=1,..m$ and $l=1,..,m$ because $\pi_{i,j}^v=0,\forall j\notin\mathcal{S}_i$ i.e. for all the storage units that does not host the chunks. The rest of the equations are just obtained by naming and substitution of variables.

This problem is hard to solve because the optimization variables depend upon each other. Changing the nodes $\mathcal{S}_i$ for chunk placement also changes the conditional probabilities $\pi_{i,j}^v$ for request scheduling because of Equation 4.9.

4.2 Sandooq Algorithm

We propose Sandooq algorithm to solve the joint optimization problem for service latency and communication cost. First we need to shorten the problem space. We replace $\mathcal{S}_i$ with an indicator function of $\pi_{i,j}^v$. Only the nodes with $\pi_{i,j}^v = 0$ is retained in the subset $\mathcal{S}_i$ of the overall node set $\mathcal{M}$, since they are the only ones that determine the access latency and network cost of a file i. This leads to $\mathcal{S}_i$ being removed as a variable from the optimization problem and the addition of an extra constraint,

$$\sum_{j=1}^{m} \mathbf{1}_{(\pi_{i,j}^v > 0)} = n, \forall i, j, v \tag{4.13}$$

The first part of the Equation (4.2), which is the service latency is convex in $\pi_{i,j}^v$ as shown by [10]. The communication cost, which constitutes the second part of Equation (4.2), clearly is convex w.r.t. $\pi_{i,j}^v$ since it is simply a linear function of $\pi_{i,j}^v$. This shows that the objective function is convex in $\pi_{i,j}^v$ when other variables are fixed and it is subjected to linear constraints. It can be solved by any of the popular convex optimization tools such as subgradient projection methods, interior-point methods etc.

We use the gradient descent method to solve our optimization problem. The algorithm works in exactly the same way as mentioned in [10] except for the part that the storage cost has been replaced by the communication cost. The proposed algorithm is shown in Algorithm 1. For each iteration, we minimize the objective function over $(\pi_{i,j}^v, \forall i, j, v)$ while keeping the parameter z fixed. The updated probabilities $(\pi_{i,j}^v, \forall i, j, v)$ in each step have to be projected on to the feasibility set $\{\sum_{j=1}^{m} \sum_{v=1}^{c} v\pi_{i,j}^v = k_i;\ \pi_{i,j}^v \in [0, 1]; \sum_{j=1}^{m} \mathbf{1}_{(\pi_{i,j}^v > 0)} = n, \forall i, j, v\}$. After this, we minimize the objective function over $z \in \mathbb{R}$ for fixed values of probabilities $(\pi_{i,j}^v, \forall i, j, v)$. We repeat this until we reach a minimum when the change in value is within a small enough tolerance value.

Algorithm 1 Sandooq Algorithm

```
Initialize $t = 0$ and feasible $z(0), \pi_{i,j}^v(0), \forall i, j$ and choose a small $\epsilon$
Initialize $P_{i,l}, C, \lambda_i, \mu_j, \sigma_j, \Gamma_j^2, \hat{\Gamma}_j^3, \forall i, j, l$ to actual or estimated values
Evaluate the initial value of the objective F(0) using (4.2)
while $F(t) - F(t-1) > \epsilon$ do
    Compute $\pi_{i,j}^v(t+1), \forall i, j$ = arg min (4.2) by calling PROJECTED_GRADIENT()
    $z(t+1)$ = arg min (4.2)
    Compute new objective F(t+1)
    Update $t = t + 1$
end while
Find $\mathcal{S}_i = j : \pi_{i,j}^v > 0, \forall i$
return $(\mathcal{S}_i, \pi_{i,j}^v), \forall i, j, v$

function PROJECTED_GRADIENT()
    Choose a small step size $\delta$
    Initialize $s = 0$ and $\pi_{i,j}^v(s) = \pi_{i,j}^v(t)$
    while $\sum_{i,j,v} |\pi_{i,j}^v(s+1) - \pi_{i,j}^v(s)| > \epsilon$ do
        Calculate gradient $\nabla(4.2)$ w.r.t $\pi_{i,j}^v$
        Compute $\pi_{i,j}^v(s+1) = \pi_{i,j}^v(s) + \delta.\nabla(4.2)$
        Project $\pi_{i,j}^v(s+1)$ onto the feasibility set:
            $\{\pi_{i,j}^v(s+1) : \sum_{j=1}^m \sum_{v=1}^c v\pi_{i,j}^v = k_i;\ \pi_{i,j}^v \in [0,1];$
                $\sum_{j=1}^m \mathbf{1}_{(\pi_{i,j}^v > 0)} = n, \forall i, j, v\}$
        Update $s = s + 1$
    end while
    return $(\pi_{i,j}^v, \forall i, j, v)$
end function
```

CHAPTER 5

EVALUATION

5.1 Testbed

We implemented the Sandooq algorithm in MATLAB to get the optimal placement and scheduling for the chunks of the files. We used MOSEK, which is an optimization framework for MATLAB. The algorithm is run on a quad-core machine with an i7 CPU processor and 8 GB RAM. We run it for 1000 files. We assume that the files have been coded using a (6,4) erasure code with each chunk equally sized at $C = 10MB$ and these chunks can be placed on any of the 12 storage units, each located in a different geographical area (different US State). The distance between these storage units is shown in the Table 5.1.

	WA	NV	CA	ND	CO	TX	IL	GA	KS	MS	DC	FL
WA	0	573	1065	1195	1021	1772	1735	2181	1504	2488	2324	2735
NV	573	0	494	1261	789	1401	1686	1993	1346	2520	2271	2472
CA	1065	494	0	1452	834	1156	1733	1890	1336	2582	2272	2271
ND	1195	1261	1452	0	643	1150	569	1116	549	1298	1138	1721
CO	1021	789	834	643	0	773	919	1211	558	1767	1491	1727
TX	1772	1401	1156	1150	773	0	981	819	637	1696	1318	1115
IL	1735	1686	1733	569	919	981	0	589	413	850	594	1193
GA	2181	1993	1890	1116	1211	819	589	0	677	937	543	607
KS	1504	1346	1336	549	558	637	413	677	0	1249	942	1244
MS	2488	2520	2582	1298	1767	1696	850	937	1249	0	394	1260
DC	2324	2271	2272	1138	1491	1318	594	543	942	394	0	928
FL	2735	2472	2271	1721	1727	1115	1193	607	1244	1260	928	0

Table 5.1: Distance between the different Storage Units in miles.

5.2 Experiments

We perform different experiments to evaluate the Sandooq algorithm.

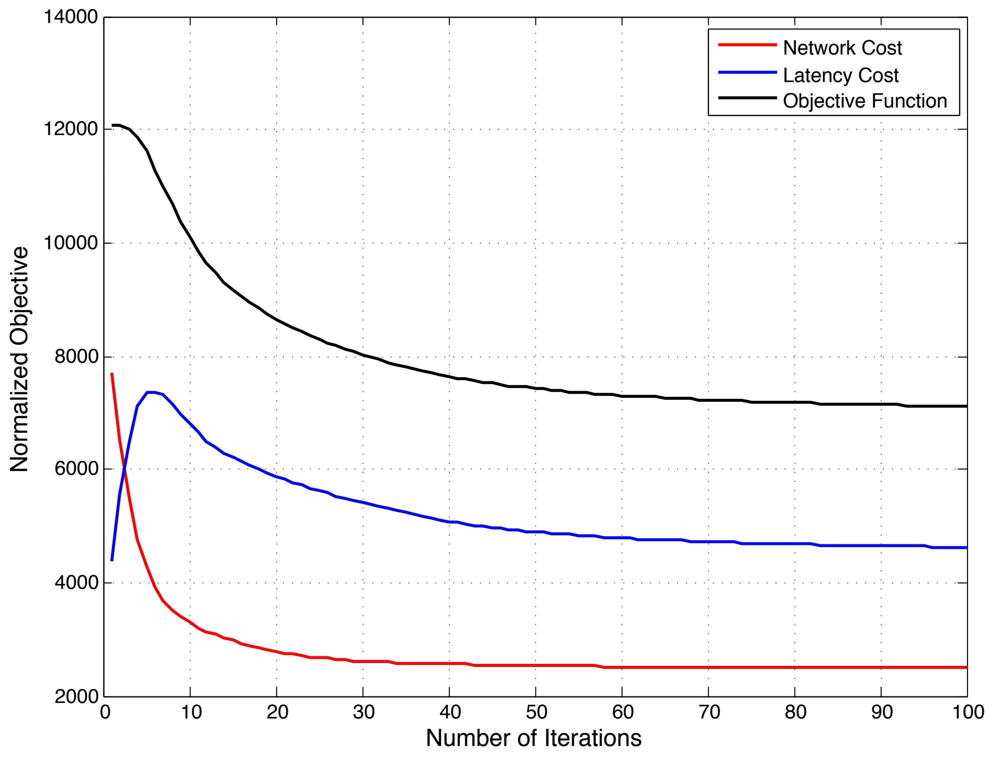

Figure 5.1: Number of iterations required for Sandooq algorithm to converge for a geo-distributed system with 1000 files and 12 storage units. It can be seen that the algorithm converges within about 100 iterations.

5.2.1 Convergence Time of Sandooq

Using the parameters described above, Figure 5.1 shows the number of iterations it takes for the Sandooq algorithm to converge. We can see that the algorithm converges in less than 100 iterations within a tolerance factor of 0.1. An iteration takes about 3.5 seconds on average. So the overall algorithm finishes in nearly 350 seconds. It is also interesting to note the trend for the objective function as well as the network and latency costs. All these values have been normalized by their minimum values respectively. The objective decreases rapidly at first but then starts smoothing off. We can also see the effect of the trade-off factor at play here. The network cost decreases at a more rapid rate initially and overwhelms the latency cost which increases. Afterwards, the increase in the latency cost factors more and pushes the algorithm to start making decisions to decrease both of them.

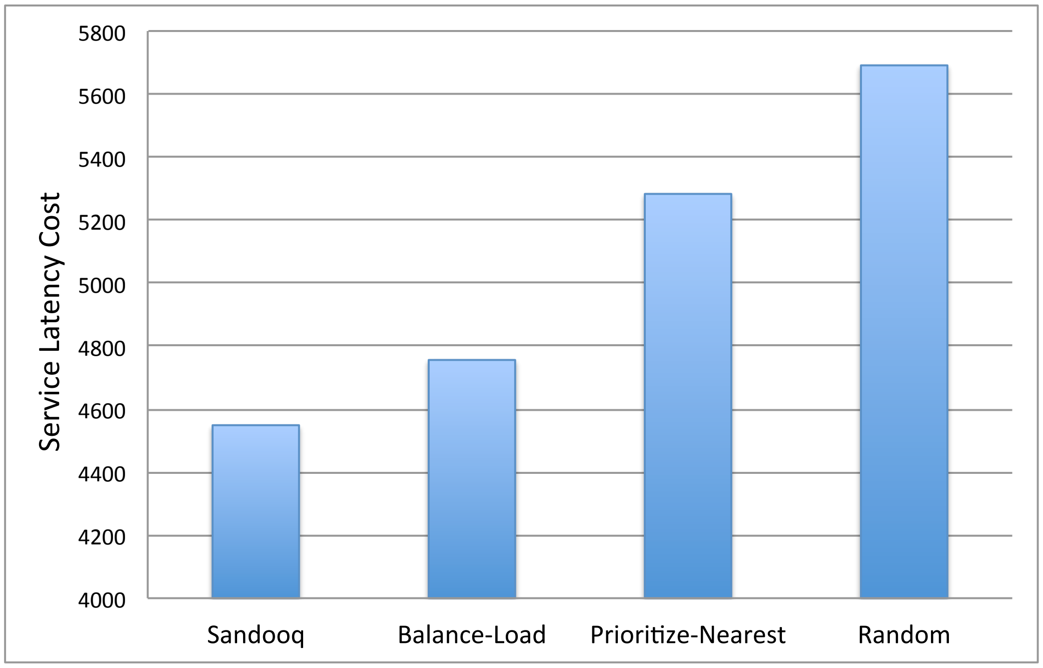

Figure 5.2: Comparison of Sandooq algorithm with other heuristics in terms of the Service Latency.

5.2.2 Performance Comparison with other Heuristics

To evaluate the performance of our algorithm, we compare it with three other heuristics, Balance-Load, Prioritize-Nearest and Random. Two of these heuristics, Balance-Load and Prioritize-Nearest try to minimize either the service latency or the network transfer cost. *Random Heuristic* places the chunks randomly to any of the 6 among the 12 storage units. For the request arrivals, it again schedules these requests randomly to 4 of the 6 storage units hosting the file. *Balance-Load Heuristic* uses the optimal chunk placement calculated by Sandooq algorithm. It then schedules the requests to the storage units proportional to their service time. So more requests would be scheduled to the faster storage units that would potentially reduce the service latency. *Prioritize-Nearest Heuristic* places the file chunks on 6 of the nearest storage units to the user who sent the put request. For request scheduling, it services the request by the storage units that have the minimum distance from the user making the get request. This heuristic strives to reduce the communication cost by reducing the distance data has to travel through the network.

We run these algorithms for 1000 files encoded with (6,4) erasure code with

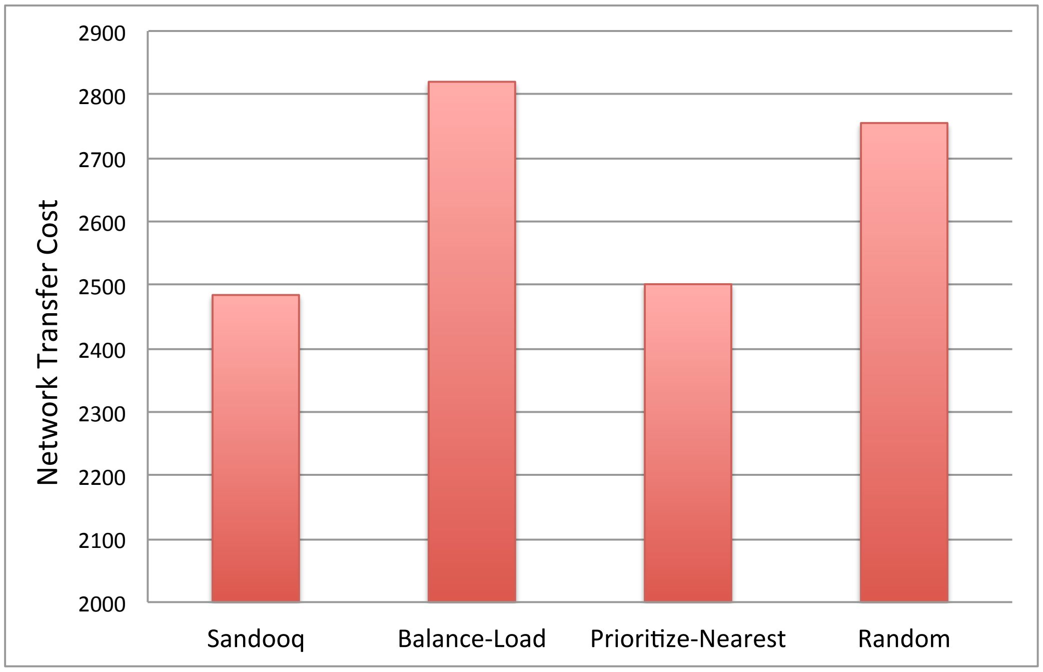

Figure 5.3: Comparison of Sandooq algorithm with other heuristics in terms of the Communication Cost.

each chunk having a size of 10 MB. We assign the cost for transmitting each chunk over a distance of 1 mile to be \$1. We assume a heterogeneous cluster with service time ranging between 0.05 requests/sec to 0.1 requests/sec. We choose the arrival rate for each file to be between 0.0001 to 0.00005 that leads to an aggregate arrival rate of nearly 0.075 requests/sec for all the files. We simulate this system and calculate the network transfer cost and service latency according to our model.

Figure 5.2 and Figure 5.3 show how well Sandooq performs compared to the heuristics. We can see that Sandooq outperforms all these heuristics in terms of minimizing both the service latency and communication cost. More specifically, the prioritize-nearest heuristic comes pretty close to Sandooq in network transfer cost. Similarly, the balance-load heuristic also does a fairly good job of minimizing the service latency. However, both of these heuristics perform poorly in terms of the other metric. However, if the system designer prioritizes only service latency or the network transfer cost, they may resort to using one of these simple heuristics.

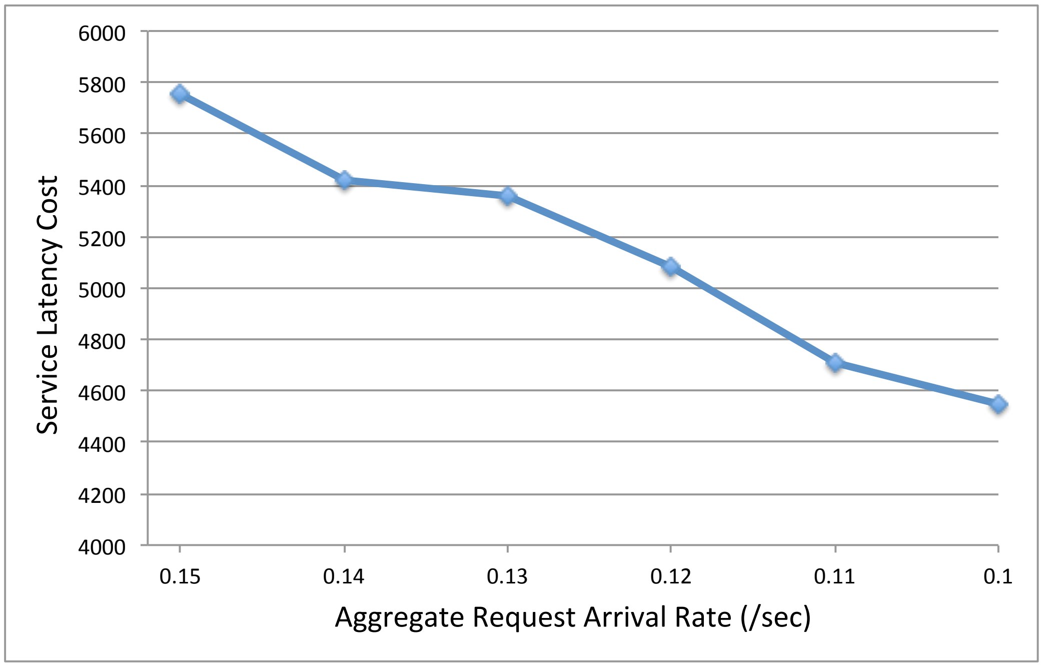

Figure 5.4: Impact of the difference in request arrival rate on the service latency under Sandooq algorithm. The service latency decreases by lowering the request arrival rate.

5.2.3 Effect of Request Arrival Rate on Service Latency

We also vary the individual arrival rate of the requests to measure the impact on the service latency under Sandooq algorithm. The aggregate request arrival rate is varied from 0.1 requests/sec to 0.15 requests/sec. Higher arrival rate means more load on the storage units that translates directly into higher service latency. The results for the experiment are displayed in Figure 5.4. We can see that the service latency is directly correlated with the request arrival rate. In order to keep the latency under control, we would either need to add more storage units to better balance the load or upgrade the storage units to faster ones

5.2.4 Effect of Chunk Size on Communication Cost

To measure the impact of choosing the chunk size on the network transfer cost under Sandooq algorithm, we vary the chunk size of the files from 10 MB to 50 MB. Even if the file is not big enough, the chunks are padded to the chosen size. A higher chunk size would directly translate into more network

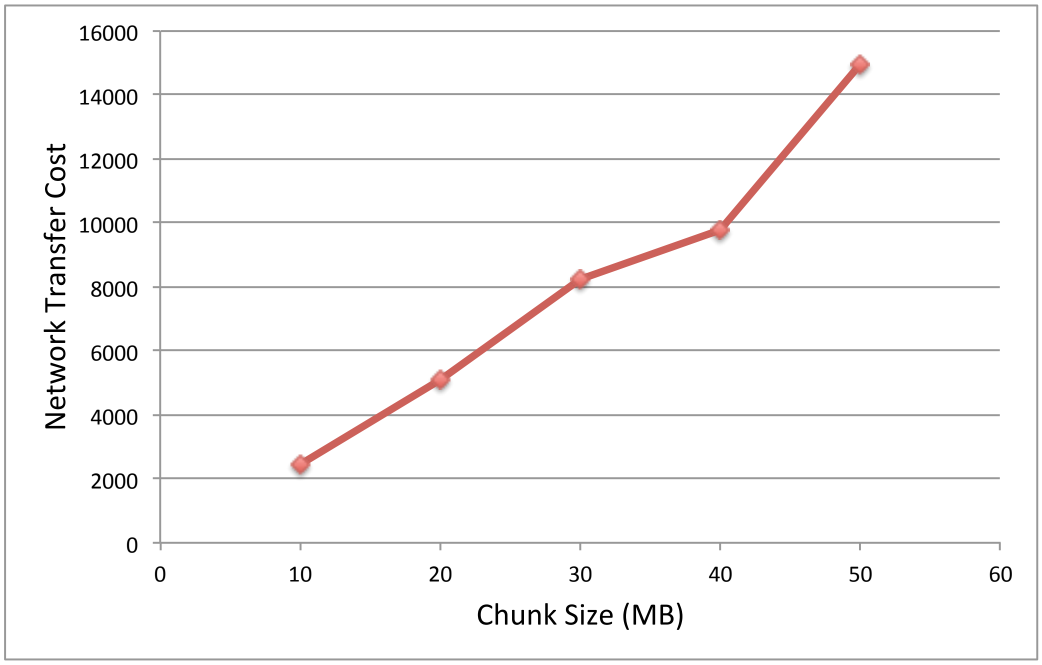

Figure 5.5: Impact of chunk size on the communication cost under Sandooq algorithm. The commuication cost increases by increasing the chunk size.

traffic in the system that raises the communication cost. The results from this experiment are shown in Figure 5.5. We can see that the communication cost increases as the chunks become bigger in size because of increasing cost associated with transporting more data.

CHAPTER 6

FUTURE WORK

6.1 Extending the System Model

In this document, we considered the design of a multi-user erasure-coded geo-distributed system and formulated the Sandooq algorithm to optimize the service latency and the communication cost associated with storing and accessing the files. We achieved this by coming up with a custom strategy that optimally places the file chunks on the storage units and performs request scheduling. The model can be extended to a system with different tiers of storage coded in a different manner. An example of this is a system in which the files in hot storage are replicated while the files in cold storage are erasure coded. We can also extend this to a system where the arrival requests for the files can have any general distribution than the simpler Poisson one. In addition, the current latency model is only for object storage. Extending it to a file system with block storage with the same reliability guarantees is also a possible future work.

The latency model right now only considers the service latency at each of the storage units. We can add network transfer latency between the storage units as well as the latency between the user and the point-of-contact data center for a better and more accurate latency model. Similarly the cost model could also be improved by including the other costs associated with a geo-distributed data center such as storage cost and power cost. We can also look for ways to incorporate other parameters other than latency and monetary cost. The joint optimization of either latency or cost with other parameters of importance such as repair time or repair bandwidth can be performed.

6.2 Evaluation on a Real System

Most of the experimental validation for the model and the algorithm has been performed through simulations for a heterogeneous cluster assuming certain service latency and arrival rates. However, real utility for the scheme should be tested on a real geo-distributed cloud system using an erasure coded object file system. The experiments could be performed by using a trace of requests from an existing multi-user geo-distributed file system. If such a trace is not available, it can be estimated by procuring a trace of requests for a simple file system and assigning the origin of these requests to different regions of the world.

CHAPTER 7

CONCLUSION

We developed a mathematical model for the service latency and communication cost for a multi-user geo-distributed cloud environment. We formulated a joint optimization problem for the service latency and communication cost and developed the Sandooq algorithm that can efficiently solve this problem. Through simulations, we show that even for a large number of users, our algorithm converges within a few iterations and outperforms other heuristics significantly in optimizing service latency and communication cost. It provides a good benchmark to test other scheduling and placement algorithms against. We also studied and showed the impact of changing different parameters of the model on the cost and latency.

REFERENCES

[1] N. Bronson, T. Lento, and J. L. Wiener, "Open data challenges at facebook," in *ICDE*, 2015, pp. 1516–1519.

[2] *Amazon Simple Storage Service (Amazon S3)*, accessed April 13, 2016. [Online]. Available: https://aws.amazon.com/s3/

[3] *Microsoft Azure Storage*, accessed April 13, 2016. [Online]. Available: https://azure.microsoft.com

[4] *Google Cloud Platform*, accessed April 13, 2016. [Online]. Available: https://cloud.google.com/

[5] *Dropbox Personal Storage*, accessed April 13, 2016. [Online]. Available: https://www.dropbox.com

[6] *Box Personal Storage*, accessed April 13, 2016. [Online]. Available: https://www.box.com

[7] *Apple iCloud*, accessed April 13, 2016. [Online]. Available: https://www.icloud.com

[8] E. Shurman and J. Brutlag, "The user and business impact of server delays, additional bytes and http chunking in web search," *OReilly Velocity Web performance and operations conference*, 2009.

[9] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[10] Y. Xiang, T. Lan, V. Aggarwal, and Y. F. R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *ACM SIGMETRICS PER*, vol. 42, no. 2, pp. 3–14, 2014.

[11] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[12] J. Li and B. Li, "Erasure coding for cloud storage systems: a survey," *Tsinghua Science and Technology*, vol. 18, no. 3, pp. 259–272, 2013.

[13] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *USENIX ATC*, 2012, pp. 15–26.

[14] K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers," *ACM SIGCOMM CCR*, vol. 44, no. 4, pp. 331–342, 2015.

[15] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *OSDI*, 2006, pp. 307–320.

[16] M. Ovsiannikov, S. Rus, D. Reeves, P. Sutter, S. Rao, and J. Kelly, "The quantcast file system," *VLDB*, vol. 6, no. 11, pp. 1092–1101, 2013.

[17] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. Lau, "Scaling social media applications into geo-distributed clouds," *IEEE/ACM TON*, vol. 23, no. 3, pp. 689–702, 2015.

[18] Amazon-AWS, *Summary of the october 22,2012 aws service event in the us-east region*, accessed April 13, 2016. [Online]. Available: https://aws.amazon.com/message/680342/

[19] E. Bauer and R. Adams, "Geographic distribution, georedundancy, and disaster recovery," *Reliability and Availability of Cloud Computing*, pp. 174–182.

[20] W. D. Gray and D. A. Boehm-Davis, "Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior." *Journal of Experimental Psychology: Applied*, vol. 6, no. 4, p. 322, 2000.

[21] A. Jonathan, A. Chandra, and J. Weissman, "Awan: Locality-aware resource manager for geo-distributed data-intensive applications," in *IC2E*, 2016.

[22] K. Le, R. Bianchini, M. Martonosi, and T. D. Nguyen, "Cost-and energy-aware load distribution across data centers," *Proceedings of HotPower*, pp. 1–5, 2009.

[23] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," in *ACM SIGMETRICS*, 2011, pp. 233–244.

[24] I. Narayanan, A. Kansal, A. Sivasubramaniam, B. Urgaonkar, and S. Govindan, "Towards a leaner geo-distributed cloud infrastructure," in *HotCloud*, 2014.

[25] N. Prakash and M. N. Krishnan, "The storage-repair-bandwidth trade-off of exact repair linear regenerating codes for the case d= k= n- 1," in *ISIT*, 2015, pp. 859–863.

[26] C. Tian, "Characterizing the rate region of the (4, 3, 3) exact-repair regenerating codes," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 967–975, 2014.

[27] K. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Annual Allerton Conference on Communication, Control, and Computing*, 2009, pp. 1243–1249.

[28] M. K. Aguilera, R. Janakiraman, and L. Xu, "Using erasure codes efficiently for storage in a distributed system," in *DSN*, 2005, pp. 336–345.

[29] H. Kameyama and Y. Sato, "Erasure codes with small overhead factor and their distributed storage applications," in *CISS*, 2007, pp. 80–85.

[30] J. Li, "Adaptive erasure resilient coding in distributed storage," in *ICME*, 2006, pp. 561–564.

[31] X. Wang, Z. Xiao, J. Han, and C. Han, "Reliable multicast based on erasure resilient codes over infiniband," in *ChinaCom*, 2006, pp. 1–6.

[32] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Distributed data storage in sensor networks using decentralized erasure codes," in *Asilomar Conference on Signals, Systems and Computers*, vol. 2, 2004, pp. 1387–1391.

[33] G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 989–997, 2014.

[34] L. Huang, S. Pawar, H. Zhang, and K. Ramchandran, "Codes can reduce queueing delay in data centers," in *ISIT*, 2012, pp. 2766–2770.

[35] N. B. Shah, K. Lee, and K. Ramchandran, "The mds queue: Analysing the latency performance of erasure codes," in *ISIT*, 2014, pp. 861–865.

[36] G. Liang and U. C. Kozat, "Use of erasure code for low latency cloud storage," in *Annual Allerton Conference on Communication, Control, and Computing*, 2014, pp. 576–581.

[37] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, “The cost of a cloud: research problems in data center networks,” *ACM SIGCOMM CCR*, vol. 39, no. 1, pp. 68–73, 2008.

[38] A. Kumar, R. Tandon, and T. C. Clancy, “On the latency of erasure-coded cloud storage systems,” *CoRR*, vol. abs/1405.2833, 2014. [Online]. Available: http://arxiv.org/abs/1405.2833

[39] Y. Liu, S. C. Draper, and N. S. Kim, “Queuing theoretic analysis of power-performance tradeoff in power-efficient computing,” in *CISS*, 2013, pp. 1–6.

[40] Y. Liu, S. C. Draper, and N. S. Kim, “Sleepscale: runtime joint speed scaling and sleep states management for power efficient data centers,” in *ISCA*, 2014, pp. 313–324.

[41] U. J. Ferner, M. Médard, and E. Soljanin, “Toward sustainable networking: Storage area networks with network coding,” in *Annual Allerton Conference on Communication, Control, and Computing*, 2012, pp. 517–524.