Towards Cost-aware Capacity Provisioning and Load Balancing in Fault-tolerant Distributed Data Centers

Rakesh Tripathi

Towards Cost-aware Capacity Provisioning and Load Balancing in Fault-tolerant Distributed Data Centers

Thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

by

Rakesh Tripathi

Under the Supervision of

Dr. T. Venkatesh

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
Guwahati 781039, India
January 2018

Dedicated to

My parents, wife and son

For their blessings, constant inspiration, and love

Declaration

I certify that

a. The work contained in this thesis is original, and has been done by myself under the general supervision of my supervisor.

b. The work has not been submitted to any other institute for any degree or diploma.

c. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

d. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT Guwahati
Date:

Rakesh Tripathi
Research Scholar
Department of Computer Science
and Engineering,
Indian Institute of Technology Guwahati,
Guwahati - 781039. India.

CERTIFICATE

This is to certify that this thesis entitled " ***Towards Cost-aware Capacity Provisioning and Load Balancing in Fault-tolerant Distributed Data Centers****" being submitted by Mr. Rakesh Tripathi to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, is a record of bona fide research work under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.*

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

Place: IIT Guwahati

Date:

Dr. T. Venkatesh
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati - 781039. India.

Acknowledgements

I would like to take this opportunity to thank many individuals who directly or indirectly supported me during my PhD process.

First, I want to thank my advisor Dr. T. Venkatesh for his guidance throughout my PhD. He has helped me understand what research truly is and constantly motivated me to pursue research. I am thankful for all that I learned from him. I also want to thank my doctoral committee members, Prof. Diganta Goswami, Prof. S. K. Bose, and Prof. S. V. Rao for their valuable comments and suggestions during my PhD.

I am thankful to my host organization, NIT Raipur for allowing me to carry out research work at IIT Guwahati by granting me leave of absence and I thank all my colleagues for their continuous support. A special thanks goes to Prof. Shrish Verma for being so supportive always.

Next, I would also like to express my heartfelt gratitude to the Director, the Deans, and other management of IIT Guwahati, whose collective effort has made this institute a place for world-class studies and research. I am thankful to all the faculty and the staff of the Department of CSE for the support received. I am beholden to my friend and collaborator S. Vignesh, without whose help, motivation and support, the work might not be possible in this duration. I thank all my friends, Bala, Rahul, Amarnath, Hema, Manoj, Akash, Subhrendu, Shounak, Durgesh, Chiranjeevi, Sanjit, Niladri, Panthadeep, Abhishek, Pradeep, Mrityunjay, Rakesh, Sunil, Achyut, Piyoosh, and Rajesh, only to name a few, with whom I have spent most of the time.

My heart-felt regards go to my father-in-law, mother-in-law, sisters-in-law, Neetu , Sudha, Renu, and Nimmi , brothers-in-law, Balram, Atul, Anunay, and Anoop , neices, Kiran, Adhya, Arohi, Aaradhya and Kirti and nephews, Arun and Achyut for their love and support. I would like to express my gratitude to my sisters, Manju and Sanju, brothers-in-law, Ramjeet Mishra and Madhusudan Mishra, nephews Prakhar and Akshat, for their continuous support, love and prayers.

I feel a deep sense of gratitude for my grand mother, mother, father, who formed part of my vision and taught me good things that really matter in life. Their infallible love and support has always been my strength. Their patience and sacrifice will remain an inspiration throughout my life. I am also grateful to my Bhaiya, Bhabhi, neice Akhsita and nephew

Atharva for their love, constant inspiration and encouragement. They were always there for me in good and difficult times.

Finally, I would like to thank my wife, Shashi who has supported, encouraged, and believed in me during this long and rough road that culminated in this dissertation. I would not have had made it this far without her sacrifice and patience. A special note of thanks to my son Avi, who was source of inspiration to work hard, and enjoyment in my life during this journey, all of which helped me become more efficient at my work. They will always inspire me to move forward towards my destination.

Place: IIT Guwahati

Date: **Rakesh Tripathi**

Abstract

Many critical e-commerce and financial services are deployed on geo-distributed data centers (GDCs) for scalability and availability. Recent market surveys show that failures are common in the data centers and this results in a huge financial loss. Designing data centers for high availability includes spare capacity provisioning across the data centers. The work in this thesis addresses the problems of cost-aware capacity provisioning and load balancing in fault-tolerant GDCs (to mask the failures at a site). We propose optimization models for cost-effective planning and operation of the GDCs and propose algorithms for solving these. First, we propose an optimization model to distribute the servers across the GDC such that, the total cost of ownership (TCO) for an operator is minimized. The model identifies the optimal server distribution and optimal request routing policy to exploit the spatio-temporal variation in the electricity prices and user demand for minimizing the TCO. Next, we extend the optimization model for capacity planning in GDCs collocated with renewable energy sources. Using this model, the operators can reduce their carbon footprint by maximizing the green energy usage, while minimizing the TCO. We also extend this model to consider GDCs powered by both brown and green energy sources. In such a case, we use an objective to minimize the total cost while ensuring that a certain percentage of green energy is always used.

In this thesis, we also address another important problem, cost-aware load balancing in large-scale fault-tolerant GDCs. We use game theory to formulate the problem of cost-aware distributed load balancing in GDCs. We use a non-cooperative game executed across a finite number of front-end proxy servers, with an objective of minimizing the linear combination of operating cost and revenue loss due to increased latency. Based on the structure of Nash equilibrium, a distributed load balancing algorithm is proposed. The proposed algorithm is decentralized, has a low complexity, and offers fairness in average latency perceived by the clients. Lastly, we propose a two-stage distributed algorithm for load balancing after the failure of a data center. The proposed algorithm spreads the load of failed data center minimizing the operating cost and then, re-routes the requests considering

the delay and green energy usage constraints. All the proposed algorithms are evaluated using real-world data set.

Results shows that the proposed approaches yield optimal results in planning and operation of fault-tolerant GDCs, powered by both brown and green energy sources. We conclude that it is indeed possible to minimize the cost of running GDCs considering the spatio-temporal dynamics and it is possible to mask single data center failure with no additional cost using the proposed models. We conclude that with a suitable model, green energy integration lowers the cost of designing fault-tolerant GDCs(despite green energy being costlier). Our model works well even with uncertainty in the available wind energy and achieves a significant reduction in the cost as the technology advances. We also show that online load balancing algorithms should be cost-aware in distributing the requests so that, the operating cost is also minimized apart from the latency. We designed an algorithm that ensures delay fairness to the clients without increasing the operating cost.

Contents

List of Figures

LIST OF FIGURES

List of Tables

List of Abbreviations

AWS	Amazon Web Services
CACP	Cost-aware Capacity Provisioning
CDN	Content Delivery Network
CF	Capacity Factor
EAC	Energy to Acquisition Cost
EC	Elastic Cloud
ESD	Energy Storage Devices
GCACP	Green Cost-aware Capacity Provisioning
GDC	Geo-distributed Data Center
GLB	Geographical Load Balancing
GOS	Global Optimal Solution
KKT	Karush-Kuhn-Tucker
LP	Linear Programming
MIDC	Measurement and Instrumentation Data Center
MILP	Mixed Integer Linear Programming
MS	Minimum server model
NBS	Nash Bargaining Solution
NCG	Non-cooperative Game
NREL	National Renewable Energy Laboratory
PPA	Power Purchase Agreement
PS	Proportional Scheme
PUE	Power Usage Effectiveness
QoS	Quality of Service
REC	Renewable Energy Certificates
SLA	Service Level Agreement
TCO	Total Cost of Ownership
VM	Virtual Machine

Chapter 1

Introduction

Geo-distributed data centers (GDCs) have drawn increasing attention from both academia and industry as emerging underlying physical infrastructure for cloud computing or Internet-scale web services. A GDC is collection of small, geographically distributed, fully automated and orchestrated data centers interconnected with transparent transport mechanism like Virtual Private LAN Service (VPLS) or Multi-protocol Label Switching (MPLS), dark fiber or Overlay Transport Virtualization (OTV) [1]. Fig. 1.1 illustrates a typical GDC spanning different parts of the world. These data centers are owned and managed by several large service providers like Amazon, Apple, Yahoo, BOA (Bank of America) and Microsoft, and offer web services, HPC, Map reduce services, banking and financial services, ERP, and other business critical applications. Apart from these large service providers, content providers such as Akamai and Chinacache also invest widely in GDCs. A GDC consists of collection of data centers placed behind front-end proxy servers. The front-end proxies are responsible for forwarding the requests to one of the data centers, according to a load balancing policy.

The main advantages of GDCs are:

- Reduced latency to the clients as their requests are served by nearby data

centers [2].

- High availability: Geo-distribution can safeguard against unplanned outage of entire data center, which component level redundancy like power backup or RAID cannot guarantee and the downtime during planned shutdown is lower [2].
- Migration of applications or data between data centers across space and time is possible to minimize delay, energy consumption or cost [3, 4].
- Local law may prohibit data being taken beyond the territory boundaries.

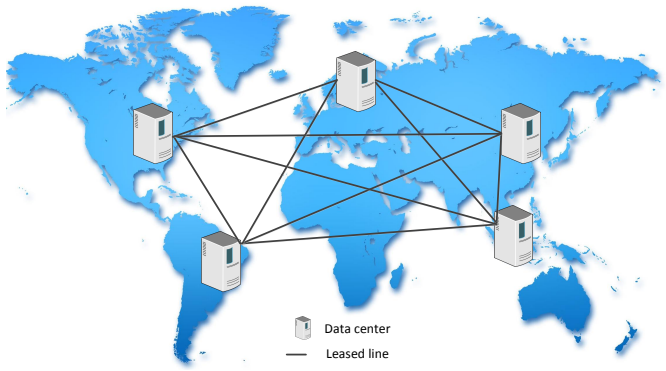


Figure 1.1: Illustration of a typical geo-distributed data center

Typically, two types of workloads are supported by these data centers. First, mice-type/request-response/delay sensitive, where each request of this type has a small transaction size and they have tight response time constraints. For example, web services and e-commerce transactions fall under this category. Second, elephant-type/delay tolerant/batch requests, where each request of this type has a large transaction size and can last for a long time. These are throughput-sensitive like, map reduce jobs, social network analysis, and HPC applications. In this thesis, we focus on web transaction requests (like e-commerce, business and financial transactions), where the response size is very small. We use geo-distributed data

centers (GDCs), distributed data centers or data centers interchangeably in the entire thesis.

Need for high availability: Any critical e-commerce and financial services running on GDCs demand high availability. By high availability we mean that the data center continues to deliver original service after the failure at a single site (may be with slightly degraded performance). A survey by Gartner estimated that 60% of companies incurred a loss to the tune of $250,000-$500,000 for an hour of downtime, and one sixth of the companies incurred a loss of $1 million or more [5]. Another survey by Ponemon Institute reported that the frequency of data center outage (complete or partial) could be as high as once a month with an average duration of three hours. It was reported to cause a loss of $1,734,433 per organization with an average cost of $690,204 per incident [6]. Instances of a data center failure at a site have been reported by many cloud service providers like Amazon, Facebook, and Google [7]. Thus, to prevent huge loss of revenue associated with downtime, designing a fault-tolerant distributed data center is important.

Energy consumption: Past decade has seen sky-rocketing proliferation of web services, which in turn prompted rapid growth in the number and scale of data centers. Due to this, there has been enormous and growing energy demand for data centers. Currently, data centers that power Internet-scale applications consume about 1.3% of the worldwide electricity supply with an estimated growth rate of 12% per year and this fraction is expected to grow to 8% by 2020 [8]. Moreover, large data centers consuming many megawatts of power, pay annual electricity bills to the tune of tens of millions of dollars. Google pays for 1,120 GWh of power $67 M, and Microsoft pays $36 M for power consumption of over 600 GWh [9]. Therefore, for sustainable design of data center it is critical to intelligently address the problem of capping the ever-increasing cost due to power consumption.

Green data centers: Besides the monetary cost, there are also environmental

considerations [10]. High energy consumption not only results in huge electricity cost but also in increased carbon emissions. This is due to the fact that most of the electricity produced worldwide comes from burning coal or natural gas (a carbon-intensive approach). In the United States, generating 1 kWh of electricity emits about $500g$ of CO_2 on an average [8]. In general, each $100MW$ power station costs \$60-100 million dollars to build and emits 50 million tons of CO_2 during its operation [11]. As a result, IT companies currently contribute 2% of global greenhouse gas emissions [12]. Therefore, there is a pressing need for renewable energy integration for sustainable geo-distributed data centers. Renewable energy refers to the energy that is collected from renewable resources like sunlight, wind, rain, and tides., which are naturally replenished on a human timescale [13].

There are many ways these data center operators can reduce their impacts on environment. They can invest in renewable energy by installing or financing new plants either on-site (collocated with data center) or off-site (remote installation), where the energy produced is pumped directly or through electrical grid to the data center. In fact, Google and Apple currently use this approach. Otherwise, they can also contribute in reducing carbon footprint by purchasing renewable energy products like Renewable Energy Certificates (REC) or Power Purchase Agreement (PPA) [14]. The current trend is that some power utilities are allowing large electricity consumers to select a mixture of brown and green energy. Based on the power requirement, the utility promises to pump a sufficient amount of green energy into the electricity grid.

Load balancing: In a typical distributed data center, the client requests are accepted by front-end proxy servers and are further redirected to a suitable data center for processing. This mechanism is known as geographical load balancing (GLB) in the literature [15]. The rules for wide-area workload distribution by front-end elements depend on several mutually interacting objectives such as,

electricity price and cost [4], carbon-footprint [16], and maximizing the use of renewable energy [17], and reducing transmission delay [18]. For example, energy consumption cost could be lowered by forwarding more workload towards a data center where electricity price is cheaper or towards a data center in colder regions. Carbon footprint can be reduced by forwarding requests towards data center where renewable energy is readily available. The problem of load balancing is challenging as these forwarding decisions also need to satisfy several constraints to meet QoS requirements, data center compute capacity, power capacity, and other service-level-agreements (SLAs).

As the electricity prices, green energy availability, and demand vary substantially during the day and across geographic regions, the workload processing cost across all the data centers does not remain the same. This spatio-temporal variation provides ample opportunity in distributed data centers to intelligently route requests to locations, where the electricity is cheaper, renewable energy is abundant, or efficient utilization is possible owing to demand multiplexing. This can help data center operators to significantly reduce their operating cost or carbon footprint.

In this thesis, we focus on leveraging such spatio-temporal variation, in minimizing the total cost of capacity provisioning and load balancing in fault-tolerant GDCs. We address the problem of spare capacity provisioning, which involves allocation of additional servers across different data center sites, so that the total cost of ownership (includes capital cost and operating cost) is minimized while satisfying a set of constraints like client latency, demand, and green energy bound. It is also important to decide online load balancing policy in the distributed data center, which is efficient and cost optimal. For this the load balancing algorithm should be intelligent enough to exploit spatio-temporal variation in the energy cost, available green energy, demand, and available compute capacity (even after the failure of a data center).

1.1 Motivation of the Research Work

Several organizations delivering services using GDCs demand high availability because of huge loss of revenue, cost of idle employees, and loss of productivity associated with downtime. For high availability, mitigating disaster risks to applications and data is the ultimate goal, which can be achieved by having multiple data centers, with sufficient reserve capacity [19]. Designing a fault-tolerant GDC usually involves spare capacity provisioning (allocation of additional servers to mask the failure) across different data center sites such that failure of data center (at a site, at any time) could be tolerated while satisfying client latency requirement and demand. Spare capacity requirement across a GDC to mask failure at a single data center site is explained by a simple example. Consider a distributed data center with 5 sites with a compute capacity of 20 units at each site. To mask the failure of any one data center at a time, we require a spare capacity of 5 units at each of the remaining data centers. Therefore, the total spare capacity required is 25 and the cost in building a fault-tolerant data center that can mask single failure increases by 25%.

Considering operating cost in capacity planning: Next, we describe the motivation behind considering the operating cost in spare capacity provisioning for data centers giving an example. Let us consider the cost of a server to be $2000, and its lifetime to be 4 years [20]. We calculate the energy to acquisition cost (EAC) defined to be the ratio of cost of running a server for 4 years to its acquisition cost, as given below in Eqs. (1.1) and (1.2).

$$\text{Power cost} = 4 \text{ years} * (8760 \text{ hours/year}) * (\text{electricity cost}) * \text{server power} * \text{PUE} \tag{1.1}$$

$$\text{EAC} = \frac{\text{power cost}}{\text{server cost}} * 100 \tag{1.2}$$

In the above equation, power usage effectiveness (PUE) of a data center is defined as the ratio of the total power entering the data center to the power used by the computing equipment.

Using the electricity prices from [21–24], an average value for server power consumption as 300W, and the PUE as 1.5, we obtain the EAC values as listed in Table 1.1. The EAC values indicate that for most of the countries, the cost of power and cooling exceeds the cost of buying servers. This suggests that greater attention should be put on optimizing data center power consumption cost instead of only minimizing the number of servers while designing fault-tolerant data centers.

Country/City	Electricity Price($/kWh)	Cost(in $)	EAC
Canada	0.06	946	47
Oregon, USA	0.06	946	47
Virginia, USA	0.07	1104	55
Switzerland	0.07	1230	62
Netherlands	0.09	1419	71
Japan	0.10	17	84
California, USA	0.12	1971	99
Ireland	0.13	2050	103
UK	0.13	2050	103
Hongkong	0.17	2680	134

Table 1.1: EAC for different countries

The main challenge in designing fault-tolerant distributed data center is to provision spare capacity so that the total cost of ownership (TCO), which includes capital cost (cost of spare servers) and operating cost is minimized while satisfying the client latency even after the failure. Note that cost-aware capacity provisioning is a variant of facility location problem known to be in NP-hard [25].

Considering data centers with collocated renewable energy sources: Rise in the scale and number of data centers not only results in high energy cost but also in carbon emissions. Renewable energy has matured enough to become utility power and maximizing on-site renewable energy usage by collocating renewable energy sources along with data center sites is a cost-effective solution [11]. Apple has used $20MW$ solar array for its data center [26]. Even though renewable energy availability is known to be highly intermittent, it turns out that renewable energy actually becomes more predictable as the number of renewable energy generators connected to the grid increases. This is due to the effect of geographic diversity and the *law of large numbers i.e.*, large number of geographically distributed renewable energy generation units installed lead to renewable energy being more predictable with certain degree of accuracy [14,27,28]. The authors in [27] established that it is possible for distributed data centers to exploit uncorrelated wind sources to satisfy 95% of energy requirement following a wind power-aware routing policy. Integrating renewable energy sources into projects are also driven by the growth of corporate social responsibility program, government imposed caps on carbon emission as well as government incentive associated with carbon neutrality. Therefore, the operator must intelligently provision compute capacity which minimizes the operating cost and at the same time maximizes the available green energy usage, while satisfying a given set of constraints.

Imposing green energy usage requirements for GDCs: Multiple options of greening data centers exist like, data centers collocated with renewable energy sources, using off-site renewable energy sources, where data centers are powered through renewable energy generated at remote locations, and indirect renewable energy options like, REC and PPA [14]. In this thesis we consider two renewable energy usage models for designing green data centers. First, maximize green energy freely available from collocated green energy sources. Second, purchasing green

energy so that a certain amount of total energy is procured from renewable energy sources. Owing to huge upfront installation cost associated with renewable energy sources, major operators have developed an interim goal for its 100% renewable energy commitment and set a target of partial renewable energy integration annually [29]. For example, Facebook aimed to be 25% green by the end of 2015, and met the benchmark [30]. Therefore, it is a good idea to start with partial renewable energy bound. That is, out of the total power consumed by data center at least a fraction of energy should come from renewable energy sources which is termed *"green energy bound"* [31]. Hence, it is crucial for data center operators to consider green energy cost apart from brown energy cost while satisfying the green energy bound. At the same time, other set of constraints like latency bound, demand, and availability requirements should also be satisfied.

Cost-awareness and latency fairness in distributed load balancing: Load balancing in fault-tolerant distributed data center is another important issue, both from perspective of data center operators and end-users as well. While operators' objective is to minimize operating cost complying to latency and availability requirements, the end-user would like to minimize the response time. Each data center is characterized by spatio-temporal variation in the electricity price, renewable energy availability, and failure percentage. The load balancing mechanism that distributes load leveraging these factors and minimizes the operating cost is profitable for the operators, but ignores the users' perspective. Users consuming the same resources may pay the same price, but experience variable delay. Ensuring fairness in service latency across the requests from different clients is also important. The load balancing algorithms in the literature designed to provide fairness, did not consider the energy cost. Therefore, it is important to consider the linear combination of operating cost (or energy cost) and revenue loss due to latency (including the network and queuing delays) as the objective function in designing

the load balancing policy. Addition of revenue loss model due to latency in objective function, captures the notion of fairness in latency perceived across the clients.

Most of the work in literature on load balancing in distributed data center considers solving proposed solution centrally. Few works proposed decentralized algorithm for load balancing but their drawback was that either they did not consider operating cost or that the proposed solution approach is computationally expensive. Therefore, designing decentralized algorithms for better performance, and scalability is important for large-scale distributed data centers.

Efficient failure detection and recovery in distributed load balancing: Most of the popular load balancing approaches discussed in the literature [15] use some sort of a central controller that solves an optimization problem and pushes the policy for load balancing to all front-end proxy servers and capacity provisioning information to the data centers. This could be useful when the controller has to solve the optimization formulation periodically (say every hour), when the system state information (like electricity price, available green energy, and demand) remains fixed for the entire duration [32]. But in fault tolerant systems, central controllers or front-end proxy servers have to periodically send heartbeat messages to data centers in local and remote locations detect the failure (either partial or complete). This probing could be expensive in terms of message and time complexity [33]. Hence, there is need for efficient failure detection and recovery mechanism for distributed data centers.

1.2 Contributions of the Thesis

Based on the several motivation factors mentioned so far, we formulated a set of problems for cost-aware capacity provisioning and load balancing in fault-tolerant GDCs. We briefly describe the five problems addressed in this thesis. For each problem, we discuss the formulation, broad solution approach, and mention the

key observations from our evaluation. The details of these are presented in various chapters of the thesis.

1.2.1 Cost-aware Provisioning of Spare Capacity for Fault-tolerant GDCs

Problem statement: What is the optimal server distribution across various locations such that, the failure of any data center at a site is masked while the total cost of ownership (TCO) is also minimized?

Designing a fault-tolerant GDC usually involves spare capacity provisioning (allocation of additional servers to mask the failure) across different data center sites, satisfying a set of constraints related to handling the demand at each location, and satisfying the delay requirement of the clients. The objective to guarantee high availability, where the system continues to provide the same level of service even after the failure should not increase the TCO for an operator significantly.

We formulate the problem of cost-aware capacity provisioning (CACP), in fault-tolerant distributed data centers using mixed integer linear programming (MILP), with an objective of minimizing the TCO. The model considers the heterogeneity in client demand, various data replication strategies (single and multiple site), spatio-temporal variation in electricity price and demand, while computing the spare capacity distribution across the locations. The outcome of solving the optimization model gives the optimal number of servers across the sites and optimal request allocation to the data centers. We solve the CACP model using real-world data and compare the TCO obtained with two other models: one that minimizes only the total number of servers (MS model), and the other that minimizes the average response time (CDN model). The proposed CACP model results in significant savings in the TCO compared to the existing models. In the following, we summarize the key conclusions from the numerical evaluation of the CACP model.

Key Observations:

- The CACP model provisions more servers but reduces the TCO up to 35% compared to the MS model, and up to 43% compared to the CDN model. The proposed model is observed to be better than the existing models due to its ability to multiplex demand considering the spatio-temporal variation in the electricity prices and the demand.

- Numerical results demonstrate that the approach of minimizing the TCO is beneficial when the electricity price varies significantly, which appears to be the case for most of the cloud providers operating GDCs.

- Our model is also useful to study the effect of the replication cost on the TCO for planning distributed data centers. We show that it is possible to achieve availability against single data center failure with no additional cost using the CACP model. We show that the contribution of the replication cost to the TCO, in multi-site replication is significantly high, when the number of data centers increases.

- The CACP model is cost effective when the latency requirement is not stringent and a data center does not operate at its peak utilization. Under heavy load, the CACP model can help the provider determine an optimal data center upgrade plan while minimizing the TCO.

- We also prove that the CACP problem for the design of a fault-tolerant distributed data center is NP-hard.

1.2.2 Capacity Planning in Fault-tolerant GDCs Collocated with Renewable Energy Sources

Problem statement: For designing a fault-tolerant distributed data center collocated with renewable energy sources, how should the servers be distributed to minimize the total cost while maximizing the usage of green energy available?

While the existing models minimize the number of servers, we seek to use more green energy for data centers to reduce brown energy consumption. In our system model, we consider data centers collocated with multiple green energy source. We extend our previous CACP model and propose new optimization model (termed GCACP) for spare capacity provisioning, which distributes the servers across the sites to minimize the TCO while maximizing the usage of freely available renewable energy. In this, we consider the variation in demand, battery storing surplus green energy, variability in green energy availability, and renewable energy sell-back revenue. To understand the advantage of considering the operating cost in spare capacity provisioning while maximizing the available renewable energy, we compare the cost of solution obtained using our model with that of two other models (MS and CDN). To compare against the two models, we extend them with the same set of constraints as GCACP model. After solving both the models to arrive at the server and load distribution, we use the same cost factors to calculate the TCO in each case.

Key Observations:

- We demonstrate that the proposed model reduces the TCO by 48% compared to CDN model (minimize latency), and by 24% compared to the MS model (which minimizes server cost). We see that the TCO for GCACP model reduces with increasing number of data centers due to demand multiplexing as well as increased options for leveraging variations in the electricity price and

renewable energy available.

- With relaxed latency bound, we notice that GCACP model achieves a reduction in the TCO of upto 29% and 52% with respect to MS and CDN models, respectively. Due to the choice in the data centers capable of serving a client region, we see better multiplexing of resources, exploitation of variation in the green energy and electricity prices. We conclude that under relaxed latency constraints GCACP model is more beneficial.

- We also observe that GCACP model is advantageous when green energy and electricity prices vary significantly across data centers in time, which appears to be the case in most of the GDCs.

1.2.3 Optimizing Energy Cost in Fault-tolerant GDCs Satisfying Green Energy Bound

Problem statement: How should spare capacity be distributed across the data centers powered by a combination of brown and multiple renewable energy sources, while the data center operators try to meet a target green energy bound at minimal cost?

In this problem, we seek to exploit the green energy available to reduce brown energy consumption. We extend our model by considering that data centers can have on-site renewable energy generation (considered in previous problem) or get the power from a combination of green and brown energy sources through the utility apart from collocated renewable energy sources. We model the problem of capacity planning as optimization problem (termed as GACED) with the objective of minimizing power consumption cost (which includes brown energy and green energy cost) for fault-tolerant data center while satisfying minimum bound on green energy usage requirement. For comparison, we designed a baseline model (termed

CED-B) that minimizes the TCO, where the data centers are powered only with brown energy while retaining other constraints.

Key Observations:

- For full data center site failure, percentage gain in the TCO using GACED model compared to the CED-B model is 2%. The gain reduces with increase in green energy usage because, our model increases the amount of (expensive) green energy purchased to meet the constraint. On the other hand, CED-B model has no cost from green energy usage. Hence, with the GACED model, unless we target high renewable energy usage, greening can be achieved with very little or no extra cost.

- We observed that the GACED model can lead to greener data center deployment with no or little additional cost (though green energy is costlier). In particular, we found that at 40% green usage bound, the TCO is almost same for both the models (GACED and CED-B) due to the fact that, the GACED model optimally uses cheaper renewable energy to reduce the TCO.

- Solving our model shows that spare capacity provisioning while considering green energy integration, not only lowers carbon footprint but also reduces the TCO.

- If the forecasted green energy availability is inaccurate, the GACED model has only 3% higher TCO with 20% green energy bound.

- Our model works well with improvement in technology and higher capacity factor (*i.e.*, lower renewable energy cost).

1.2.4 Game-theoretic Model for Load Balancing in Fault-tolerant GDCs

Problem statement: How to design a distributed load balancing algorithm which provides better fairness to the clients (in terms of service latency) without increasing the operating cost in fault-tolerant GDCs?

We formulate the load balancing problem in GDCs as a non-cooperative game executed across a finite number of front-end proxy servers. The objective of the game is to minimize the sum of the energy cost and the revenue loss due to delayed service. We consider the spatio-temporal variation in the electricity price, the offered load, and the availability in the model. In the game, each front-end proxy tries to minimize the cost while satisfying the demand constraint. We prove that the objective function is convex and the first order KKT condition is necessary and sufficient for optimality. This closed form solution is termed *best reply* and it provides a minimum cost for that player, given the other players' load balancing strategies. The Nash equilibrium exists for our game because the proposed objective function is continuous, convex and increasing [34, 35]. At the Nash equilibrium, the strategy profile is such that every player's load balancing strategy is a *best reply* given the other players' strategies. For the characterized Nash equilibrium, we propose a distributed algorithm to compute the same. We also perform the complexity analysis of our proposed algorithm. We compare the performance of the proposed algorithm with the existing approaches. The execution of this algorithm is restarted periodically when the data center system parameters (*e.g.*, electricity price, demand) change or a failure occurs. Once the equilibrium is reached, the front-end proxies continue to use the same strategy and the system remains in equilibrium until a new execution is initiated. We evaluate the performance of the non-cooperative game theoretic algorithm (abbreviated as NCG) along with the existing ones, such as the proportional scheme(PS) and the global optimal scheme(GOS), using real-world

data.

Key Observations:

- For all scenarios like, increasing the system size, varying the demand, latency, and latency weight factor numerical results demonstrate that the solution achieved by the proposed algorithm (NCG) approximates the global optimal solution in terms of the cost.

- The main advantage of NCG algorithm is that it is decentralized, it has a low complexity (close to the optimal GOS), and it offers fairness and good average latency across all the client regions.

1.2.5 Distributed Failure Detection and Efficient Load Balancing in Fault-tolerant GDCs

Problem statement: How to devise a data center-initiated load balancing algorithm so that failure could be detected early and corresponding updated load balancing policy can be determined quickly?

We assumed in the NCG algorithm that efficient failure detection mechanism is in place. The problem of load balancing becomes more challenging when it has to consider failure of data center (either partial or complete), since it can happen at anytime and with any frequency [6]. When front-end proxy sends keep-alive message periodically for health checking, it could be computationally expensive in terms of message and time complexity. Therefore, load balancing strategy has to be determined, which takes into account the over-provisioned servers, renewable energy usage requirement, and the power consumption cost to minimize the TCO.

For a scalable, fault-tolerant load balancing system, we propose a data center-initiated distributed load balancing approach to ensure post-failure QoS while minimizing the TCO. Our approach makes an early attempt to balance the load

on a failed data center to mask the failure with a marginal increase in the operating cost. We model the problem of load balancing in fault-tolerant data centers using linear programming (LP) to optimize both the cost of energy consumption and the client latency. For scalability, we also propose a two-stage distributed algorithm based on greedy heuristic method. In the first stage, we propose an algorithm to distribute the load on a failed data center across the remaining ones targeting minimal increase in the operating cost. Next, the min-cost network flow model is used to derive an optimal request mapping policy, where the cost function considers the propagation delay to account for QoS requirements.

Key observations:

- For all scenarios like varying number of data centers, demand, and failure percentage, the proposed algorithm achieves same TCO as the centralized global solution.

- Analysis shows that the proposed algorithm has low computational complexity, yet exactly matches the cost obtained using global optimal solution (which is a centralized approach).

1.2.6 Organisation of the Thesis

The rest of the thesis is organized as follows: In the next chapter, we present the background material required to understand the setting in which we addressed the problems discussed. We also present the state-of art literature on cost-aware data center placement, capacity provisioning, and load balancing in GDCs. In Chapter 3, we addressed the problem of cost-aware spare capacity provisioning in GDCs capable of masking single data center failure. We propose an MILP model to reduce the TCO in spare capacity provisioning. In Chapter 4, we present an extended model that considers the data centers collocated with green energy sources. Here, we determine

the optimal server distribution that minimizes the TCO while maximizing the usage of available green energy. In Chapter 5, we consider data centers powered by multiple on-site and off-site renewable energy sources, for which we propose a model for optimal server distribution to minimize the total cost of energy consumption (which includes brown energy and green energy cost), while meeting the requirements for the green energy usage. Next, we proposed two distributed load balancing algorithms for fault-tolerant GDCs with an objective function to minimize the TCO while meeting the delay constraints. In Chapter 6, we used non-cooperative game theory to model the load balancing problem. The model is designed to ensure that there is fairness in the latency perceived by the clients while minimizing the operating cost. In Chapter 7, we propose a data center-initiated distributed load balancing algorithm to ensure post-failure QoS while minimizing the operating cost. Finally, the thesis ends with summary and future work in Chapter 8.

Chapter 2

Background and Literature Survey

In this chapter we present some background material regarding the architecture of the GDCs we consider in this thesis. We discuss the popular model to ensure high availability in GDCs and the efforts to minimize the operating cost. We also present the current models used for greening the GDCs, using renewable energy sources and reducing the carbon footprint. We also show how demand multiplexing can be exploited in minimizing the operating cost of GDCs. Finally, the chapter discusses the state-of-art literature related to cost-aware spare capacity provisioning and energy cost-aware load balancing in GDCs.

2.1 Architecture of a GDC

A typical GDC has front-end proxy servers and back-end data centers as shown in Fig. 2.1 [36]. The front-end proxies inspect incoming requests from clients and transparently direct them to the appropriate back-end data centers that can serve them. While there is a logical separation between the front-end and back-end, they can be physically in the same data center. The data in back-end data centers are usually replicated across multiple data centers (typically two or three), so that there

is high performance and the data center or network failures can be tolerated [36]. The number of mirrors is typically smaller because, replicating content across many sites would increase state-coherence traffic without a commensurate benefit in the availability or performance. The front-end proxies spread the load based on policies defined to meet certain objectives like, minimizing the service latency or minimizing the operating cost. Load balancing is implemented using different protocol level mechanisms like DNS/HTTP redirection and persistent HTTP proxy connections.

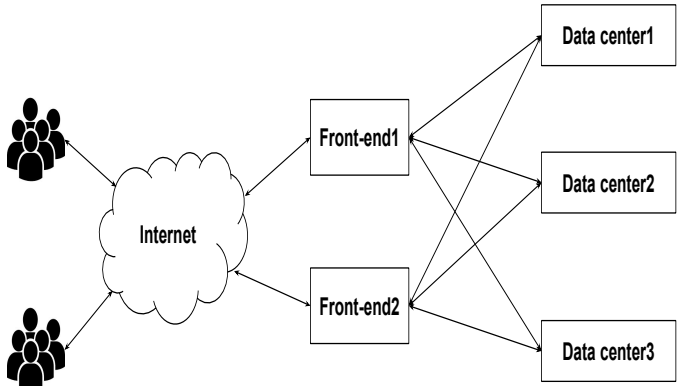


Figure 2.1: Architecture of a typical geo-distributed data center

2.2 High Availability Requirement

The failures in a data center leading to degraded performance could lead to huge financial loss for any organization. According to a report by Fierce Enterprise Communications, downtime of large data centers costs businesses $26 million dollars each year in North America alone [37]. A survey by Ponemon Institute reported that 59% of Fortune 500 companies experience at least 1.6 hours of downtime a week, which is about 83 hours an year [6]. Data center failures can be due to reasons such as, building fire, power outage, human errors, software bugs, Internet attacks, hardware failures, and natural disasters among which, 70% are reportedly due to

human errors [38]. There are many instances of complete data center failure in the recent past. For example, because of a software bug Amazon suffered an outage during which many companies using Amazon Web Services (AWS) EC2 instances suffered loss of service. The failure lasted more than 3 days after which some data were permanently lost [39].

The amount of downtime acceptable for different customers depends significantly on their services deployed on data centers. For example, some may survive being off-line for a day and later restore all the data from the backup site. For large organizations offering e-commerce or financial services, downtime of a minute lead to huge loss in revenue and therefore, the service needs to be delivered from an alternate site within a minute or almost instantly with no noticeable disruption. The most common strategy to counter failures in a distributed system is based on redundancy. In redundancy-based approach, critical system components are backed-up with spare components. Provisioning spare capacity ensures that the system is available even after the failures and continues to deliver services with same or degraded performance.

Generally in a GDC, if data center at a site fails, other data center housing sufficient spare capacity can be used to serve the load of the failed data center and thus mask the failure. We consider this approach of provisioning spare capacity across the sites to mask the unavailability of the data center at any site. We assumed that the failure of the data center at a site is an independent process, i.e., simultaneous failure of data center at more than one location is rare. This holds because the data centers at different locations are not susceptible to common disaster-like situations [40]. For example, the power outage, building fire or any natural calamity at one data center site does not affect the functioning of other data centers. This is done by choosing two locations sufficiently farther apart so that each data center is operationally isolated from the other one. Further, all the data

centers are designed in identical fashion so that, any site is capable of serving any request and there is sufficient spare capacity for fail-over operation.

2.3 Energy Cost Components and their Dynamics

This section gives a brief introduction to the various sources of energy powering GDCs, their costing models and the important dynamics that can be used to minimize cost incurred due to power consumption.

2.3.1 Brown Energy Pricing

Utilities typically offer their customers three types of contracts [9]: (i) fixed, where the customer has to pay fixed price throughout the usage; (ii) time of use (TOU), where the price varies with time of the day; (iii) dynamic pricing, where the customer pays variable price based on the supply and demand. In TOU pricing, for example, there might be two prices, one for 8 AM to 8 PM of weekdays (called on-peak price), and another for all the other times (called off-peak price). Dynamic pricing is further classified into three types depending on how far ahead the price is set: day-ahead pricing, hour-ahead (or 30 minutes in the UK), and real-time pricing (set 5 minutes ahead). The fact that electricity prices vary substantially during the day (because of dynamic pricing) and across geographic regions (because of difference in demand and supply and/or time zones), means that intelligently routing requests to locations where electricity is cheaper can help the operators significantly reduce their power bill.

2.3.2 Renewable Energy Sources and Cost Model

Most of the electricity produced across the globe comes from burning coal or natural gas (a carbon-intensive approach). To avoid consuming brown electricity, GDC

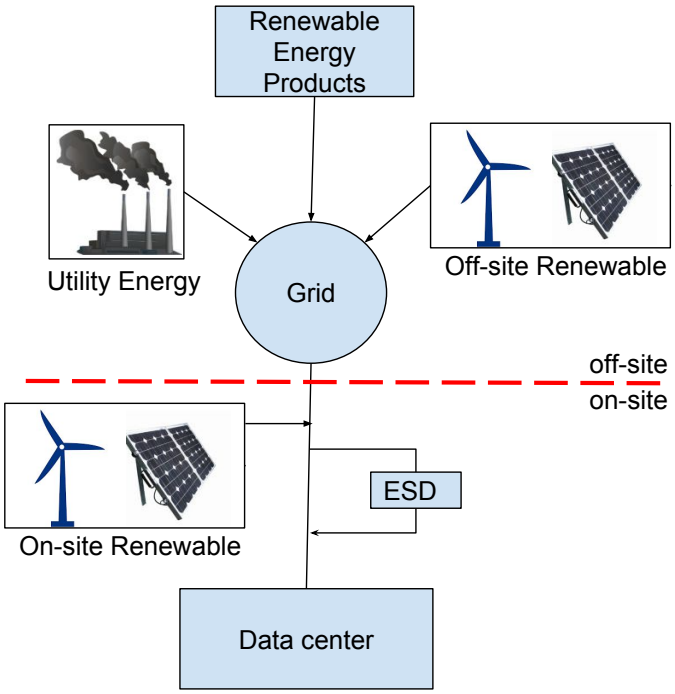


Figure 2.2: Various renewable energy options

owners can use renewable or green energy sources such as solar cells, wind mills, geo-thermal and bio-mass plants. These energy sources reduce reliance on fossil fuel thereby, minimizing carbon dioxide emissions and other greenhouse gases which contribute to global warming. Various renewable energy options available to a data center are depicted in Fig. 2.2. There are different ways of incorporating renewable energy power into their overall energy portfolio as discussed below.

- **On-site renewable energy generation:** In this model, renewable energy sources such as solar panels and wind turbines, are collocated with the data center as shown in Fig. 2.2. They are used to supply some of the power required at the data center facility. Many major companies use some type of on-site renewable energy generators to provide power to their facilities. For example, Apple uses 200MW solar array power plant for its data centers [41].

- **Off-site renewable energy generation:** This involves installation of renewable energy plants at a location with more abundant renewable energy generation potential. The power generated by an off-site source is transported through grid to a data center located at any remote location. The grid essentially acts as the "carrier" of the energy produced for which the utility charges the data center a fee termed as *wheeling charge* [14]. The data center is given incentive for its contribution to greening via some form of discount in its utility bill.

- **Implicit renewable energy products:** Apart from direct installation and provisioning of renewable energy generators by data center operators, there are multiple ways to implicitly incorporate renewable energy into their energy portfolio. In these options, data centers avail renewable energy by paying for it in some fashion as discussed below:

  - Renewable energy certificate(REC): Also know as *green certificate*, it is a market-based instrument to promote renewable energy and facilitate compliance of renewable purchase obligations (RPO). It is aimed at addressing the mismatch between availability of renewable energy and the requirement of the obligated organization to meet the RPO. Purchasing an REC allows the operators to claim that the corresponding portion of its overall energy consumed was *green*. Finally, installation of renewable energy generators not only permits data center operators to achieve greener data centers but also enables them to derive RECs directly through an audit/accreditation process, which the data center operators can choose to sell to the REC market [42].
  - Power purchase agreement (PPA): It involves a contract between a consumer and a renewable energy producer which allows the consumer to purchase a portion or all the electricity generated by the producer at

a negotiated price for which it accumulates some form of credits such as RECs [9, 14].

Carbon offsetting target: A carbon offset is a reduction in emission of carbon dioxide or greenhouse gases to compensate for, or to offset an emission made elsewhere [43]. In recent years, numerous carbon cap policies and regulations have been introduced world-wide. They are either government-mandated, utility-imposed, or voluntary. For example, European Union Emissions Trading System (EU ETS) has imposed cap on EU members' national carbon emission volume. In these countries large carbon emitters who fail to offset or reduce their carbon footprint face heavy penalty. Further, alternative forms of carbon pricing also exist like carbon tax, which is an environmental tax levied on corporate carbon footprints. Apart from taxing, government also provides incentives to set up renewable energy generation plants. For example, federal and state incentives in New Jersey reduced the capital costs of deploying renewable energy plants by 60 percent [11].

Renewable energy cost model: The generating cost of renewable energy seems to reduce over the time due to the technological improvements in the equipment efficiency and increasing deployment. In particular, power generation efficiency and reduction in cost/Watt of renewable energy will reduce the deployment cost significantly in the future. For example, the efficiency of solar panels is expected to triple and the cost/Watt of solar panels is expected to be halved by 2030 [11, 44]. Further, a report from Lawrence Berkeley National Lab found that the average capacity factor among projects built in 2014 reached 41.2 %, compared to an average of 31.2% for the period 2004–2011 and just 25.8% for the period 1998–2003 [45, 46]. Similarly, from 2006 to 2014, world-wide average photo-voltaic solar cell prices have dropped by about 78% [47]. These can promote the usage of green energy over long term. Installation cost and capacity factor are used to calculate the green energy

cost for every hour (h) (in $/Kwh) as defined below [14]:

$$Cost(\$/kWh) = \frac{Installation\_Cost(\$/KW)}{Total\_Lifetime(in\ hr) * Capacity\_Factor(h)} \tag{2.1}$$

where, $Capacity\_Factor(h)$ is the ratio of the actual power output during an hour to the maximum potential output during the hour, when operated at full-rated capacity.

2.3.3 Demand Multiplexing for Improving Utilization

Under the condition that the requests from different client regions are independent and negatively correlated, and the infrastructure is provisioned for peak requirement, multiplexing of demand from different client regions onto data centers (lying in different time zone) results in higher utilization. The resultant under-utilized resources could be used to make up for a capacity loss in failure. This can be quantified formally as discussed below [48].

At a data center indexed i, let $D_i(t)$ be the load at time t,(for $0 < i < N$ and $0 < t < T$), P_i be the peak load, defined as $\max(D_i(t))$, μ_i be the mean load, and σ_i be the standard deviation of the load. Let the demand from n traffic generating sources indexed by i be served at a data center with the aggregate load being $\sum_i D_i(t)$. Then,

$$\begin{aligned} \mathrm{E}\big[\text{load}\big] &= \mathrm{E}\Big[\sum D_i(t)\Big] \\ &= \sum \mathrm{E}\big[D_i(t)\big] \\ &= \sum \mu_i \end{aligned} \tag{2.2}$$

Effective coefficient of variation after multiplexing is

$$\frac{\sqrt{\sum \sigma_i^2}}{\sum \mu_i} \tag{2.3}$$

It can be observed that when the mean and standard deviation are same for all traffic sources, the coefficient of variation reduces by a factor of $\frac{1}{\sqrt{n}}$ *i.e.*, peak to mean ratio

in aggregated traffic is minimized which ensures better resource utilization [48]. In general, multiplexing demand from multiple sources will reduce the coefficient of variation, when the demand is not correlated or simultaneous peaks do not occur. This perfectly suits workload in GDCs which exhibits diurnal pattern due to different time zones. The demand seen across two client regions as shown in Fig. 2.3, demonstrates that the peaks are shifted in time and do not coincide in time. Now we argue that intelligent demand multiplexing improves resource utilization and the under-utilized resources could be used to mask failure. This could be quantified as discussed below.

Spare capacity for fault-tolerance: Let the demand at two data centers be denoted by the random variables $D_1(t)$ and $D_2(t)$, with mean μ_1 and μ_2. The total demand at any point of time is given by $D_1(t) + D_2(t)$ and the expected load is given by

$$\begin{aligned} \mathrm{E}\big[\text{load}\big] &= \mathrm{E}\big[D_1(t) + D_2(t)\big] \\ &= \mathrm{E}\big[D_1(t)\big] + \mathrm{E}\big[D_2(t)\big] \\ &= \mu_1 + \mu_2 \\ &< P_1 + P_2 \end{aligned} \tag{2.4}$$

The workload from interactive applications is dynamic with high peak to mean ratio and the average utilization of a server is typically around 18% [10]. The spare/excess capacity can be used for masking failures by intelligent request routing [49]. This can be illustrated using a simple example. Fig. 2.3 depicts the real-time work load in two data centers at New York and Oregon, obtained from [2]. Let the capacity of each data center be sufficient enough to meet the peak demand. It can be seen that the normalized aggregate compute capacity required at both locations is 1.45, when each client region is served by a data center and it is provisioned in accordance to its peak load. However, intelligent request routing of work load from Oregon to New York would require a maximum aggregate compute capacity of 1.13 *i.e.*, 22%

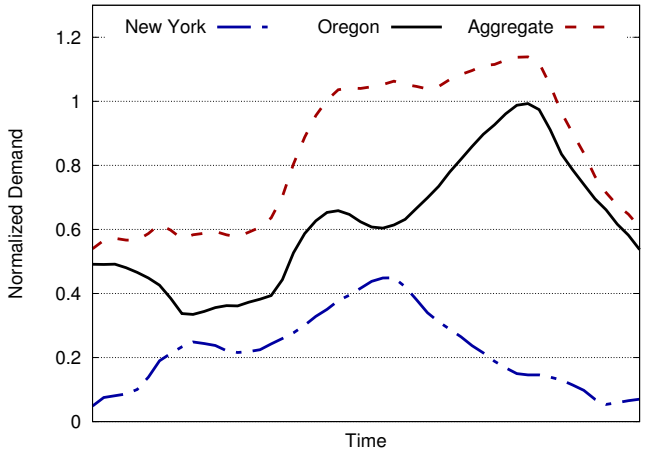


Figure 2.3: Variation in the demand across regions

of compute capacity can be freed by intelligent routing that can be utilized to mask failure.

2.3.4 Geographical Load Balancing

As discussed earlier, the front-end proxy servers map the client requests to the appropriate data centers. GLB defines the request distribution policy to be actuated by the front-end proxies in GDCs. It also decides the number of servers to be activated across data centers (in power saving scheduling mechanisms). The problem addressed in intelligent GLB is to determine routing and capacity provisioning across data centers with the primary objective of minimizing the operating cost or minimizing the average service latency. The challenge in this is to leverage spatio-temporal variation in the electricity price, available green energy, replication cost, and considering demand multiplexing under a predicted demand pattern.

In general, a load balancing strategy can be classified as static, semi-static or dynamic. In the static approach, all the information necessary for the decision making is available before the execution of the algorithm and it remains constant during the execution [50]. In the semi-static approach, the required information is

available at the beginning of each time step or a well defined point [51]. For example, the load on the system and the cost of serving the same is available with reasonable accuracy just before the time slot. In the dynamic approach, the information is not known till the point of execution and it might change during the course of execution [52]. Further, depending upon the way this computation is performed, load balancing algorithms can also be classified as centralized or decentralized. In the centralized approach, one node in the system collects the information necessary to decide the strategy for load balancing. For example, central node could be responsible for solving load balancing algorithm or framework and pushes the load balancing policy to nodes responsible for request routing. In the decentralized approach, multiple nodes participate to decide the load balancing strategy, either cooperatively or independently. Decentralized approach (cooperative or otherwise) is resilient and scalable compared to the centralized one, particularly for large-scale GDCs.

2.4 Related Work

In this section, we summarise some important papers that deal with cost-aware data center placement and capacity provisioning, and energy cost-aware load balancing in GDCs. We present only the literature that considered various approaches in minimizing the operating cost of data centers. To the best of our knowledge, there are very few papers that consider these problems in the context of fault-tolerant GDCs.

2.4.1 Data Center Placement and Capacity Provisioning

The location of data center has direct impact on service response time, capital and operational cost. Selecting a location involves many important considerations,

including its proximity to population centers, power plants, and network backbones; the source of the electricity in the region; the electricity, land, and water prices at the location; and the average temperatures at the location. As there can be many potential locations and many issues to consider for each of them, the selection process can be extremely involved and time-consuming.

The authors of [16], studied how to select an optimal subset of data center locations from a probable set, and how many servers should be deployed at each data center to satisfy the customer QoS requirements. The costs considered were electricity cost and bandwidth cost between data center and client locations. Three objectives were used namely, to minimize total carbon footprint, to minimize total cost with carbon tax and to minimize average service latency. However, the land cost, infrastructure cost and inter-data center latency were not considered and failure of data center was not considered. Similarly, in [20], an optimization framework for selecting additional data center locations to upgrade the capacity of an existing data center was proposed. The objective was to minimize the TCO subject to response time, consistency and availability constraints. The cost factors considered in computing the capital cost were: land, data center construction, transmission line to nearest power grid, OFC line to nearest network backbone, cooling infrastructure, and internal networking. For the operating cost the factors were: electricity, bandwidth, cooling the data center, carbon tax, and administration cost. The work in [53], addressed how to maximize the profit while building a new data center or expanding the existing data center (increase the number of servers) to meet the increasing demand. Annual inflation rate was also taken into account while computing the cost of bandwidth, cooling infrastructure, electricity and the revenue generated. An optimization framework was proposed that takes as input, the current data center locations along with the number of servers and suggests the best option to maximize revenue (either building new data center or expanding existing data

center). The authors of [18] proposed an optimization framework for data center placement/capacity provisioning and request flow control/resource allocation in a joint manner. The objective was to minimize the average cost for new and upgrading data centers in the network subject to a maximum allowed average latency for each user.

The authors of [40] proposed general guidelines to design a disaster-resilient GDC which includes site placement and topology design, selection of VM placement and backup strategies. The authors of [2] proposed a simple optimization framework to find the minimum number of additional servers required and how many of them need to be placed at each data center such that latency constraints are satisfied and any failure of a single data center is masked.

Global Content Balancing model introduced in [54], advocates the placement of data center with the associated data in the ISP network closer to client regions. This makes sure that the client requests are routed to a nearby data center, which also avoids expensive fetching of overseas content and excessive delay. Our model assumes the data center to be optimally placed and the client data is placed locally at a nearby data center (could also be multiple data centers).

In summary, the literature mostly addressed the problem of cost-aware data center placement and capacity provisioning for GDCs considering the demand and QoS constraints, it does not address the problem of handling failures in a data center with minimum TCO.

2.4.2 Geo-Distributed Load Balancing Approaches

The central idea of a GLB algorithm is wide-area workload placement. Informed load balancing policy exploits spatio-temporal variation in the electricity price, available renewable energy, demand multiplexing with the objectives of reducing power consumption cost [4], maximizing the use of renewable energy [11], or minimizing

the number of servers used [2]. We classify the existing literature on load balancing algorithms in GDC based on their primary objectives as discussed below

Electricity Price-aware Load Balancing

One of the earliest works that leverages electricity price differential was [4]. In this work, for a given system of servers distributed geographically, the problem addressed was to map client requests to cluster such that the total power consumption of the system is minimized. They considered different variations in the price such as, daily price fluctuations, different market types, and hour-to-hour volatility. They used Akamai trace to show that leveraging price differential yields substantial savings in cost. In particular, they proposed cost-aware routing heuristics which map client requests to cheaper price regions lying within its radial geographical distance.

The authors of [32] have proposed request distribution policy that minimizes the electricity cost, moving jobs where energy is cheaper subject to latency constraints. However, they did not consider network latency or time varying workload. They showed that significant cost saving can be achieved using an intelligent request distribution scheme. In [55], the authors proposed on-line algorithm for migrating batch jobs between data centers based on electricity price differential. However, job migration comes with non-trivial bandwidth cost. Their work differs from the others in that, they incorporated bandwidth cost in the objective. Unlike other works, they considered delay-tolerant batch jobs.

The work in [56], used the concept of electricity price capping, arguing that large scale data centers are not only price-takers but also price makers in dynamic electricity market. They proposed two step cost-capping electricity minimization algorithm. In the first step, they model the effect of power demand on electricity price and the effect of power consumption on the electricity cost. In the second step, their solution forces the electricity cost capping by providing QoS to premium users

and best effort service to occasional users, when the power consumption cost exceeds the monthly budget. Their solution not only achieved significant reduction in the electricity cost but also maximized their throughput while capping the desired cost.

In [57], the authors demonstrated the effect of relaxing delay constraint for delay-tolerant work load. Their solution exploits temporal and spatial variation of both workload and electricity prices. They demonstrated a power cost-delay trade-off which can be exploited to achieve further reduction in cost at the expense of service delay. Authors of [58], proposed a simple heuristic of mapping HPC workload or web search in accordance to electricity price variability as search query workload and electricity prices show similar spatio-temporal variation.

From the above discussion it can be observed that one of the primary objectives being considered by most of the works is to leverage the variation in the electricity price to achieve substantial savings in cost. In all these works the secondary objective was service latency reduction, bandwidth cost reduction, or capping monthly power budget.

Renewable Energy-aware Load Balancing

To reduce the carbon footprint and for sustainable design of data centers, research efforts were made to efficiently integrate renewable energy sources along with traditional brown sources in powering data centers. In [11], the authors presented an excellent summary of the efforts made for integration of renewable energy sources in data centers for scheduling and load balancing. There are a few studies that have considered additional factors like powering of data center using renewable energy sources (at-least partially). In [27] authors showed that it is possible for a GDC to exploit uncorrelated wind sources to satisfy 95% of energy consumption by wind power following power-aware routing policy.

In [28] authors have addressed the problem of sitting and provisioning green

data centers, where the objective is to minimize the data center and renewable plant building cost while satisfying bound on green energy integration, availability, and delay guarantees. The authors of [59], first observed that the renewable energy production is highly unpredictable and expensive. Owing to high upfront installation cost maximizing renewable energy may overburden the service operator. Therefore, they proposed fractional linear programming to determine request distribution policy maximizing renewable energy usage under monthly budget and QoS constraints. The work in [60], proposed an optimization framework to optimize the energy cost of GDC with QoS constraints, under the assumption that each data center is equipped with renewable energy power generators, where the data center draws power from grid only if the consumption exceeds the green power available.

In [36], authors proposed optimization framework for enabling GDCs to cap their brown energy usage leveraging green energy, while satisfying SLA and minimizing the energy cost. The outcome of the framework is request distribution policy to be implemented at the front-end proxy server. Every hour, request distribution policy is arrived by considering electricity price variability, estimated load, and renewable energy availability. In [61], the authors used online load balancing policy that follows the available renewable energy to provide significant environmental benefit and reduction in cost. Here, the benefit comes from dynamically adjusting the routing and service capacity at each location. However, achieving the objective is challenging due to inaccurate prediction in future workload, renewable energy availability (that highly variable and intermittent) and the electricity prices beyond a short time interval.

In [62], the authors claimed that using GLB to reduce energy cost leveraging electricity price variability across the regions, might lead to increase in total energy use while reducing the energy cost. This is mainly due to the fact that data centers at locations with cheaper energy may serve demand at higher frequency to satisfy the

delay bound. The authors proposed two distributed algorithms for achieving optimal load balancing and showed the advantages of integrating renewable energy into the grid using a long-term energy model. In [8], the authors formulated optimization model and proposed greedy heuristics for request-routing and traffic engineering in GDCs. They presented the three-way trade-off between access latency, electricity cost, and carbon footprint. They also highlighted the impact of carbon taxes on data-center carbon footprint reduction and reported that carbon tax is not effective because taxes are only about 5% of the electricity price.

From the above, it is evident that renewable energy integration is not only a obligation for reducing carbon footprint but also an economically and technically viable solution. Further, the generating cost of renewable energy tends to reduce with time due to the technological improvements in the equipment efficiency and the increasing deployment. This can benefit the usage of green energy over the long-term. Hence, integration of renewable energy is crucial for sustainable growth of data centers and GLB strategies can be designed intelligently to minimize the operating cost while greening the data centers.

2.5 Summary

In summary, we discussed the issues in minimizing the cost due to power consumption in GDCs. While the most common strategy to tolerate failure at a data center site is to provision sufficient spare capacity at the other sites, exploiting spatio-temporal variation in the electricity price, renewable energy availability, and demand multiplexing will lead to lower TCO. The motivation for the work in this thesis is primarily based on the aforementioned factors in capacity provisioning and load balancing to reduce the TCO. We also reviewed important literature in these areas and observed that there is not much work in cost-aware capacity provisioning and load balancing for fault-tolerant GDCs. Because of requirement of large spare

capacity to handle failures at a site, sufficient care needs to be taken to minimize the CAPEX and OPEX while handling failures. Accordingly, the work in this thesis proposes optimization models and algorithms for cost-aware design of fault-tolerant GDCs and load balancing thereof. In the next chapter, we discuss the problem of cost-aware spare capacity provisioning and the optimization model for the same.

Chapter 3

Cost-aware Provisioning of Spare Capacity for Fault-tolerant GDCs

3.1 Introduction

Designing a fault-tolerant GDC usually involves spare capacity provisioning (*i.e.*, allocation of additional servers to mask the failure) across different data center sites. A naive approach uniformly distributes the spare capacity. However, all data centers do not have the same capacity and are characterized by different electricity prices, bandwidth cost, carbon tax and varying user demand over time. Along with service restoration, it is also important that the required data is available at an alternate location after failure. This is handled by replication of data according to a pre-determined policy. There are two options possible for data replication namely, single site replication and multiple site replication. In single site replication, the data is replicated to another nearby data center. In case of a failure, if the replicated site is overloaded, client requests are directed to any other data center meeting the latency requirement. In this case, the data would be pulled from the replica, which results in greater latency and bandwidth cost (we call this a post-failure

penalty). In order to ensure co-location of data with the compute servers, the data is often replicated at multiple sites. However, multi-site replication involves large replication cost since the data center operators are typically charged for the number of bytes transfered [63] and/or the bandwidth cost between the replication sites [16]. Therefore, the replication cost should be considered while designing the data centers for high availability.

In summary, designing a fault-tolerant, highly available, GDC involves minimizing the spare capacity (number of servers) across the data centers considering the cost of power consumed and data replication, subject to a set of constraints related to client demand, delay bound, and the power and capacity available. We call this problem cost-aware capacity provisioning (CACP) wherein, the main challenge is to minimize the TCO for data center operators by leveraging the spatio-temporal variation in electricity price and user demand.

3.1.1 Motivation

The work in this chapter is motivated by the following observations about the GDCs currently in use.

- **Electricity price variation:** In a de-regulated electricity market, electricity prices vary across space and time. Recent trends show that the operating cost exceeds the server cost at many data center locations. Therefore, we argue that greater attention should be put on optimizing data center power consumption cost instead of only minimizing the servers while designing fault-tolerant data centers.

- **Replication Cost:** Usually, cloud service providers connect their data centers with dedicated wide area network (WAN) links that are significantly expensive. Therefore, informed data replication must be carried out in order to minimize

the operating cost involved. For example, Amazon Web Services (AWS) charges an inter-data center transfer at around \$0.12-0.2/GB across geographic regions and \$0.01/GB in the same region [63]. Literature also suggests that the data replication may be charged based on the distance between the replicating sites, *e.g.*, \$1 to transfer 2.7 GB of data over 100km [16].

To the best of our knowledge, the only work that advocated the importance of fault-tolerant capacity provisioning in a distributed data center was that in [2]. Though the basic problem was similar, we used minimization of the TCO as the objective apart from handling the replication cost. As reported in Table 1.1, the electricity cost of powering a server is comparable to (or higher than) the server acquisition cost. Therefore, we used minimization of the TCO as an objective in spare capacity provisioning while considering different models for data replication.

In this chapter, we give an MILP-based solution for the CACP problem to optimize the TCO, while complying to the customer demand, latency requirements, and being cost effective while masking the failure of any one data center (at a time). We modelled two strategies for data replication, single site and multiple site for data affinity. Evaluation of our model suggests that although the multiple site model is costlier, it is preferable when the post-failure penalty is large in the single site model. Numerical results show that the CACP model results in significant saving in the TCO compared to the existing models.

The rest of the chapter is organized as follows: In Section 3.2 we present the cost models used, formulation of the CACP problem and discussed the complexity of the formulation. We also illustrate the working of the model with a small example. Numerical results demonstrating the advantages of the proposed model over the existing ones are reported in Section 3.3. We conclude the chapter in Section 3.4.

3.2 MILP Model Formulation

In this section, we first state the assumptions used in the model and present the models considered for various cost factors. Next, we present the MILP formulation of the CACP problem and also prove that the problem is NP hard.

3.2.1 Assumptions

The following assumptions are used in the model.

- We assume that the failure of the data center at a site is an independent process, *i.e.*, data centers are not susceptible to common disaster situations [40]. For example, a power outage, building fire or any local disaster at one data center location will not effect the remaining data centers.
- Data replication is handled with any popular geo-distributed data replication strategy.
- Failure detection and request re-routing is handled by the load balancer proxy.
- Data centers are connected using dedicated virtual links and the cost of the data transfer is based on the actual usage.
- The demand from a client region is proportional to the population. Propagation delay within the client region is assumed to be negligible.
- All the servers have similar configuration and can serve requests for any service. However, the response sizes can be variable.

3.2.2 System Model

In this section, we define the variables and cost models used in the formulation. Table 3.1 lists all the input parameters, decision variables, and cost components in

Variable	Meaning
Input Parameters	
S	set of data center locations
U	set of client locations
A	set of application types
H	total time horizon
s	index for data center location
u	index for client region
f	index for failed data center
h	index for time slot in time horizon
a	index for application request type
B	processing rate of server in bits per second
J_a	job size for request of type a in kB
P_s^{fh}	power consumed at data center s for application a during hour h with failed data center f
$P_s^{h\ max}$	maximum power available at data center s during hour h
γ_s^{fh}	average server utilization at data center s during hour h and failed data center f
γ^{max}	maximum value of γ to avoid waiting
L_u^{ah}	total number of requests generated for application A from user location u during hour h
D_{su}	propagation delay between client region u and data center s
D_{max}	the maximum tolerable latency
θ_s^h	electricity price per kWh at data center s at hour h
ρ_s	transmission loss of electricity at data center s
α	server acquisition cost
δ_s	carbon tax at data center s
M^{min}	minimum number of servers at any data center
M^{max}	maximum number of servers at any data center
ν_{si}	bandwidth cost for data center s to data center i
ξ	number of bytes required for data replication of single request
Decision Variables	
m_s	number of servers in data center s
λ_{su}^{afh}	number of requests for application a from user location u, served by data center s during hour h and failed data center f
y_{su}	binary variable that denotes whether client location u lies within the latency bound of data center s
Cost Components	
F	total cost of ownership, including server acquisition cost, operating cost and data replication cost
Φ	server acquisition cost
η	cost of data replication to nearest data center for durability
κ	cost of multi-site data replication
Θ	power consumption cost
τ	carbon tax incurred

Table 3.1: Summary of notation used in the model

the model.

Failures: Let S denote the set of data centers. The data centers are indexed between 1 and $|S|$. We use an index variable f to represent the failure of a data center. f takes values from the set $\{0, |S|\}$, where $f = 0$ indicates the case of no failure and $f = s$ indicates that the data center indexed $s \in \{1, 2, \ldots, |S|\}$ has failed. We assume that the probability of a single data center failure, *i.e.*, $f \neq 0$, is very small.

Demand: Let λ_{su}^{afh} denote the number of requests for an application type a, from a client region u, served by the data center at site s, during hour h after the data center indexed $f \in \{1, |S|\}$ has failed. Let L_u^h be the total demand from the client region u at hour h.

Server Provisioning: Let m_s denote the number of servers required in a data center at s. We define M^{min} and M^{max} to be the minimum and maximum number of servers that can be provisioned at any data center based on the space and power availability.

Delay: Let D_{max} be the maximum latency for the service and D_{su} be the propagation delay between client region u and data center site s. A data center must be assigned to the client region such that even after the failure of a site, the latency continues to be lower than D_{max}.

Server Utilization: Let the processing rate of the server be B bps and let J_a be the response size for an application type $a \in A$. The service rate for type a is defined by $\frac{B}{J_a}$ requests per second. There are three approaches to model the average utilization of servers as given below:

1. *Mutually Exclusive (ME) approach:* Each type of application is assigned to a pre-defined set of servers. Let m_{sa} be the number of servers allocated to serve the requests of type a. Requests for different services are queued in a single queue, from which a scheduler dispatches the requests to the corresponding

servers. The average utilization of servers serving the requests of type a can be defined as

$$\gamma_s^{fh} = \frac{\sum_u \lambda_{su}^{afh} J_a}{m_{sa} B}, \tag{3.1}$$

This approach of scheduling simplifies the resource provisioning but leads to under-utilization of servers.

2. *Maximum (MAX) approach:* Assuming all the requests to be homogeneous, the servers can be provisioned according to the highest processing rate required. In this case, the average utilization of any server can be defined as

$$\gamma_s^{fh} = \frac{\sum_{u,a} \lambda_{su}^{afh} J_{max}}{m_s B}, \tag{3.2}$$

where J_{max} is the maximum mean file size across different application types. This approach also suffers from resource under-utilization. On the other hand, provisioning based on the smallest processing rate leads to under-provisioning of resources.

3. *Multiplexed (MUX) approach:* In a virtualized environment, any type of workload can be served by one of the free servers. All the requests are placed in a common queue and served by a set of identical servers. This model is followed in most of the recent papers [64]. The average utilization of a server in this case can be defined as

$$\gamma_s^{fh} = \frac{\sum_{u,a} \lambda_{su}^{afh} J_a}{m_s B} \tag{3.3}$$

In this chapter, we consider this model for server utilization but study the implications of other models in Section 3.3.2.

3.2.3 Cost Models

Next, we define the various cost factors used in the formulation and define the models for them.

Server Acquisition: Let the cost of a server normalized over the duration considered for evaluation be denoted by α. The total cost of servers across all the data centers, denoted by Φ is simply given as

$$\Phi = \alpha \sum_{s} m_s \tag{3.4}$$

Data Replication: Let ν_{sg} be the bandwidth cost for data replication from data centers s to g. For every request served by a data center s, let ξ be the volume of data to be replicated. We consider two possible replication models.

1. *Single site replication*: In this case, the data from a primary data center is replicated to the nearest data center.

2. *Multiple site replication*: In this case, the data from a primary data center is replicated to all possible data centers where the client's request may be routed without exceeding the latency bound, denoted by PD_s.

We define the cost of replication R for these two options using the equation below.

$$R = \begin{cases} \sum_{a,f,u,s,h} \left(\lambda_{su}^{afh}\ \xi\ \nu_{sg}\right) & \text{Case 1} \\ \sum_{a,f,u,s,h} \left(\lambda_{su}^{afh}\ \xi \sum_{i \in PD_s} \nu_{si}\right) & \text{Case 2} \end{cases} \tag{3.5}$$

Power Consumption: Let θ_s^h denote the electricity price at data center location s in hour h of the day. Let P_{idle} be the average power consumed in idle condition and P_{peak} be the power consumed at peak utilization. Let E_s be the PUE of a data center. The total power consumed at s over an hour h can be expressed as [16]

$$P_s^{fh} = m_s(P_{idle} + (E_s - 1)P_{peak}) + m_s(P_{peak} - P_{idle})\gamma_s^{fh}. \tag{3.6}$$

The cost of power consumed at all data centers Θ can be expressed as

$$\Theta = \sum_{s,h,f} \theta_s^h P_s^{fh} \tag{3.7}$$

Carbon tax: Let δ_s denote the carbon tax levied at data center location s and ρ_s denote the transmission loss incurred. The total cost due to carbon tax is

$$\tau = \sum_{s,f,h} \delta_s(\rho_s + 1)P_s^{fh} \tag{3.8}$$

3.2.4 CACP Model

Considering all the cost factors defined above, we can define the CACP problem as the problem of minimizing the TCO subject to the set of constraints on latency and availability. The TCO, denoted by $\digamma$ is the sum of the server cost Φ, data replication cost R, electricity cost Θ, and carbon tax τ. For notational simplicity, we define the following decision variables:
$\mathbf{m} \triangleq [m_s, \forall s \in S]$,
$\boldsymbol{\lambda} \triangleq [\lambda_{su}^{afu}, \forall s \in S,\ \forall u \in U,\ \forall a \in A,\ \forall h \in H, \forall f \in \{0, 1, 2.., S\}]$ and
$\boldsymbol{y} \triangleq [y_{su}, \forall s \in S,\ \forall u \in U]$

The CACP problem can be formally expressed as an optimization model given below.

$$\underset{\boldsymbol{m},\boldsymbol{\lambda},\boldsymbol{y}}{\textbf{minimize}}\ \ \digamma = \Phi + R + \Theta + \tau \tag{3.9}$$

subject to,

$$\sum_{s \in S} \lambda_{su}^{afh} = L_u^{ah} \qquad \forall u, a, h, f \qquad (3.10)$$

$$0 \leq \lambda_{su}^{afh} \leq y_{su} L_u^{ah} \qquad \forall s, u, a, h, f \qquad (3.11)$$

$$M^{min} \leq m_s \leq M^{max} \qquad \forall s \qquad (3.12)$$

$$P_s^{fh} \leq P_s^{h\ max} \qquad \forall s, h, f \qquad (3.13)$$

$$2 D_{su}\, y_{su} \leq D_{max} \qquad \forall s, u \qquad (3.14)$$

$$\gamma_s^{fh} \leq \gamma^{max} \qquad \forall s, h, f \qquad (3.15)$$

$$\lambda_{su}^{afh} = 0 \qquad \forall u, a, h, s = f \qquad (3.16)$$

Among the constraints, Eq. (3.10) ensures that the demands of all client regions in every hour are met. Eq. (3.11) ensures that all the client requests are served by data centers within the latency limit. Eq. (3.12) ensures that capacity limit of a data center (in terms of number of servers) is not exceeded. The constraint on the total power available at a data center is taken care of in Eq. (3.13). Eq. (3.14) ensures that the delay experienced by a client lies within the maximum bound. Eq. (3.15) is used to limit the queuing delay at a data center by bounding the average server utilization to a constant value ($\gamma^{max} \in (0, 1]$), similar to that in [16]. Eq. (3.16) ensures that no demand is served by the failed data center.

The inputs to the CACP problem are as follows: the set of data center locations with the associated costs, maximum average utilization of servers, processing rate of the servers, maximum latency, demand distribution, maximum number of servers at each site, and maximum power available at each site. The model then gives the number of servers across the sites, request assignment to the data centers and the data centers within the latency limit for each client location.

3.2.5 Example for Working of the CACP Model

In this section, we give a simple example to illustrate the impact of the CACP model on the TCO. The proposed CACP model mainly reduces the TCO by exploiting demand multiplexing and spatio-temporal variation in the demand and electricity price. For easier understanding on how this works, we show two examples for *(a)* the impact of demand multiplexing on capacity provisioning and *(b)* the impact of demand multiplexing and electricity price variation on the TCO.

Impact of demand multiplexing on capacity provisioning

Both the CACP and MS models take into account demand multiplexing while provisioning capacity when the CDN model trivially maps requests to the nearest data center to minimize the latency. Consider a scenario with three data centers and three client regions with a maximum latency bound of 25 ms for the service. Fig. 3.1 shows the system used for illustration. Data centers DC_1, DC_2, and DC_3 serve the requests from client regions C_1, C_2, and C_3 given in Table 3.2. Each edge between a data center and client region is weighted by the propagation delay. For simplicity, we considered a case where all the data centers were within the latency bound (25ms) for all the client regions.

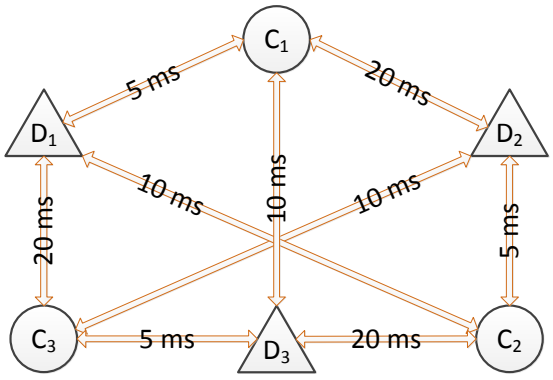
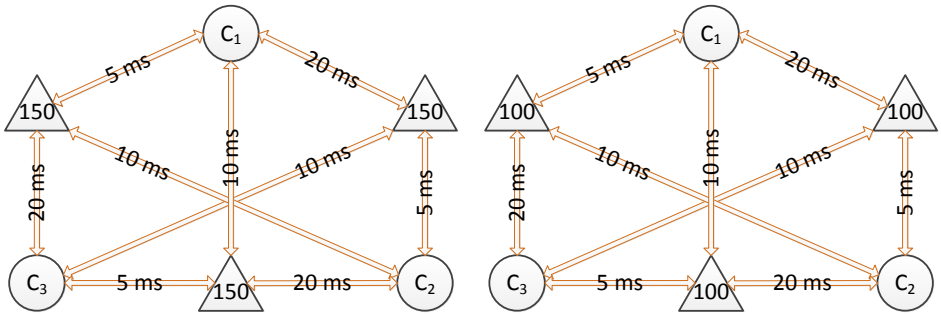


Figure 3.1: System used for illustration

	Timeslot 1 (in hrs)	Timeslot 2 (in hrs)	Timeslot 3 (in hrs)
Client 1	100	50	50
Client 2	50	100	50
Client 3	50	50	100

Table 3.2: Demand across different intervals



(a) (b)

Figure 3.2: Capacity allocation using (a) CDN model, (b) MS model

Considering the case of a data center failure, a demand of 200 units generated from all the client regions needs to be served by the remaining two data centers. In the MS model, we distribute the workload equally across all the active data centers. This gives 100 servers at each data center and the total number of servers to tolerate any data center failure is 300, as shown in Fig. 3.2b. In case of the CDN model, a client region is always served by the nearest data center after failure. For example, C_1 was served by DC_1 before failure, whereas it is served by DC_2 after failure. Therefore, DC_2 should be provisioned not only to satisfy C_2's demand but also with sufficient spare capacity to make up for the failed data center DC_1. This gives rise to DC_2 being provisioned with 150 servers to meet the demand across any interval (when DC_1 might fail). Accounting for the possibility of any data center failure, the server distribution across all the data centers is obtained to be 150 units as shown

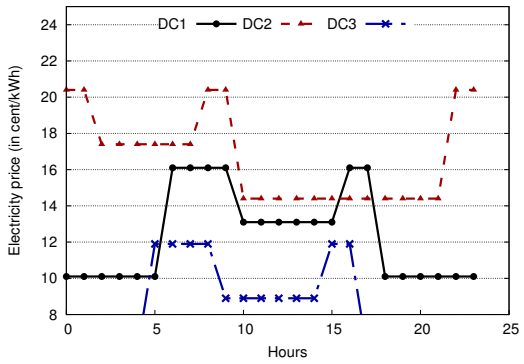


Figure 3.3: Electricity price at three data center locations

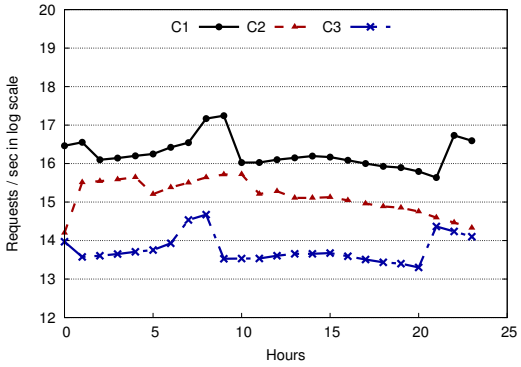


Figure 3.4: Demand distribution at three chosen client regions

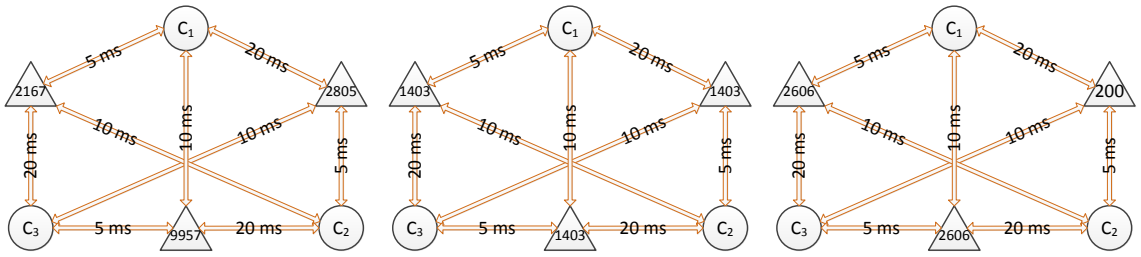
in Fig. 3.2a. We can conclude that the MS model exploits demand multiplexing while satisfying the latency bound of 25 ms and requires only 300 servers against 450 units with the CDN model. CACP model also gives the same result if we ignore the variation in the operating cost across the data centers.

Impact of demand multiplexing and electricity price variation on the TCO

Consider the three data centers shown in Fig. 3.1 with the electricity price variation as shown in Fig. 3.3. It may be noted that the electricity price is highest at DC_2. The demand across the three regions C_1, C_2, and C_3 is shown in Fig. 3.4. For simplicity, we considered the processing rate as 100/sec, P_{peak} and P_{idle} as 400 W and 200 W, respectively, and the server cost as \$2000 (17 cents/hr, assuming 4 years life). We assumed that all the data centers are within the latency limit for any client region. The server distribution obtained after solving the optimization model for CDN (minimize average latency), MS (minimize number of servers), CACP (minimize total cost) is shown in Fig. 3.5a, Fig. 3.5b, and Fig. 3.5c, respectively. The number of servers allocated across all the data center locations is the same with the MS model. However, the CACP model allocates fewer servers at DC_2, where the electricity price is higher. The CACP model always allocates more capacity at

a site where the electricity price is cheaper while satisfying the latency and other constraints.

The normalized TCO obtained using the model is given in Table 3.3. Even though CACP model allocates a larger number of servers than the MS model, the TCO is lowered by exploiting the spatio-temporal variation in the electricity prices for demand distribution. Though the MS model minimizes the number of servers provisioned at each location, it does not give the minimum TCO because of being oblivious to operating cost.



(a) (b) (c)

Figure 3.5: Capacity allocation using (a) CDN model, (b) MS model, (c) CACP model

Models	Normalized TCO
CDN	1
MS	0.89
CACP	0.62

Table 3.3: Normalized TCO with various models

3.2.6 Complexity Analysis

The number of variables in the above formulation is $S + (S + 1)SUAH$
and the number of constraints is $S+(S+1)\{UAH+SUAH+2SH\}+SUAH+SU$.
The asymptotic complexity of proposed CACP model is $\mathcal{O}(S^2UAH)$. With an increase in the number of data centers, the complexity increases quadratically but linearly with the number of client locations, time slots, and application types. The following theorem states the complexity of the problem.

Theorem 3.1. *The feasibility problem of CACP in a distributed data center is in NP-hard.*

Proof. The CACP problem in a distributed data center (without fault tolerance) is in NP-hard, even when resources are of unit size and unit operating cost. The reduction is from the set cover problem.
In a basic formulation, the cost aware capacity provisioning problem (without failure considerations) consists of a set of data center locations DC where the cost of running servers at a data center i is given by $Cost_i$, and a set of client locations C generating a demand to be served. Each client can be served by a data center lying within a given latency bound $Delay$. The goal is to provision a number of servers across data centers so that the total cost incurred is minimum while satisfying client demand and latency bound.

In Lemma 3.1 we reduce the decision version of the set cover problem to the decision version of the CACP problem, which is sufficient to show that Theorem 3.1 holds. The decision version of the CACP problem can be stated as follows. *Given a set of data centers and their server running costs, a set of demand generating client regions and latency bound, does there exist a subset of data centers that can satisfy the client demand with the total cost incurred being at most k?*

Lemma 3.1. *The decision version of the CACP problem is in NP-hard.*

Proof. The decision version of the set cover problem is defined as follows. Given a set system $(\mathcal{U}, \mathcal{S})$ with $\bigcup_{S \in \mathcal{S}} S = \mathcal{U}$ and a positive integer k. The question is does there exist a collection of k or fewer sets of $\mathcal{S}$ that cover $\mathcal{U}$ [65]? This problem is known to be NP-complete and we give a reduction of this problem to the decision version of the CACP problem as follows.

Given an instance of the set cover problem $\mathcal{I}_S$, let us map it to an instance $\mathcal{I}_C$ of the decision version of the CACP problem. For each $u \in \mathcal{U}$, we assign a client region c_u that generates a demand of unit compute capacity to meet its needs. For each $S \in \mathcal{S}$, we assign a data center d_S that is within the delay bound for the clients specified as its elements. For instance, if $S = \{u_1, u_2, \ldots, u_m\}$ then, d_S has $c_{u_1}, c_{u_2}, \ldots, c_{u_m}$ within the delay bound constraint. The cost associated with each data center is 1 unit and each of them has infinite capacity. This completes the reduction of instance $\mathcal{I}_S$ to $\mathcal{I}_C$. It is easy to observe that the reduction from $\mathcal{I}_S$ to $\mathcal{I}_C$ is in polynomial time in the input size of instance $\mathcal{I}_S$. To complete the proof, we need to show that $\mathcal{I}_S$ admits a solution if and only if $\mathcal{I}_C$ has a solution that costs at most, k units.

Suppose $\mathcal{I}_C$ has a solution with less than or equal to cost k units. Without loss of generality, let $d_{S_1}, d_{S_2}, \ldots, d_{S_l}$ be the solution to $\mathcal{I}_C$ that meets demands of all client regions. Note that $l \leq k$ as each data center consumes 1 unit of energy. Each of the clients c_u is served by at least one data center in $d_{S_1}, d_{S_2}, \ldots, d_{S_l}$. Correspondingly, the $S_1, S_2, \ldots, S_l$ cover each $u \in \mathcal{U}$ and thus, it is a solution to $\mathcal{I}_S$ having the size of $l \leq k$.

Conversely, if $\mathcal{I}_S$ admits a solution $S_1, S_2, \ldots, S_j$ with $j \leq k$ we can construct a solution to $\mathcal{I}_C$ that costs at most, k units. The set of data centers $d_{S_1}, d_{S_2}, \ldots, d_{S_j}$ is able to meet the demand of all the client regions c_u as $\bigcup_{1 \leq i \leq j} S_i = \mathcal{U}$. Thus we have constructed a solution to $\mathcal{I}_C$ that costs $j \leq k$ units.

□

Comments: Though the problem is in NP-hard, solving it is a one-time effort only at the time of design. We do not see the running time to be a matter of concern since the CACP problem is always solved offline. We solved all the models centrally using CPLEX with MATLAB on a server with an Intel Xeon processor, 64 GB of RAM, and 64-bit OS. We could not solve the model for more than ten data centers on this server in a reasonable amount of time (few minutes) for an evaluation period of one day (24 hourly slots). We can solve the model optimally for capacity planning in large data centers with higher computational power. For much larger number of variables, we need to go for online heuristics or approximate algorithms.

3.3 Numerical Results

In this section, we solve the CACP model using real-world data and compare the TCO obtained with two other models from the literature. The MILP is solved using CPLEX (Interactive Optimizer 12.6.2.0.) with MATLAB on a Ubuntu 14.04 server based on an Intel Xeon processor with 64 GB of RAM. All the models were evaluated under identical constraints and we used the same cost factors for all the models. The two other models considered were as follows:

- MS model: A rudimentary version of this model was defined in [2]. The main objective of this model is to minimize the total number of servers deployed across all the data centers. The TCO for this model would be the cost of that data center provisioned after minimizing the number of servers.

- CDN model: In this model, the objective is to balance the load across the data centers so that the average response time is minimized. The provisioning of servers in this model would be done so that the client latency is minimal [66].

We compared the TCO obtained using all three models in the results. We also studied the advantages of the CACP model by varying the number of data centers,

demand, request rate, and latency bound. We also studied the impact of server utilization models and replication models discussed earlier on the TCO. We first provide details on the scenarios used and the data set used for the evaluation.

3.3.1 System Parameters

Data center locations: The locations for the data centers are (10 of them): California, Oregon, Virginia, Switzerland, U.K, Ireland, Netherlands, Hong Kong, Japan and Canada. At each location, the number of servers was varied between 1000 and 100,000. This would help us consider both smaller and mega data centers across the world.

Client locations: Based on the data collected for the number of Internet users from [67] we selected the following client regions (15 of them): Brazil, China, Egypt, France, Germany, India, Indonesia, Japan, Mexico, Nigeria, Russia, South Korea, UK, USA, and Vietnam. The propagation delay between the data center locations and client locations was varied linearly with geographical distance in the order of 10 ms for every 1000 km [16].

Electricity prices: We used historical industrial electricity price data ($ per MWh) from publicly available government databases corresponding to various data center locations [21–24]. For the sake of brevity, we do not discuss regulated electricity market prices. Interested readers may see [21]. We use the electricity price model similar to the one in [53], where the price for each location varies during on-peak hours (7 A.M. to 11 A.M. and 5 P.M. to 7 P.M.), mid-peak hours (11 A.M to 5 P.M.)and off-peak hours (7 P.M. to 7 A.M.). The price varies across the periods by as much as 3 cents/kWh [53]. Some states in the USA (like California and Colorado) also add about $0.04 to $0.6/kWh as carbon tax for power consumed from brown energy sources. We ignore the same in the results due to its small contribution in the TCO (less than 1%).

Traffic model: We used the trace of requests to Wikipedia services downloaded from [68]. We downloaded the workload traces for the month of December 2015, containing the total number of requests and aggregate response size for different services of Wikipedia. The demand profile for a 24 hour period, averaged over a month, is plotted in Fig. 3.6. Since the demand had a diurnal pattern we used $H = 24$. This distribution of requests was used to derive an hourly demand for different client regions. We upscaled the number of requests by a factor of 3000 to reflect the traffic handled by larger service providers [4]. For each client region, we divided the workload proportional to the number of Internet users in that region. Table 3.4 shows the split of workload across different client regions obtained from the number of Internet users. Fig. 3.7 shows the hourly demand for a few client regions. The demand during the on-peak period was kept as 1.4 times the mid-peak demand and the demand during the off-peak period was kept at 0.6 times that in the mid-peak period.

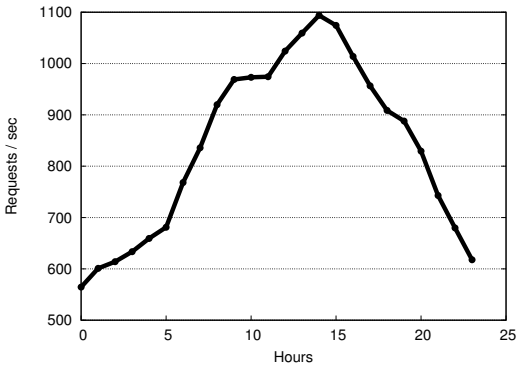


Figure 3.6: Demand profile from Wikipedia.org

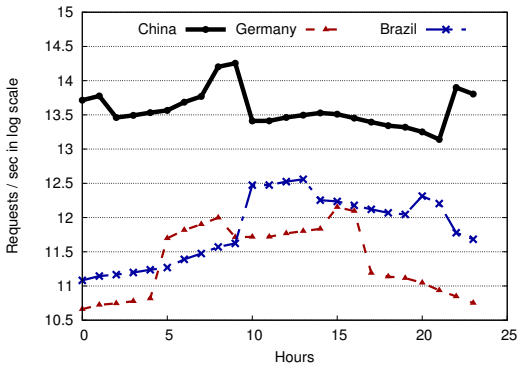


Figure 3.7: Demand distribution from representative client regions

Inter-DC communication cost: For the inter-data center communication cost we used a pricing model similar to the one charged by AWS EC2 services [69]. For example, AWS charges $\$0.12 - \$0.2$/GB across geographical regions.

Other parameters: P_{idle} and P_{peak} were set to 200W and 400W, respectively [70].

Country	Brazil	China	Egypt	France	Germany	India	Indonesia
% Demand	5.33%	31.76%	2.17%	2.78%	3.50%	15.43%	2.04%
Japan	Mexico	Nigeria	Russia	S. Korea	UK	USA	Vietnam
5.64%	2.65%	3.37%	4.50%	2.13%	2.93%	13.69%	2.09%

Table 3.4: Percentage of demand from different regions

The average PUE was set to 1.5 [71]. P^{max} was taken as 100MW/hr for all the locations [53]. The default value for maximum latency was set to 300ms. The size of data to be replicated per request was assumed to be 10KB [72]. We set $\gamma^{max} = 0.8$ [73]. We set the probability of a single data center failure to be 0.005 that corresponded to 1.8 days of failure per year.

3.3.2 Results

In this section, we present the numerical results from evaluating the models by varying the number of data centers, demand, and latency bound. We also study the effect of different models for server utilization and data replication (single site and multi-site) on the TCO. In all the results, we show the normalized values of TCO, where the normalization was done using the maximum TCO seen across all the experiments.

TCO comparison

In this experiment, we varied the number of data centers between 6 and 10 in order to serve the client requests as reported earlier within a maximum latency of 300 ms. Fig. 3.8 shows the normalized TCO for all the models with various numbers of data centers. In this experiment, we used the single site replication model.

Table 3.5 reports the normalized TCO for different cases (third row from the bottom). Reduction in the TCO (in percentage) with the CACP model (compared

Country	6 data centers			8 data centers			10 data centers		
	CACP	MS	CDN	CACP	MS	CDN	CACP	MS	CDN
Japan	20000	15592	20000	20000	11138	20000	16072	8663	20000
Ireland	20000	15592	20000	200	11138	13463	200	8663	13463
California	20000	15592	20000	17360	11138	20000	200	8663	20000
Hong Kong	200	15592	20000	200	11132	20000	200	8663	20000
Virginia	20000	15592	20000	20000	11138	20000	20000	8663	12052
UK	17760	15592	20000	200	11138	20000	200	8663	20000
Netherlands	-	-	-	20000	11138	20000	1089	8663	20000
Switzerland	-	-	-	20000	11138	20000	20000	8663	20000
Canada	-	-	-	-	-	-	20000	8656	15723
Oregon	-	-	-	-	-	-	20000	8663	8035
No of servers	97960	93552	120000	97960	89098	153463	97961	86623	169273
Normalized TCO	0.69	0.89	0.9	0.63	0.89	0.95	0.57	0.88	1
% reduction (w.r.t CDN)	23.35	0.51		33.34	5.66		42.62	12.39	
% reduction (w.r.t MS)	22.96			29.34			34.5		

Table 3.5: Comparison of number of servers provisioned and TCO for all models

to the MS and CDN models) is shown in the last two rows. The fourth row from the bottom shows the total number of servers provisioned with each model across the data centers. The table also shows the locations chosen and the number of servers at each location as the number of locations increases. Since the CACP model exploits the spatio-temporal variation in the electricity prices, the TCO is lowest in the case of the CACP model. Though the MS model minimizes the number of servers provisioned at each location, it does not lead to minimum TCO because, it does not consider the operating cost in capacity provisioning.

From Table 3.5 it can be observed that even with six data centers, the benefit of the CACP model is significant, while the other two models have a similar TCO. This is due to the fact that with fewer data centers, there is not much scope for demand multiplexing. On the other hand, the CACP model assigns a larger workload at a data center location with a lower electricity price. With the addition of another location (the Netherlands, with a lower electricity price compared to the U.K. and

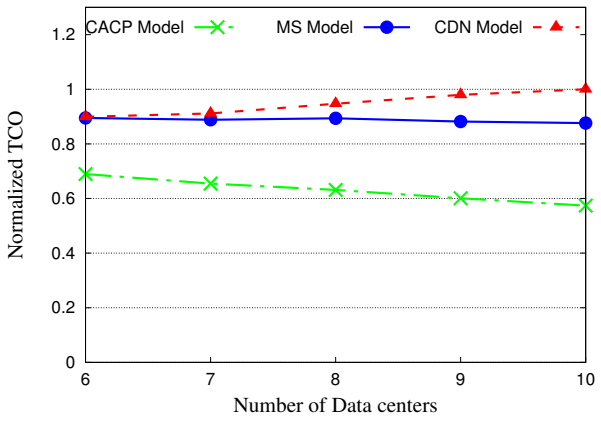


Figure 3.8: Normalized TCO with varying number of data centers

Ireland) the CACP model shifts the servers provisioned in the U.K. and Ireland to the Netherlands (see Table 3.5). This improves the TCO in the CACP model by about 3.5%. While the CACP model suggests more servers, the TCO is minimized due to shifting them to locations with a lower operating cost. This can be observed from the table that shows the MS model gives the same number of servers at each location. We can observe that the CACP model achieved a TCO reduction of up to 35% compared to the MS model, and up to 43% compared to the CDN model.

Impact of data center locations on the TCO

We also studied how the choice of data center locations affects the TCO with the CACP model. We evaluated our model for the sets of locations listed in Table 3.6. The TCO obtained with the CACP model is shown in Fig. 3.9. Between *Set 2* and *Set 5*, Oregon replaced Ireland, where the electricity price was lower (refer Table 1.1). Oregon (being in the USA) also meets the latency constraints for the largest number of users (from Americas as reported in Table 3.4). Both these factors lead to a lower TCO for *Set 2* than *Set 5*.

To understand the contribution of the replication cost to the TCO, we evaluated the CACP model considering the single site replication (SR) and multiple site

Set 1:	California, Japan, Hong Kong, Ireland, Switzerland, Virginia
Set 2:	California, Japan, Hong Kong, Netherlands, Oregon, UK
Set 3:	Japan, Hong Kong, Netherlands, Oregon, Switzerland, Virginia
Set 4:	California, Japan, Hong Kong, Ireland, Netherlands, UK
Set 5:	California, Japan, Hong Kong, Ireland, Netherlands, UK

Table 3.6: Different sets of locations

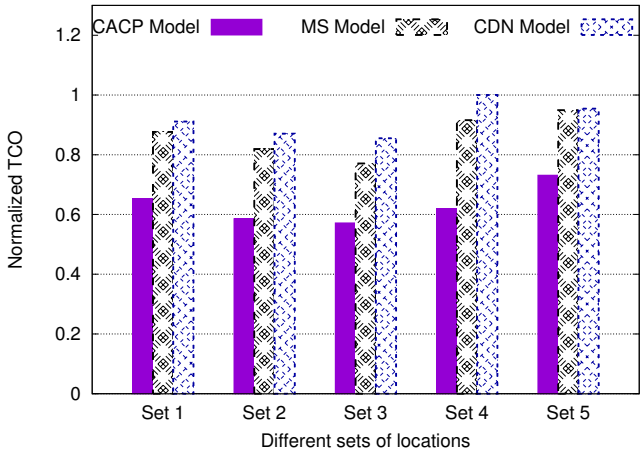


Figure 3.9: Normalized TCO with different sets of locations

replication (MR) models with various numbers of data centers. The maximum latency was set to 300ms and the demand was generated as reported in Section 3.3.1. Fig. 3.10 shows the TCO split into the replication cost and the cost due to power consumed for both the replication models. It can be observed that in the SR model, the contribution of the replication cost is small in the TCO. On the other hand, the MR model is costly for replication and the replication cost increases with the number of data centers as shown in Fig. 3.10. Therefore, this approach may be preferred only when the post-failure penalty is very high.

Fig. 3.11 shows the TCO for the CACP model with and without the replication cost being considered as the number of data centers varies. It can be observed that

the single site replication cost alone accounts for 20% of the TCO. Therefore, the CACP model (without replication cost) lowers the TCO by about 20% compared to the model with replication.

In all the subsequent experiments, we considered only a single site replication model while evaluating the TCO.

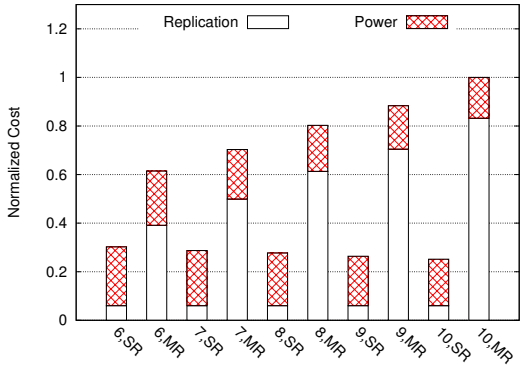


Figure 3.10: Split up in TCO: Replication cost and electricity cost

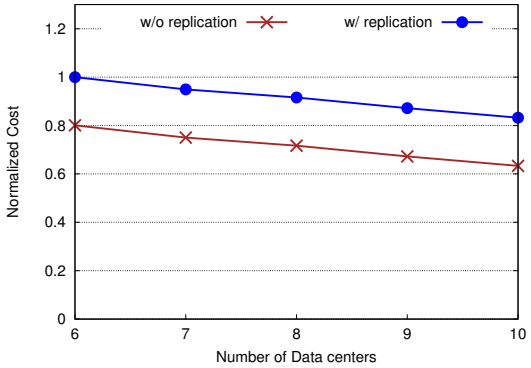


Figure 3.11: Impact of single site replication cost on TCO

TCO vs. Worst-case Latency

Next, we studied the impact of maximum latency bound on the TCO. We evaluated the models for 8 data centers, 15 client regions, and the aggregate demand as mentioned in Section 3.3.1. The maximum latency was chosen in the range of $150 - 350$ ms. Fig. 3.12 shows the normalized TCO for all the models with varying latency. We can observe that the CACP model results in a lower TCO by upto 38% and 32% compared to the CDN and MS models, respectively. In the CACP model, there is a choice in the number of data centers capable of serving the requests from a particular client region that leads to better multiplexing of resources and a reduced TCO. Apart from this, the CACP model also selects the data centers in regions with lower electricity prices while meeting the latency bound. Although the

CDN model gives minimum latency, request routing is oblivious to the variation in the electricity price. Therefore, the TCO is higher for the CDN model particularly when the latency requirements are not very stringent. We conclude that the CACP model is more advantageous for services without stringent latency requirement.

In Table 3.7, we report the increase in worst-case latency (compared against the CDN model), when the CACP model targets TCO reduction. At a worst-case latency of 150 ms, our model has about a 25% lower TCO. When we target a higher reduction in the TCO, the worst-case latency in the CACP model increases. For about 40% reduction in the TCO, our model leads to worst-case latency of 300 ms. The reduction in the TCO is due to the fact that the CACP model exploits demand multiplexing and variation in the electricity prices, when the latency requirement is relaxed.

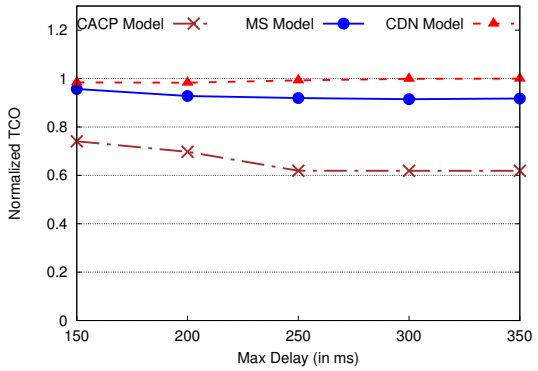


Figure 3.12: Normalized TCO by varying maximum latency bound

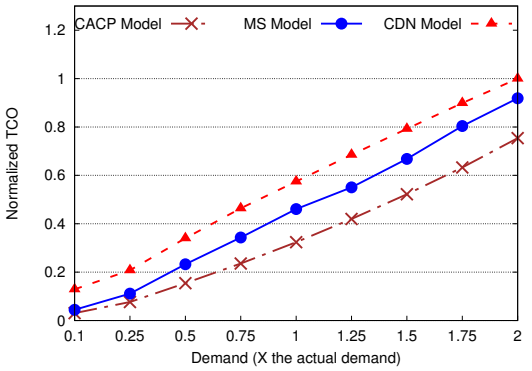


Figure 3.13: Normalized TCO by varying demand

Impact of Demand

We evaluated all the models varying the total demand with 8 data centers and a maximum latency bound of 300ms. Results in Fig. 3.13 show that as the demand increases the TCO for the CACP model reduces compared to other models. Due to

Target Reduction in the TCO (%)	Worst-case Latency (ms)
25	**150**
30	**200**
35	**250**
40	**300**

Table 3.7: Worst-case latency with the CACP model corresponding to the TCO reduction (compared to CDN model)

the capacity limit of a data center, higher demand causes saturation of all the data centers in the regions with cheaper electricity. This reduces the choices available and leads to the selection of other locations with costlier electricity prices. The proposed model is advantageous only when the data center does not operate at peak utilization. Under a heavy load, the CACP model can help the provider determine an optimal data center upgrade plan while minimizing the TCO.

Impact of demand multiplexing

To study the impact of demand multiplexing on the TCO, we evaluated the models by varying the number of data centers from 6 to 10. The electricity price for all the data centers was fixed at 10 cents per kWh throughout the day and the replication cost was fixed to \$0.2/GB. The delay bound was set to 300ms. It can be observed from Fig. 3.14 that CACP and MS models have the same TCO, which is lower than that with the CDN model. This is because, the CDN model does not use demand multiplexing due to the latency minimization objective. The CACP model reduces the TCO by almost 45% compared to the CDN model. We also noticed that the CACP model eventually gives the same TCO as the MS model, because cost reduction is only possible by demand multiplexing that minimizes the total number of servers (due to a uniform electricity price). The TCO reduction of about 10%

can be attributed to demand multiplexing as the number of data centers increases.

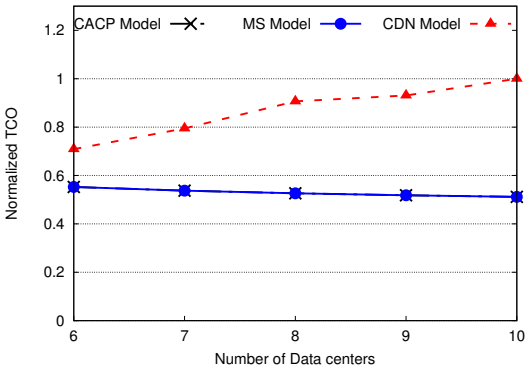


Figure 3.14: Effect of demand multiplexing

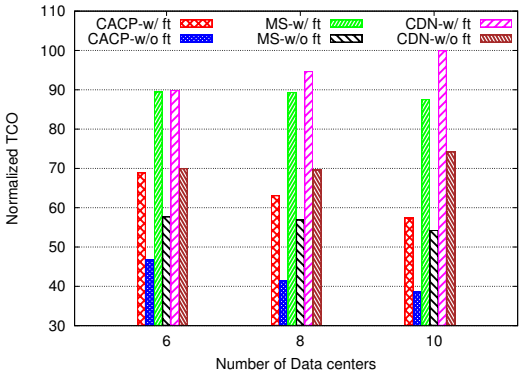


Figure 3.15: Cost of provisioning with and without failure

Cost of over provisioning

To study the cost of over-provisioning for fault tolerance, we evaluated all the models by varying the number of data centers. Fig. 3.15 shows the normalized TCO obtained with and without fault tolerance using each model (normalized with respect to the largest TCO across all the cases). In the plot, CACP-w/ft and CACP-w/o ft indicate the TCO achieved using the CACP model with and without failure, respectively. Results show that when fault tolerance is not considered, the TCO is always lower, because fault-tolerance demands over-provisioning of servers. This increases both CAPEX and OPEX and hence, the TCO. For the case of 6 data centers, provisioning for fault-tolerance increases the TCO for the CACP, MS, and CDN models by 47%, 55%, and 28%, respectively. On the other hand, when the number of data centers increases to 10, the cost of over-provisioning is 49%, 61%, and 34% for the CACP, MS, and CDN models, respectively. We also notice that the CACP model with failure leads to a lower TCO than the CDN model without failure across all scenarios. This means that the CACP model provides resilience against a single data center

failure with no additional cost compared to the CDN model.

Server models for heterogeneous workload

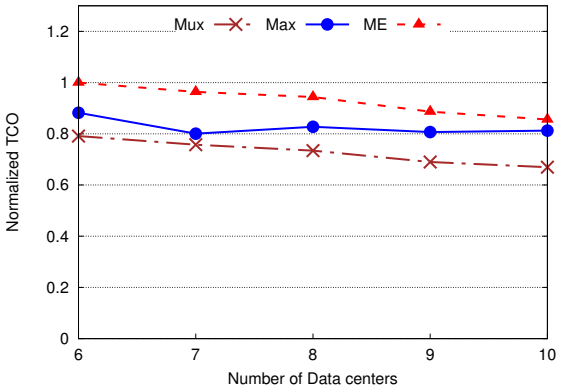


Figure 3.16: Normalized TCO considering different approaches to address workload heterogeneity

We evaluate the CACP model using each of the three models discussed in Section 3.2.2 to understand their impact on the TCO. We set the delay bound to 300ms and used the same demand as in other cases. Fig. 3.16 shows the TCO for different server utilization models. It can be seen that the MUX approach results in a maximum reduction in the TCO (about 22% and 18% compared to the ME and MAX, respectively). This is due to the fact that compute resources are effectively utilized in this approach. The ME and MAX approach both have a drawback of resource under-utilization (due to over-provisioning of resources). The ME approach leads to the maximum cost because there is no scope for multiplexing demand across servers assigned for different types of services. In the MAX approach, there is a scope for multiplexing of servers due to the use of a single server pool.

Key Observations:

- The CACP model provisions more servers but reduces the TCO up to 35% compared to the MS model, and up to 43% compared to the CDN model.

- The CACP model is cost effective when the latency requirement is not stringent and a data center does not operate at its peak utilization.
- It is possible to achieve availability against single data center failure with no additional cost using the CACP model compared to the CDN model. Choice of replication strategy (SR and MR) plays an important role in determining the TCO. Particularly, the contribution of the replication cost to the TCO is significantly high when the number of data centers increases with the MR approach. Therefore, the MR approach is good only when the post-failure migration penalty is high.
- The MUX approach to handle a heterogeneous workload at a data center results in a maximum reduction in the TCO, and this is a viable approach due to virtualization.

3.4 Conclusion

In this chapter, we addressed the problem of cost-aware capacity provisioning for geo-distributed data centers capable of masking single data center failure. We proved that this problem is NP-hard and proposed an MILP formulation to reduce the TCO. The proposed model leads to a lower TCO than the MS and CDN models due to its ability to multiplex demand considering the spatio-temporal variation in electricity prices and the demand. We also modeled different approaches to serve heterogeneous demand and data replication. The CACP model achieves a cost reduction of up to 34% and 50% compared to the MS and CDN models, respectively. Our model leads to a lower cost in designing a fault-tolerant data center particularly, when the electricity costs vary widely across the data center locations along with higher PUE values, which appears to be the case with most of the GDCs. Our model is also useful to study the effect of the replication cost on the TCO for planning distributed

data centers.

In this chapter, we assumed complete failure of a data center at a site and also that the data centers are powered by brown energy sources only. However, there is a pressing need for renewable energy integration for greener GDCs. In the next chapter, we consider a generalized failure model (partial and complete failure at a site) and data centers collocated with renewable energy sources. For such GDCs, we seek to determine the optimal server distribution that minimizes the total cost while maximizing the usage of renewable energy.

Chapter 4

Capacity Planning in Fault-tolerant GDCs Collocated with Renewable Energy Sources

4.1 Introduction

It is reported that data centers consume about 1.3% of electricity worldwide with an estimated growth rate of 12% per year and by 2020 this fraction is estimated to grow upto 8% [11]. High energy consumption not only results in electricity cost but also in increased carbon emissions. Hence, there is a pressing need for integrating renewable energy sources for *greener* GDCs. As discussed in Chapter 2, multiple options exist for integration of renewable energy sources with a data center. In this chapter, we consider renewable energy-powered data center (or green data center), where the renewable energy generators are collocated with a data center. We consider wind and solar power plants, which have a great potential for use as green power sources [11, 14].

Designing a fault-tolerant green GDC requires over-provisioning servers across

sites, taking into account various factors like green energy availability, electricity price variation, user demand, delay and availability requirements. Availability requirements demand the capability of masking partial or complete data center failures. An interesting aspect of capacity provisioning in green data centers is that the cost of powering the servers depends heavily on the spatio-temporal variation in the electricity price, green energy availability, and user demand (for diurnal applications). Therefore, the operator must intelligently provision compute capacity to maximize the green energy usage while satisfying the delay and availability constraints. We call this problem green energy cost-aware capacity provisioning (GCACP) problem. It may be noted that capacity provisioning problem is a variant of facility location problem which is proved to be NP-hard.

Motivation: The work in this chapter is motivated by the following observations from the literature.

- Even though renewable energy production is highly intermittent, it turns out that it becomes more predictable with an increase in the number of renewable energy sources connected to the grid across multiple locations [74]. This is due to the effect of geographic diversity and the *law of large numbers i.e.*, large number of geographically distributed renewable energy sources installed makes the availability of renewable energy more predictable with a certain degree of accuracy [27, 28]. The work in [27] established that it is possible for GDCs to exploit uncorrelated wind sources to satisfy 95% of energy requirement following a wind power-aware routing policy.

- Electricity price variations: In the current de-regulated electricity market, electricity prices vary across space and time. Recent trends show that server operating cost is gradually exceeding the acquisition cost and it could be 1.5 times the server cost.

To the best of our knowledge there is no earlier work on capacity provisioning

for fault-tolerant green data centers. With the pressing need to use green energy in data center operations, it is critical to provision compute capacity taking into account operating cost and optimal renewable energy usage simultaneously. Note that green energy is costlier than brown energy. In order to design fault-tolerant green GDCs, we propose an MILP model to maximize the use of renewable energy while minimizing the total cost of spare capacity provisioning.

The rest of the chapter is organized as follows. The optimization model is discussed in Section 4.2 and results that validate and demonstrate the impact of proposed model on TCO are presented in Section 4.3. The chapter is concluded in Section 4.4.

4.2 MILP Framework

In this section we discuss the proposed optimization model for GCACP problem in data centers powered by renewable energy sources. First, we present the architecture of a typical renewable energy powered GDC along with the assumptions used in the model.

4.2.1 System Setup and Assumptions

The schematic of a renewable energy-powered GDC is shown in Fig. 4.1. S is the set of data centers each housing m_s number of servers ($s \in S$), and powered through grid as well as on-site renewable energy sources (wind and/or solar). Energy storage devices (ESDs) are used to store green energy. Besides using ESD, our model also considers selling the surplus green energy back into the grid at utility sell back price (this approach is called net metering) [14]. There are front-end proxy servers associated with each client region handling a load of L_u^{ah} to map the requests to backend data centers as shown in Fig. 4.1. Let λ_{su}^{fah} be the number of requests from

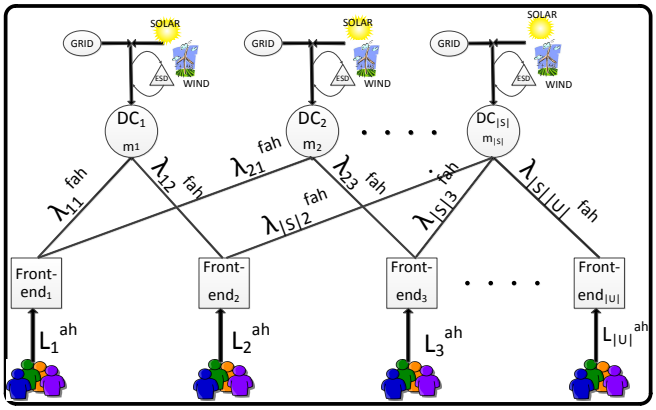


Figure 4.1: Architecture of renewable energy powered GDC

client region u to data center s at hour h for application type a when data center indexed f has failed ($f \in \{1, 2, \ldots S\}$). In our optimization model, m_s and λ_{su}^{fah} are the decision variables while other parameters like L_u^{ah}, brown electricity price and green energy availability are the input parameters.

Assumptions: The following assumptions are used for the model.

- We consider on-site renewable energy generation as several companies like Apple and McGraw Hill, have built or announced plans to use a similar setup [28].
- We assume fully replicated data center model where backup is taken care by popular replication mechanism [72].
- Each data center performs optimal workload consolidation and its power consumption is proportional to the workload being serviced in the data center [77].
- Failure of at most single data center has been considered. Failure of more than one data center simultaneously is unlikely since the choice of locations guarantee that no two sites share a common resource group.

- We assume that a data center has already been installed at a location that has the right renewable energy potential for a profitable deployment and provisioned with fixed generation capacity.
- We assume that it is possible to perform capacity planning based on intermittent renewable energy supply based on the studies and results reported in [14, 27, 28].
- Client demand at a location is proportional to the number of Internet users. Propagation delay within the client region is assumed to be negligible [2].
- All servers are homogeneous in capacity and capable of serving all types of requests (for heterogeneous applications) [64].

4.2.2 Definitions and Cost Model

In this section, we define the various parameters used in GCACP problem formulation and also define the cost models considered in the objective function. The notation used in this chapter is listed in Table 4.1.

Demand: Given that there is reasonably accurate information on workload across different client regions in a time window, let L_u^{ah} denote the total demand generated from a client region u for an application type a during the hour $h \in T$.

Delay: Let D_{su} be the propagation delay between user location u and data center location s. Let D_{max} be the maximum latency allowed for a client based on the SLA with the cloud provider. We also define a binary variable, y_{su} to represent the ability of data center s to serve requests from client region u.

Mean service rate: Let the mean service rate of each server be B bps and each application is characterized by a request generation rate and job size. To model, change in service rate before and after failure, we assume the mean file size for an application to vary between J_a^0 to J_a^1. The mean service rate of application type a

Input	Symbol	Description
Data center	p	Percentage of total servers failed at any data center
	M^{min}	Minimum number of servers at any data center
	M^{max}	Maximum number of servers at any data center
	α	Server acquisition cost
Client	L_u^{ah}	Total number of requests generated for application a from user location u during hour h
	D_{su}	Propagation delay between client region u and data center s
	D_{max}	The maximum tolerable latency
Server Utilization	B	Service rate of server in bits per second
	J_a^o	Mean file size of application a in kB
	γ_s^{fh}	Average server utilization at data center s during hour h and failed data center f
	γ^{max}	Maximum value of γ_s^{fh} to avoid waiting
Energy	θ_s^h	Electricity price per kWh at data center s at hour h
	G_s^h	Total available green energy at data center location s during hour h
	δ_s^h	Utility sell-back price at data center s during hour h
	E_{max}	Maximum capacity of the battery
	Z_D	Maximum energy that can be withdrawn from the battery (during hour h)
	Z_C	Maximum energy that can be supplied to the battery (during hour h)

Table 4.1: Summary of notation used in the chapter

can be defined as $\frac{B}{J_a^o}$ jobs per second, where o denotes the occurrence of failure (0 without failure, 1 with failure).

Average data center utilization: We have considered that the data center failure

at a site can be partial or complete, with p percentage of servers failing during an event of failure and therefore the available compute capacity reduces to $(1-p)m_s$. Assuming that requests are first placed in a common queue, before being served by any of the available servers, we define the average utilization, γ_s^{fh} to be

$$\gamma_s^{fh} = \begin{cases} \frac{\sum_{u,a} \lambda_{su}^{afh} J_a^o}{m_s B} & \forall s,h,f \neq s \\ \frac{\sum_{u,a} \lambda_{su}^{afh} J_a^1}{(1-p)m_s B} & f = s, p < 1 \\ 0 & f = s, p = 1 \end{cases} \tag{4.1}$$

i.e., data center offers degraded service to the applications after the failure at a site.

Power consumption: Let P_{idle} be the average power drawn in idle condition and P_{peak} be the power consumed when server is running at peak utilization. Then total power consumed at a data center location $s \in S$, at hour $h \in H$ is given by [16]

$$P_s^{fh} = m_s(P_{idle} + (E_s - 1)P_{peak}) + m_s(P_{peak} - P_{idle})\gamma_s^{fh} + \epsilon, \quad \forall s,h,f \tag{4.2}$$

where E_s is the PUE of a data center at s and ϵ is an empirical constant.

Power model: The power from various renewable energy generators is assumed to be known based on the meteorological data (may be obtained from [31]). G_s^h is the actual renewable energy generated (both wind and solar energy) at a data center s during hour h. For each data center s, hour h and failed data center f, we denote GU_s^{fh} as the renewable energy used, GS_s^{fh} as the renewable energy sold (net metering) and Z_s^{fh} as the energy supplied to the battery (when $Z_s^{fh} > 0$) or that consumed from the battery (when $Z_s^{fh} < 0$). Thus, the available green energy can be expressed as

$$G_s^h = GU_s^{fh} + Z_s^{fh} + GS_s^{fh} \qquad \forall s,h,f$$

We also define PB_s^{fh} as the total brown energy drawn from utility at a data center s during hour h and failed data center f. PB_s^{fh} is calculated as the difference between

the power consumed at a data center (P_s^{fh}) and the amount of renewable energy used (GU_s^{fh}), given by

$$PB_s^{fh} = P_s^{fh} - GU_s^{fh} \qquad \forall s, h, f \tag{4.3}$$

Battery model: For each data center s, during hour h, we denote E_{max} as the maximum battery capacity[1], and Z_D and Z_C as the maximum power that can be supplied and withdrawn, respectively. The battery level at any data center s, during hour h, after f^{th} data center failed is given by

$$E_s^{f(h+1)} = E_s^{fh} + Z_s^{fh} \qquad \forall s, h, f \tag{4.4}$$

Cost models: The TCO consists of the following components:

- *Server cost:* Let α be the server acquisition cost. The total cost of servers across the data centers is

$$\Phi = \alpha \sum_s m_s \tag{4.5}$$

- *Brown energy cost:* To account for the spatio-temporal variation in the electricity price, we define θ_s^h as the electricity price at location s during hour h of the day. We also express the cumulative brown energy cost across all data centers throughout the time horizon as

$$\Theta = \sum_{s,h,f} \theta_s^h PB_s^{fh} \tag{4.6}$$

- *Renewable energy sell-back revenue:* We denote the utility sell-back price by δ_s^h at a data center s during hour h. Let R be the cumulative on-site sell back revenue generated across all the data centers throughout the time horizon, defined as

$$R = \sum_{s,h,f} \delta_s^h GS_s^{fh} \tag{4.7}$$

[1]Energy storage devices are known to have enough capacity to power a data center at its maximum load between $5 - 30$ minutes [78].

4.2.3 Optimization Problem Formulation

Based on the cost factors and input parameters defined above, the GCACP problem is expressed as the following MILP.

$$\textbf{minimize} \quad \Psi = \Phi + \Theta + R \tag{4.8}$$

subject to,

$$GU_s^{fh} + Z_s^{fh} + GS_s^{fh} = G_s^h, \qquad \forall s, h, f \tag{4.9}$$

$$Z_s^{fh} \leq min\{Z_C, E_{max} - E_s^{fh}\}, \qquad \forall s, h, f \tag{4.10}$$

$$Z_s^{fh} \geq -min\{Z_D, E_s^{fh}\}, \qquad \forall s, h, f \tag{4.11}$$

$$\sum_{s \in S} \lambda_{su}^{afh} \;=\; L_u^{ah}, \qquad \forall u, h, f \tag{4.12}$$

$$2D_{su}\, y_{su} \leq D_{max}, \qquad \forall s, u \tag{4.13}$$

$$0 \;\leq\; \lambda_{su}^{afh} \;\leq\; y_{su} L_u^{ah}, \qquad \forall s, u, h, a, f \tag{4.14}$$

$$\gamma_s^{fh} \;\leq\; \gamma^{max}, \qquad \forall s, h, f \tag{4.15}$$

$$M^{min} \;\leq m_s \;\leq M^{max}, \qquad \forall s \tag{4.16}$$

$$0 \leq E_s^{fh} \leq E_{max}, \qquad \forall s, h, f \tag{4.17}$$

$$\lambda_{su}^{afh} \;=\; 0, \qquad \forall u, a, h, s = f \tag{4.18}$$

$$y_{su} \in \{0, 1\}, \qquad \forall s, u \tag{4.19}$$

Eq. (4.9) ensures that the sum of the green energy used to service workload at a data center, energy involved in battery charge (or discharge) and the surplus energy being sold to the grid is bounded by the available green energy. Eq. (4.10) limits the charging rate between the maximum charging rate and remaining capacity to be charged. Similarly, Eq. (4.11) is the constraint for the battery level. Eq. (4.12) ensures that the demand of all client regions in every hour is met. Eq. (4.13) ensures that the delay experienced by a client lies within the maximum delay bound.

Eq. (4.14) ensures that all the client requests are served by the data centers within the latency limit. Eq. (4.15) is used to limit the queuing delay at a data center, by bounding the average server utilization at each data center to $\gamma^{max} \in (0, 1]$ [16]. This constraint also ensures that in event of failure, workload assigned to a failed data center is bounded by its available compute capacity based on utilization defined in Eq. (4.1). Eq. (4.16) ensures that capacity limit of a data center (in terms of number of servers) is not exceeded. Eq. (4.17) is used to ensure that battery level is always non-negative and is limited by the capacity of the battery. Eq. (4.18) ensures that no request is served by a failed data center

Table 4.2 summarizes the decision variables used in the proposed optimization model.

4.3 Numerical Results

In this section, we evaluate the advantages obtained with the GCACP model under different scenarios. The proposed MILP framework is solved using CPLEX with Matlab on a server with Intel Xeon processor and 64 GB of RAM, running Ubuntu 14.04 (64 bit) OS. To understand the advantage of considering the operating cost in spare capacity provisioning, we compare the cost of solutions obtained using our model with that of the other two models (MS and CDN) as defined below.

- *MS model*: A rudimentary version of this model has been defined in [2]. The main objective is to minimize the total number of servers deployed across all the data centers.

- *CDN model*:In this model, the objective is to balance the load across data centers such that average response time is minimized.

We extended both the models with the same set of constraints as GCACP model (Eq. (4.9)-Eq. (4.19)) for fair comparison. After solving both the models to arrive

Variable	Description
m_s	Number of servers in data center s
λ_{su}^{afh}	Number of requests for application type a from user location u, served by data center s during hour h after f^{th} data center failed
y_{su}	Binary variable that denotes whether client location u lies within the latency bound of data center s
PB_s^{fh}	Brown energy required at data center s during hour h after f^{th} data center failed
GU_s^{fh}	Green energy used at data center s during hour h after f^{th} data center failed
GS_s^{fh}	Green energy sold at data center s during hour h after f^{th} data center failed
E_s^{fh}	Battery level at data center s during hour h after f^{th} data center failed
Z_s^{fh}	Energy supplied to/discharged from the battery at data center s during hour h after f^{th} data center failed

Table 4.2: Decision variables of the model

at the server and load distribution, we used the same cost factors (used for GCACP model) to calculate the TCO in each case. In the following, we present results based on evaluation of the three models to show the advantage of GCACP model.

4.3.1 Experimental Setup

To evaluate the models we used real-world meteorological data, workload trace, and electricity prices from utility companies.

Data center parameters: In our evaluation, we simulate a GDC composed of

six data centers located in the US. The locations are: California, Iowa, Arizona, Colorado, Oklahoma and Texas. Other parameters used are shown in Table 4.3. Electricity prices at these locations at different times are taken from [21].

Parameter	**Value**
Min. No. of Servers	200
Max. No. of Servers	50000
P_{idle}, P_{peak}	200 W,400W
Server acquisition cost	$2000 for 4 years
No. of clients	9
PUE	1.5
Max delay	50 ms

Table 4.3: Input parameters

Renewable energy availability: We used meteorological data from MIDC of National Renewable Energy Laboratory [31] to estimate the power generated from solar and wind energy based on hourly weather data. We used power generation models for wind and solar energy from [59] and [79]. We assumed that there are 200 NE-3000 wind turbines and 10,000 BP-MSX 120 solar panels except for Iowa and Oklahoma [17]. In order to account for full span of weather conditions and workload, we took quarterly average of renewable energy available for every hour of the day. Therefore, we have 96 time slots in our time horizon each spanning an hour.

Demand: We used trace of traffic from Wiki dump [68] to build the workload profile. We took quarterly average of client demand for every hour of the day. Considering the hourly workload, we distribute demand across different client locations proportional to the number of Internet users at each location. The peak demand from the trace is of the order of thousand requests per second, whereas literature suggests that data centers serve requests to the tune of million requests

per second [4]. Therefore we have upscaled the demand by a factor of 3000. We considered two types of applications with service rates as listed in Table 4.4 under the assumption that server processing rate is 1.6 MBps.

Type	Pre-failure		Post-failure	
	Reply size (KB)	Service rate(Req/s)	Reply size (KB)	Service rate(Req/s)
I	26	61	14	113
II	15	105	15	105

Table 4.4: Application service rates

4.3.2 Results

We present the results obtained from solving the GCACP model by varying the number of data centers, demand, latency bound, and failure percentage. The metric used to compare the three optimization models is the TCO as defined in Eq. 4.8. In all the plots, we show the normalized values of TCO seen with respect to the maximum TCO across all the models for an experiment.

TCO Comparison

In this experiment, we compare the TCO for GCACP model with the existing models. We increased the number of data centers from three to six with nine client regions and maximum latency of 50ms. Fig. 4.2 shows the normalized TCO for all the three models. We see that the TCO for GCACP model reduces with increasing number of data centers due to demand multiplexing and variations in the electricity prices and renewable energy availability. Though MS model uses demand multiplexing to reduce the number of servers, it is not cost-aware while multiplexing the demand and using the renewable energy. Therefore, as we see in Fig. 4.2, there is a slight increase in the cost with increasing number of data centers with MS model. We can observe that GCACP model achieves improvement upto 24% and 48% with

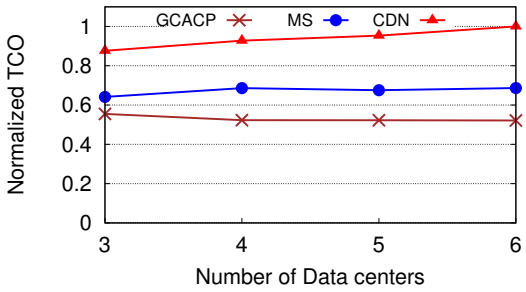


Figure 4.2: TCO vs number of data centers

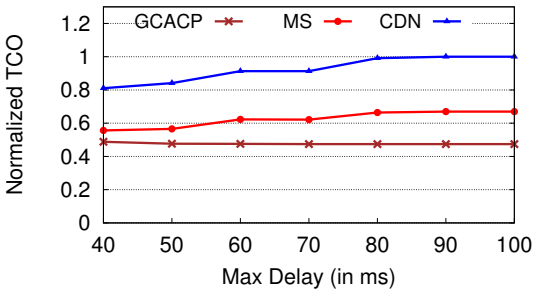


Figure 4.3: Variation in the TCO with latency bound

respect to MS and CDN models, respectively. We conclude that GCACP model is advantageous when green energy availability and electricity prices vary significantly across data centers in time, which appears to be the case in most of the real scenarios where data centers are geographically distributed .

Impact of Latency

In this experiment, we studied the impact of maximum latency bound on the TCO. We evaluate the models with six data centers and nine client regions. The maximum latency was chosen between 40 and 100 ms. Fig. 4.3 shows the normalized TCO for the three models. We notice that GCACP model achieves a reduction in the TCO of upto 29% and 52% with respect to MS and CDN models, respectively. In our model, we get more choice in the data centers capable of serving a client region with relaxed latency bound, which leads to better multiplexing of resources and exploitation of variation in the green energy prices. Although CDN model minimizes latency, its capacity provisioning is oblivious to both demand multiplexing and operating cost. Therefore the TCO is higher when latency requirement is not stringent. We conclude that under relaxed latency constraints GCACP model is more advantageous.

Impact of Demand

To understand the impact of increase in demand on TCO, we evaluate all the models for 6 data centers with varying demand (kx which is multiple of baseline demand x, where $k \in \{0.5, 1, 1.5, 2, 2.5\}$). From Fig. 4.4 it can be observed that with an increase in demand, TCO increases for all the models which is quite intuitive. Fig. 4.5 depicts the percentage reduction in the TCO for GCACP model with respect to the other models. It can be seen that with an increase in demand, the TCO decreases by 31% to 20% with respect to MS model, and by 62% to 38% with respect to the CDN model. This is mainly due to the fact that increasing the demand saturates data centers located in regions with cheaper electricity prices. The excess demand is then served from data centers in regions with higher electricity prices. We can conclude that GCACP model is advantageous when all the data centers do not operate at peak utilization.

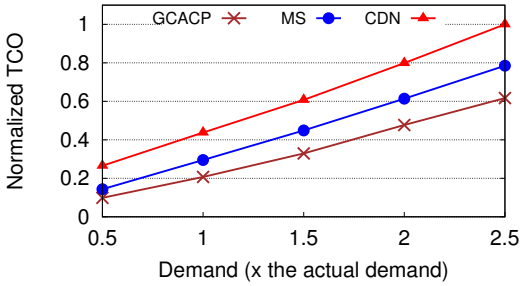


Figure 4.4: Variation in the TCO with demand

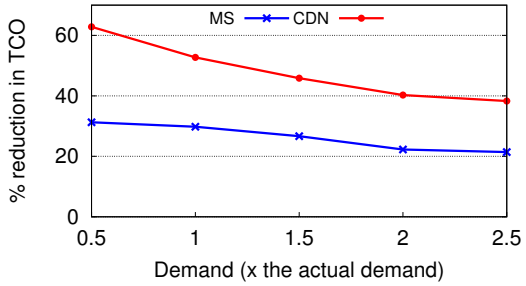


Figure 4.5: Percentage reduction in TCO with GCACP by varying demand

Impact of failure percentage

We evaluated the optimization models on six data centers with maximum latency of 50ms. We varied the fraction of servers failing at a data center and Fig. 4.6 shows the TCO with GCACP model. We notice that with increased failure rate,

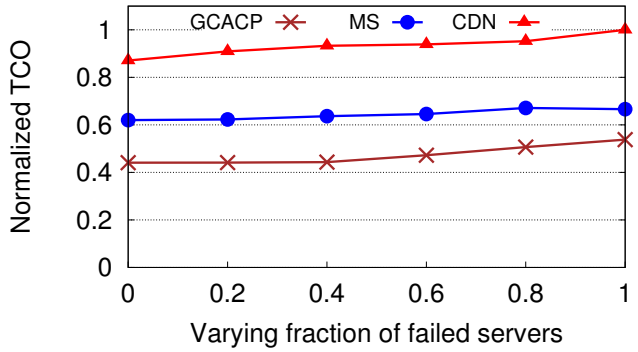


Figure 4.6: Variation in the TCO as the fraction of failed servers changes

the TCO increases due to the fact that more compute servers are required to mask failure. However, we can see that even if a data center fails completely, GCACP solution gives about 19% and 46% lower cost with respect to MS and CDN models, respectively. We conclude that with partial failure, GCACP model is advantageous because of lower spare capacity requirement and lower brown energy consumption (since the available renewable energy is constant).

4.4 Conclusion

We investigated the problem of capacity planning for fault-tolerant data centers powered by renewable energy sources. We designed an optimization model to provision the spare capacity and allocate requests with least TCO. We used real-world data to compare the proposed GCACP model against the baseline models considering various scenarios. We conclude that the GCACP model achieves cost reduction of upto 24% and 48% with respect to MS and CDN models, respectively. We also conclude that the GCACP model is beneficial for designing green data centers, when the available green energy and the electricity prices vary widely, and the utilization of data center is not near its peak.

In this chapter we assumed that renewable energy sources are collocated with

data centers. In the next chapter, other ways of greening data center are also considered. Considering the fact that the data center operators try to gradually increase their renewable energy usage, we model the problem of spare capacity provisioning to satisfy a target green energy usage at a minimal cost, when the data center is powered by a combination of brown and green energy sources.

Chapter 5

Optimizing Energy Cost in Fault-tolerant GDCs Satisfying Green Energy Bound

5.1 Introduction

Owing to huge installation cost associated with renewable energy sources, major operators set a target of partial renewable energy integration every year [29]. For example, Facebook targeted being 25% powered by green energy by the end of 2015 [30]. Many GDC operators start with partial renewable energy bound *i.e.*, the green energy used should be at least some fraction of the total power demand. We call this *green energy bound* in this thesis. Therefore, for cost-aware capacity provisioning, we consider both green energy cost and brown energy cost, while satisfying green energy bound along with the other constraints. In this chapter, we consider the cost of green energy procurement while optimizing the operating cost for the GDCs powered by both brown and multiple renewable energy sources.

We address the problem of capacity provisioning while forcing the green energy

usage which is essential for sustainable green data center design [11]. We consider a generalized failure model that can accommodate both partial and complete failures. It is mostly observed that the frequency of partial data center failure is very high, while complete data center failure is rare (may be once in two years) [80].

Our work is the first one to consider the real-time price of electricity (while enforcing a minimum green energy usage), to design cost-efficient fault-tolerant GDC. We provision spare capacity across the data centers so that the demand is met even after the failure at a data center site (either partial or complete), while minimizing the TCO. We consider both green and brown energy cost in minimizing the operational cost. We model the problem using MILP, where the main constraints include: green energy usage bound, latency bound, and the failure probability at a site (partial or complete). Solving our model gives the optimal server distribution across the sites and the demand distribution that minimizes the TCO.

The rest of the chapter is organized as follows. Section 5.2 discusses the optimization model. Section 5.3 presents the numerical results that demonstrate the impact of green energy cost on the TCO. The chapter is concluded in Section 5.4.

5.2 Optimization Model

In this section, we formulate the green energy cost-aware, capacity provisioning problem (termed GACED) for fault-tolerant GDCs as a constrained optimization problem. Before that, we discuss the architecture of the data center powered by different renewable energy sources considered in our work, followed by the assumptions and input parameters used.

We considered a GDC powered by multiple renewable energy sources as illustrated in Fig. 5.1. There are $|S|$ data centers housing m_s number of servers (we use the index s for data center). To achieve carbon footprint and energy cost reduction, each data center is integrated with multiple green energy sources such as,

on-site renewable generator (wind and/or solar), off-site renewable generator, PPA, and utility power transmitted through grid. ESD is used to store surplus green energy. There are front-end proxy servers collocated with each client region, which map the requests to multiple data centers as shown in Fig. 5.1. L_u^{ah} denotes the demand generated per hour (h) from each client region u, corresponding to each application type a. λ_{su}^h denotes the number of requests mapped from client region u to data center s at hour h for application type a. In this model, m_s and λ_{su}^h are the decision variables while other parameters like L_u^h, brown electricity price and green energy availability are the input parameters.

5.2.1 System Architecture

The following assumptions are used in the model.

- Each data center consolidates the workload to keep the power consumption proportional to the workload served.
- Service time (including queuing delay) is same for all the data centers and the latency is due to the propagation delay.
- The requests are placed in a single queue to be served by any server.

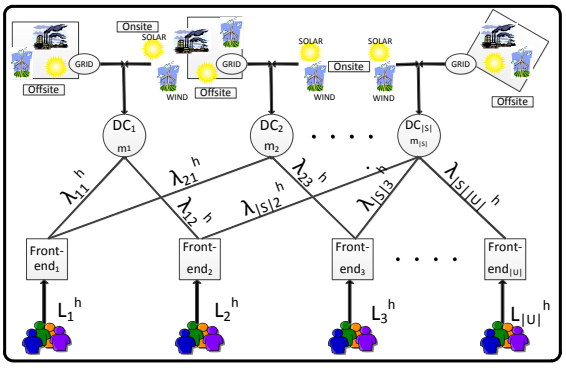


Figure 5.1: Architecture of the GDC powered by multiple green energy sources

- Only one data center can completely or partially fail at any point in time [2]. Failure of more than one data center at the same time is avoided by the choice of locations.

5.2.2 System Model

Demand: Let S be the set of data centers housing m_s number of servers. Let L_u^{ah} denote the demand from a client region u during hour h for application type a. Let λ_{su}^{afh} denote the number of requests mapped from client region u to data center s ($s \in S$), at hour h for an application type a. Here $f = 0$ indicates the case of no data center failure and $f \in 1, 2, \ldots, |S|$ indicates the failure of f^{th} data center.

Heterogeneous workload: Since we assume that a data center can serve different types of workload, we explicitly considered the heterogeneity while calculating the server utilization. Let the processing rate of the server be B bps and the mean job size be J_a for application type a. The effective service rate is $\frac{B}{J_a}$. We define the average utilization as

$$\gamma_s^{fh} = \frac{\sum_{u,a} \lambda_{su}^{afh} J_a}{m_s B} \tag{5.1}$$

Failure model: Let p be the fraction of servers failing at any given site. The processing rate of a failed data center reduces to $(1-p)m_s B$. The data center utilization after the failure can be expressed as

$$\gamma_s^{fh} = \begin{cases} Eq.(5.1) & \forall f \neq s \\ \frac{\sum_{u,a} \lambda_{su}^{afh} J_a}{(1-p)m_s B} & f = s, p < 1 \\ 0 & f = s, p = 1 \end{cases} \tag{5.2}$$

Delay: Let D_{su} be the propagation delay between the data center s and the client region u. We define a target delay of D_{max} for all types of workloads (with different processing rate), when no data center has failed and D_{max}^f ($D_{max}^f \geq D_{max}$) for the

case of a data center failure. We use a binary variable, y_{su}^{f} to indicate the ability of data center s to serve the requests from client region u when data center f has failed.

Power consumption: Let P_{idle} be the average power drawn in idle condition and P_{peak} be the power consumed when server is running at peak utilization. The total power consumed by $s \in S$, at hour $h \in H$ is modeled as [16]

$$P_s^{fh} = m_s(P_{idle} + (E_s - 1)P_{peak}) + m_s(P_{peak} - P_{idle})\gamma_s^{fh} + \epsilon, \quad (5.3)$$

where E_s is the PUE of a data center at s and ϵ is an empirical constant.

Modeling brown energy usage: Let θ_s^h be the price of brown energy at a data center s during hour h and δ_{si}^h be the price of green energy of type i, $i \in \{1, 2, 3, 4, 5\}$ corresponding to onsite wind, offsite wind, onsite solar, offsite solar and PPA, respectively. Let PB_s^{fh} denote the amount of brown energy drawn at hour h and Δ_{si}^{fh} denotes the amount of renewable energy drawn from source i. Since the brown energy is used only after exhausting the green energy available, the brown energy drawn from the grid is given by

$$PB_s^{fh} = P_s^{fh} - \sum_i \Delta_{si}^{fh} \qquad \forall s, h, f \quad (5.4)$$

5.2.3 Cost Model

Next, we define the cost components used in the objective function of the MILP formulation.

- **Server cost:** Let α be the cost of acquiring a server. The total cost of the servers in all the data centers is

$$\Phi = \alpha \sum_s m_s \quad (5.5)$$

- **Brown energy cost:** The cost of brown energy consumed across all the data centers is given by

$$\Theta = \sum_{s,h,f} \theta_s^h P B_s^{fh} \tag{5.6}$$

- **Renewable energy cost:** The total cost incurred in using renewable energy across all the data centers is given by

$$R = \sum_{s,h,f} \delta_{si}^h \Delta_{si}^{fh} \tag{5.7}$$

MILP model: The objective for capacity provisioning in fault-tolerant green GDC is to minimize the TCO, denoted by F, which is simply the sum of all the aforementioned costs while satisfying constraints on delay, green energy usage and availability. Formally, the problem is expressed as

$$\textbf{minimize } F = \Phi + \Theta + R; \tag{5.8}$$

subject to

$$\sum_{s,i,h} \Delta_{si}^{fh} \geq \rho P_s^{fh}, \qquad \forall f \tag{5.9}$$

$$\sum_{s} \lambda_{su}^{afh} = L_u^{ah}, \qquad \forall u, a, h, f \tag{5.10}$$

$$2 D_{su}\, y_{su}^f \leq D_{max}, \qquad \forall s, u, f = 0 \tag{5.11}$$

$$2 D_{su}\, y_{su}^f \leq D_{max}^f, \qquad \forall s, u, f \geq 1 \tag{5.12}$$

$$0 \leq \lambda_{su}^{afh} \leq y_{su}^f L_u^{ah}, \qquad \forall s, u, a, h, f \tag{5.13}$$

$$\gamma_s^{fh} \leq \gamma^{max}, \qquad \forall s, h, f \tag{5.14}$$

$$M^{min} \leq m_s \leq M^{max}, \qquad \forall s \tag{5.15}$$

$$\lambda_{su}^{afh} = 0, \qquad \forall u, a, h, s = f \tag{5.16}$$

$$y_{su}^f \in \{0, 1\}, \qquad \forall s, u, f \tag{5.17}$$

Among the constraints, Eq. (5.9) ensures that the power from green energy sources is at least ρ percentage of the total power demand. Eq. (5.10) makes sure that the demand during every hour is met. Eq. (5.11), (5.12), and (5.13) ensure that all types of workload are served within the latency bound (before and after failure). Eq. (5.14) is used to limit the queuing delay by bounding the average server utilization to $\gamma^{max} \in (0, 1]$. It also ensures that workload assigned to a failed data center is bounded by its capacity. Eq. (5.15) ensures that capacity limit of a data center (in terms of number of servers) is not exceeded. Eq. (5.16) ensures that no request is served by a failed data center.

The decision variables in the MILP are: m_s, the number of servers in a data center s, λ_{su}^{afh}, the number of requests from client region u mapped to data center s at hour h for workload type a, and Δ_{si}^{fh}, the renewable energy drawn from source i.

5.3 Numerical Results

The proposed MILP model (termed GACED) is solved centrally using CPLEX and MATLAB tools on a Linux server with Intel Xeon processor and 64 GB of RAM. Since spare capacity provisioning in data centers is a one-time effort at the time of design, the running time is not a matter of concern. We obtained the meteorological data for three locations: Texas, Illinois, and California from [31]. The server processing rate is set to 1.6 MBps. Two types of workload are considered with the mean job size 13KB and 26KB, and the service rate of 120 and 60 requests/sec. D_{max} and D_{max}^{f} are set to 40 ms and 80 ms, respectively. PPA price for TX, CA, IL is taken as 8, 8, 4 cents/kWh, respectively. All the other parameters are same as those mentioned in 3.3.1. The client demand was generated from traces of Wikipedia [68]. We chose nine regions of USA: Illinois, Tennessee, New York, Arizona, Massachusetts, California, Florida, Missouri, and Louisiana. The demand at each location was proportional to the number of Internet users [67].

Table 5.1 reports on-site and off-site renewable energy sources at each location with the corresponding average capacity factor (CF), defined as the ratio of the actual power output to the maximum rated capacity.

	Source	**Location**	**Avg CF(%)**
Onsite	Wind	California	27
	Wind	Illinois	32
	Wind	Texas	33.6
	Solar	California	22.8
	Solar	Texas	23.46
Offsite	Wind	Arizona	30
	Wind	Colorado	43
	Wind	Iowa	34
	Solar	Arizona	27.74
	Solar	Colorado	24.61

Table 5.1: Capacity factor for various green energy sources

We used the models from NREL [79] and [59] for solar and wind energy generation, respectively. Based on the meteorological data from NREL [31], we calculated the total power generated. At each site, we considered 20 wind turbines of capacity *1.5MW*, each and 10,000 solar panels of *120W*, each. The cost of generating wind and solar power is obtained by taking the installation cost of 1630$/kW and 3100$/kW, and life time of 20 yrs and 25 yrs, respectively [14]. We took quarterly average of client demand and renewable energy generated for every hour of the day. The brown electricity price at different locations is taken from US energy information administration website [21].

5.3.1 TCO Comparison

To show the TCO reduction with our model, we designed a baseline model (termed CED-B) that minimizes the TCO when the data centers are powered only with brown energy keeping the other constraints same. For a full-site failure scenario, Fig. 5.2 shows the percentage gain in the TCO using GACED model compared to the CED-B model. Even after failure, the gain is 2% with the green energy usage of 20%. The gain reduces with increase in green energy usage because, our model increases the amount of (expensive) green energy purchased to satisfy the constraint. On the other hand, CED-B model has no cost from green energy usage. With the GACED model, greening can be achieved with very little or no extra cost unless we target high renewable energy usage.

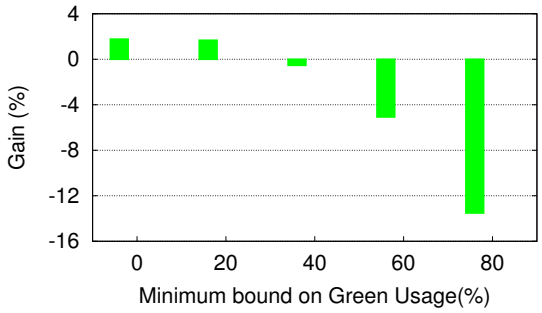


Figure 5.2: Impact of varying green energy usage on the TCO of GACED model

5.3.2 Impact of Failure Percentage

In this experiment we studied the impact of varying the failure percentage on the TCO while green usage bound is satisfied. Fig. 5.3 compares the TCO for the two models while forcing 40% green energy usage. We see that the TCO is almost similar for both the models due to the fact that, the GACED model optimally uses cheaper renewable energy to reduce the TCO. Fig. 5.4 illustrates the offsite wind energy usage in the GACED model and its corresponding price at the Texas data center.

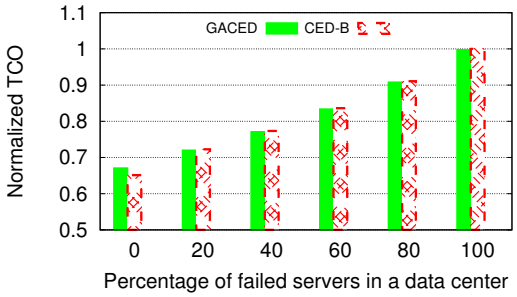


Figure 5.3: Impact of varying failure percentage on the TCO of GACED model

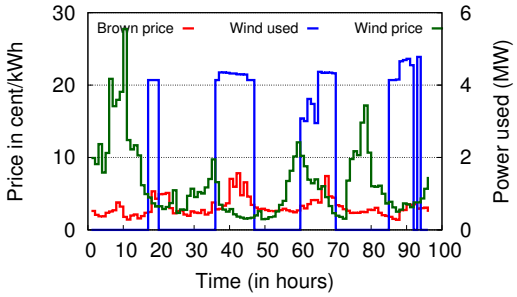


Figure 5.4: Illustration of wind energy usage

When the wind energy is cheaper, GACED uses more of it to maintain the same TCO (as with CED-B), albeit with reduced carbon footprint. Due to intelligent usage of green energy, GACED meets the target renewable energy usage of 40% at all times. We conclude that the GACED model can lead to greener data center deployment with no or little additional cost (though green energy procurement is costlier).

5.3.3 Impact of Demand

To understand the impact of increase in demand on TCO, we evaluate the models for 6 data centers with varying demand. The demand is varied as a multiple of baseline demand. The multiplicative factor is varied between 1 and 5. Fig. 5.5 shows the impact of increase in demand on the TCO and green energy procurement decision. We set green energy usage and failure percentage to 40% and 20%, respectively. As demand increases, the TCO for both the models increases due to obvious reasons. However, TCO with GACED model increases with the demand due to the green energy usage constraint. We note that, even with five-fold increase in the demand (with a cost of almost 30 cents/kW for wind energy at Texas), it is possible to meet the 40% green energy usage constraint with a meagre 4% increase in the TCO.

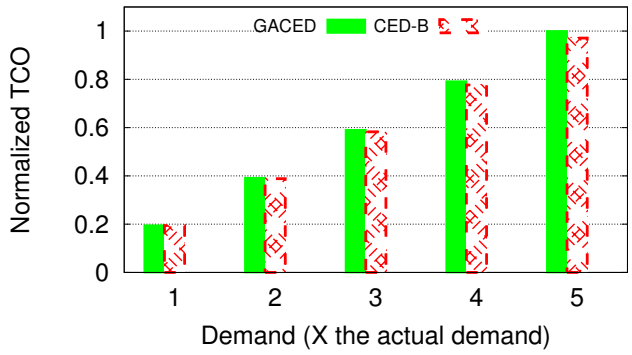


Figure 5.5: Impact of demand variation on the TCO of GACED model

5.3.4 Impact of Latency

In this experiment we studied the impact of relaxing the latency bound on the TCO. Fig. 5.6 shows the impact of relaxing latency requirement on the TCO. D^f_{max} is set to twice D_{max}. We notice that both the models reduce the TCO for relaxed latency bound, since there is more choice in selecting the data center to serve the requests. However, GACED model lowers the TCO by considering locations powered by cheaper green energy.

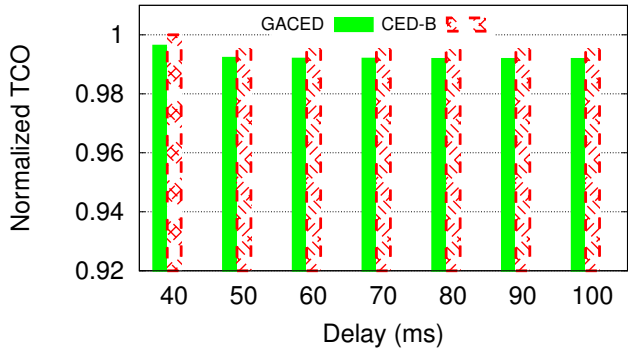


Figure 5.6: Impact of latency relaxation on the TCO of GACED model

5.3.5 Sensitivity Analysis

We quantitatively evaluate the impact of uncertainty in the renewable energy availability on the performance of GACED model. For the case of complete data center failure and 40 % green energy usage requirement, the capacity factor was varied between -40% and 40%. Fig. 5.7 shows the percentage gain in the TCO with GACED model (compared to the CED-B model). Since, the cost of renewable energy decreases with increasing capacity factor [14], the GACED model has lower TCO compared to the CED-B model (by about 4%). This is because it efficiently exploits cheaper green energy.

We also conducted another experiment here to demonstrate that, if the forecasted green energy availability is inaccurate, we can work with 20% green energy usage requirement, where GACED model has only 3% higher TCO. To do this, we vary the capacity factor by -40% and the result is shown in Fig. 5.8.

The cost of generating renewable energy tends to reduce with time due to the technological improvements in the equipment efficiency and the increasing deployment. Harnessing green energy depends upon the cost and efficiency of technology, which is constantly improving thereby reducing the cost. In order to understand the impact of long term prediction of green energy cost on both models,

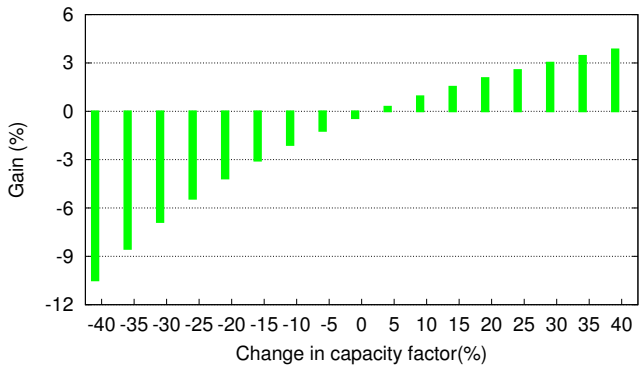


Figure 5.7: Gain in TCO for GACED and CED-B models varying CF

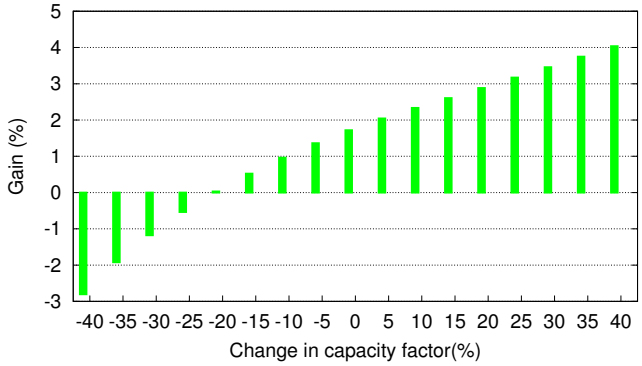


Figure 5.8: Percentage gain in the TCO with GACED model (compared to CED-B model) with varying CF and 20% green energy usage

we optimistically assume an yearly reduction of 10% in green energy cost [47]. Therefore, after 10 years the cost will effectively decrease by 65 %. Considering this reduction in the price after 10 years, we evaluate both the models by varying green energy bound and assuming complete data center failure. Fig. 5.9 shows the gain in the TCO with GACED model (compared to CED-B model) with increasing green energy usage. We notice that the reduction in the TCO with GACED model will be atleast 10 % even if 80% of the power used is from green energy sources. Hence, our model is sustainable and economically viable with improvement in technology and capacity factor (reduction in the price of renewable energy).

5.4 Conclusion

We used MILP to formulate the cost-aware capacity provisioning problem for fault-tolerant data centers ensuring a minimum green energy usage (GACED model). The proposed model outperforms the baseline model (CED-B) which minimizes the TCO considering only brown energy. Results demonstrate that even with renewable energy integration, the TCO can be lowered with GACED model, despite green energy being costlier. Even forcing a green energy usage of upto 80% leads to

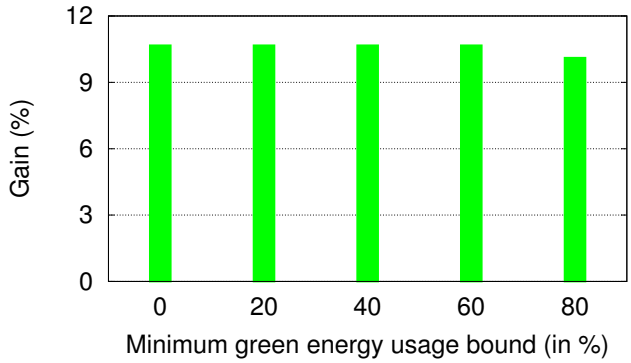


Figure 5.9: Percentage gain in the TCO with GACED compared to CED-B considering reduced cost of green energy after 10 years

an additional cost of only 15% compared to the CED-B model. The proposed model optimally schedules demand considering the availability of green energy and its price variation to minimize the TCO. We conclude that with an appropriate model, green energy integration lowers the cost of designing fault-tolerant GDCs with reduced carbon footprint. Our model would be further beneficial with an improvement in technology leading to larger capacity factor (lower renewable energy cost). Therefore, we expect that our work can help data center operators make informed decision about capacity planning in presence of green energy usage target and variation in the electricity prices, demand, and failure rate. In the next chapters, we address the problem of distributed load balancing in fault-tolerant GDCs which have already been provisioned with spare capacity to counter failures. We formulate the load balancing problem with an objective of minimizing the operating cost while ensuring that the latency constraints are met.

Chapter 6

Game-theoretic Model for Load Balancing in Fault-tolerant GDCs

6.1 Introduction

As discussed earlier, the operational cost of a data center is influenced by factors such as electricity prices, server/data center failure, green energy availability, and client demand. Therefore, geographical load balancing is challenging as the policy must consider the spatio-temporal variation in these factors to minimize the operating cost. Most of the literature on load balancing in GDCs considered minimizing the operating cost, which is profitable for the operators, but has ignored the users' perspective [4, 32]. Users consuming the same resources may pay the same price, but experience variable delays. From a business perspective, ensuring fairness in service latency across the requests from different clients is also important. The load balancing algorithms in the literature designed to provide fairness, did not consider the operating cost. Therefore, we consider the linear combination of operating cost (or energy cost) and revenue loss due to latency (including the network and queuing delays) as the objective function.

6.1 Introduction

In a GDC, the user requests are served by the front-end proxy servers independent of each other. Each proxy server prefers to get its requests served by the data center first to minimize the service delay. In order to model this selfish nature in distributed load balancing, we use the non-cooperative game theory approach. Load balancing and resource management in distributed systems have been addressed using non-cooperative game theory in earlier works like [81] and [82]. The work in [81] addressed the problem of load balancing in a system consisting of n computers (or nodes) shared by m clients (classes). They proposed a distributed algorithm for load balancing using non-cooperative game theory. The work in [81] also considered that the jobs submitted to a heavily-loaded computer are transferred to other lightly-loaded computers. Their objective function included communication delay along with the computational delay. In their model, a player has a selfish interest of optimizing his/her own expected response time. The problem is formulated as a non-cooperative game among the users, who try to minimize the expected response time of their own jobs.

We propose a game-theoretic distributed load balancing algorithm, that is executed across a finite number of front-end proxy servers. The objective of the game is to minimize the sum of the operating cost and the revenue loss due to delayed service. The proposed approach reduces the cost compared to an earlier approach [59] that only minimizes the operating cost. In summary, the main contributions of this work are as follows.

- For the first time, we model the load balancing in GDCs as a non-cooperative game among the front-end proxies. We consider the spatio-temporal variation in the electricity price, the offered load, and the availability in the model. We prove that the Nash equilibrium is the solution of this game, which is guaranteed to exist since the proposed objective function is continuous, convex and increasing [34, 35]. We characterize the Nash equilibrium and propose a

distributed algorithm for computing the same.

- We evaluate the performance of the non-cooperative game theoretic algorithm (abbreviated as NCG) along with the existing ones, such as the proportional scheme and the global optimal scheme, using real-world data. The proposed NCG algorithm shows better fairness (in service latency) at a comparable cost.

The rest of this chapter is organized as follows. In Section 6.2 we present the architecture and the cost model used in the formulation. We present the non-cooperative game model and derive the structure of Nash equilibrium in Section 6.3. A distributed algorithm to solve the game and its analysis are given in Section 6.4. In Section 6.5, we present numerical results that compare the performance of our algorithm against the optimal approach. Section 6.6 concludes the chapter.

6.2 System Model

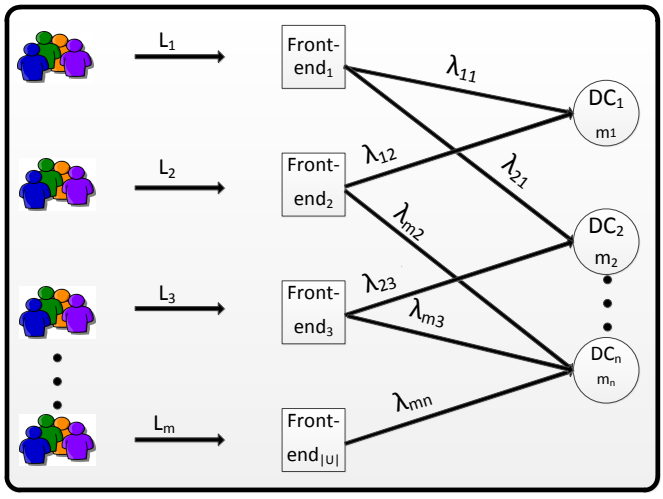


Figure 6.1: Illustration of the architecture of GDC used in the model

Fig. 6.1 shows the architecture of the GDC used in our model. It shows a set of n data centers denoted by S and a set of m front-end proxy servers denoted by

U. For simplicity, we consider the client regions to be co-located with the front-end proxy servers. Each data center houses m_s number of servers and is modeled as an M/M/1 queueing system. It is characterized by an expected (or average) processing rate $\mu_s = m_s\mu$, where μ is the processing rate of each server, $s = 1, ...n$. The arrival rate of demand from each client region is represented by L_u, $u = 1, ...m$. A front-end proxy server maps the client requests to multiple data centers, where λ_{su} is the portion of the demand mapped from a client region u to a data center s. The utilization of a data center, denoted by η, is defined as

$$\eta = \sum_u \lambda_{su}/\mu_s \; m_s \tag{6.1}$$

The following conditions need to be satisfied.

- To ensure that all the requests are served,

$$\sum_s \lambda_{su} = L_u \qquad \forall u \tag{6.2}$$

- Since the number of requests served cannot be negative,

$$\lambda_{su} \geq 0 \qquad \forall s, u \tag{6.3}$$

- To ensure the stability of the system,

$$\sum_u \lambda_{su} < \mu_s \quad \forall s \tag{6.4}$$

Power consumption cost: As reported in [83], the power consumption of a server varies linearly with the load. Let P_{idle} be the average power drawn by the server in idle condition, P_{peak} be the power consumed at the peak utilization, and E_s be the PUE of a data center s. The data center power consumption includes three components: the power consumed by idle servers, given by $m_s(P_{idle})$; the power consumed by the servers operating at a utilization η, given by $m_s(P_{peak} - P_{idle})\eta$;

and the power consumed by the cooling and auxiliary equipment, given by $(m_s(E_s - 1)P_{peak})$. Therefore, the total power consumed at a data center location $s \in S$ can be expressed as [16],

$$P_s = m_s(P_{idle} + (E_s - 1)P_{peak}) + \sum_u \lambda_{su} m_s (P_{peak} - P_{idle})/\mu_s m_s \tag{6.5}$$

Eq. (6.5) can also be expressed as an affine function of the total workload at a data center as

$$P_s = \frac{(P_{peak} - P_{idle}) \sum_u \lambda_{su}}{\mu_s} + \epsilon' \tag{6.6}$$

where $\epsilon' = m_s(P_{idle} + (E_s - 1)P_{peak})$.

Due to the recent advances in hardware, DVFS scaling, and processor scheduling, it is possible to keep the power consumption proportional to the utilization [84]. Due to efficient cooling systems, PUE has also been significantly lowered [85]. The state-of-the-art average PUE could be as low as 1.02 [86]. When PUE is close to unity and P_{idle} is very low compared to P_{peak} [87], ϵ' is very low compared to the actual power consumed. Therefore, we assume $\epsilon' = 0$ [8, 36].

This assumption makes Eq. (6.6) to be linear in terms of the load. Thus, we can easily determine the power consumed to serve the requests λ_{su} from a client region u at a data center s as

$$P_{su} = \frac{(P_{peak} - P_{idle})\lambda_{su}}{\mu_s} \tag{6.7}$$

Given the load λ_{su} and a unit electricity price ρ_s, the cost incurred due to the power consumed at a data center (also termed the operating cost in this chapter), is given by

$$\Theta_{su} = \rho_s \frac{(P_{peak} - P_{idle})}{\mu_s} \lambda_{su} \tag{6.8}$$

$$\Theta_{su} = \theta_s \lambda_{su} \tag{6.9}$$

where $\theta_s = \rho \frac{(P_{peak} - P_{idle})}{\mu_s}$ is constant.

Delay cost: Let d_{su} be the propagation delay between the data center s and a client region u. We assume an M/M/1 model for the queue at the proxy, so that the expected average queuing delay at a data center s for a request from a client region u is given by

$$D_{su} = \frac{1}{\mu_s - \sum_u \lambda_{su}} \tag{6.10}$$

Therefore, the total delay incurred by a request, δ_{su} is given by

$$\delta_{su} = d_{su} + D_{su} \tag{6.11}$$

Since we assumed Web workload, the transmission delay is neglected. We use a linear model for the loss in revenue due to the delay incurred [62]. The cost incurred due to the delay experienced by a request λ_{su} of client region u at a data center s, denoted by Δ_{su} is given by

$$\Delta_{su} = \beta \lambda_{su} \delta_{su} \tag{6.12}$$

$$= \beta \lambda_{su} (D_{su} + d_{su})$$

$$= \beta \left(\frac{\lambda_{su}}{\mu_s - \sum_u \lambda_{su}} + d_{su} \lambda_{su} \right) \tag{6.13}$$

where β is a constant.

6.3 Load Balancing as a Non-cooperative Game

In this section, we model the load balancing problem as a non-cooperative game. In a non-cooperative game there could be finite (or infinite) number of players who try to minimize/maximize their objective independently, but eventually reach an equilibrium. For a finite number of players, this equilibrium is called Nash equilibrium, whereas for infinite number of players this equilibrium is called Wardrop equilibrium [34].

In distributed load balancing, the problem at hand for each front-end proxy server is to determine λ_{su}, which we model as a non-cooperative game among the

front-end proxies. We define the vector $\boldsymbol{\lambda_u} = (\lambda_{1u}, \lambda_{2u}, ..., \lambda_{nu})$ as the load balancing strategy of a user $u, u = 1, 2, ..., m$ and the vector $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_m)$ as the load balancing strategy profile of the entire system.

Objective function: The objective function we use has two components, the power consumption cost defined in Eq. 6.9 and the delay cost defined in Eq. 6.13. Let Ψ_u denote the expected cost incurred for a client region (CIC). We define the objective function for a front-end proxy serving a client region u as

$$\Psi_u(\boldsymbol{\lambda}) = \sum_{s=1}^{n} (\Theta_{su} + \Delta_{su}) \tag{6.14}$$

$$\Psi_u(\boldsymbol{\lambda}) = \sum_{s=1}^{n} \Big(\theta_s \lambda_{su} + \beta \lambda_{su} (\frac{1}{\mu_s - \sum_i \lambda_{si}} + d_{su}) \Big) \tag{6.15}$$

The goal of a front-end proxy at u is to find a feasible load balancing strategy $\boldsymbol{\lambda_u}$ such that $\Psi_u(\boldsymbol{\lambda})$ is minimized. The strategy of u depends on the strategies of other front-end proxies as Ψ_u is a function of $\boldsymbol{\lambda}$.

Definition 6.1. *Feasible strategy profile is a strategy $\boldsymbol{\lambda}$ that satisfies the following*

1. *Positivity:* $\lambda_{su} \geq 0, \quad \forall s, u;$
2. *Conservation:* $\sum_s \lambda_{su} = L_u \quad \forall u;$
3. *Stability:* $\sum_u \lambda_{su} < \mu_s \quad \forall s;$

Definition 6.2. *The non-cooperative load balancing game is a game played among a set of players. Each player has a set of strategies and an associated cost with each strategy. The game in our scenario is a normal form game with continuous objective function and can be described as follows.*

- *Players: A finite set of players m denoted as U,* $U = \{1, 2, \ldots, m\}$
- *Strategy sets: Strategy sets:* $\lambda_u = \lambda_{1u}, \ldots, \lambda_{su}, \ldots, \lambda_{nu}$ *where,*
  $\lambda_{su} \in [0, L_u]$ $\forall u \in \{1, 2, \ldots, m\}$, $\forall s \in \{1, 2, \ldots, n\}$, *s.t.* $\sum_s \lambda_{su} = L_u \quad \forall u$

- *Cost: The cost of a player u is represented by Ψ_u. Each player wants to minimize the cost.*

Claim 1. *We can get an upper bound on the objective function in Eq. 6.15, denoted by $U_{Bnd}(\Psi_u)$.*

Proof: From Eq. 6.15 we have

$$\Psi_u(\boldsymbol{\lambda}) = \sum_{s=1}^{n} \Big(\theta_s \lambda_{su} + \beta \lambda_{su}(\frac{1}{\mu_s - \sum_i \lambda_{si}} + d_{su})\Big) \tag{6.16}$$

$$\Psi_u(\boldsymbol{\lambda}) = \sum_{s=1}^{n} \theta_s \lambda_{su} + \beta \sum_{s=1}^{n} \lambda_{su}(\frac{1}{\mu_s - \sum_i \lambda_{si}}) + \beta \sum_{s=1}^{n} \lambda_{su} d_{su} \tag{6.17}$$

Note that, according to Eq. 6.2 we have $\sum_s \lambda_{su} = L_u$.

Therefore, all three terms of Eq. 6.17 can be expressed as

$$\sum_{s=1}^{n} \theta_s \lambda_{su} = L_u \sum_{s=1}^{n} \theta_s \tag{6.18}$$

$$\beta \sum_{s=1}^{n} \lambda_{su}(\frac{1}{\mu_s - \sum_i \lambda_{si}}) = \beta L_u \sum_{s=1}^{n}(\frac{1}{\mu_s - \sum_i \lambda_{si}}) \tag{6.19}$$

$$\beta \sum_{s=1}^{n} \lambda_{su} d_{su} = \beta L_u \sum_{s=1}^{n} d_{su} \tag{6.20}$$

A lower bound of the objective function is zero because all terms are non-negative in the Eq. 6.15. We can get an upper bound of the objective function by noting the sum of the three terms above is less than $L_u[\sum_{s=1}^{n} \theta_s + \beta + \beta \sum_{s=1}^{n} d_{su}]$. Each of the variables L_u,θ_s,d_{su} has an upper bound, which can be used to get an upper bound $U_{Bnd}(\Psi_u)$ for the objective function.

Typically, the payoff or cost function used in normal form game is maximization. Even though in our game we are minimizing the cost function, we can transform the minimization objective into maximization as discussed further.

Claim 2: *It can be shown that the proposed game is equivalent to a game where the objective function is maximized.*

Proof. We note that minimizing Ψ_u is equivalent to maximizing $U_{Bnd}(\Psi_u) - \Psi_u$. Thus, our game definition with this objective is in normal form as the players maximize the objective function: $\Phi = U_{Bnd}(\Psi_u) - \Psi_u$ [88].

In order to obtain the load balancing strategy for the GDC, the above game has to be solved. The Nash equilibrium is the most commonly used solution for such games.

Definition 6.3. *Nash equilibrium of the above mentioned load balancing game is a load balancing strategy* $\boldsymbol{\lambda}$ *such that, for every front-end proxy* u

$$\boldsymbol{\lambda_u} \in \arg\min_{\boldsymbol{\lambda_u}} \Psi_u(\boldsymbol{\lambda_1}, .., \boldsymbol{\lambda_u}, .., \boldsymbol{\lambda_m}) \tag{6.21}$$

A strategy $\boldsymbol{\lambda}$ is a Nash equilibrium if no player can gain by changing its current strategy to another feasible one. In our load balancing game, the Nash equilibrium has the property that no player can decrease the total cost incurred by choosing a different load balancing strategy $\boldsymbol{\lambda_u}$ given the other players' load balancing strategies. The Nash equilibrium exists for our game because Ψ_u is continuous, convex and increasing. At the Nash equilibrium, the strategy profile is such that every player's load balancing strategy is a best reply given the other players' strategies. This best reply for a player provides a minimum cost for that player's demand given the other players' strategies. Thus, we first determine the best reply strategy $\boldsymbol{\lambda_u}$ for every player u and then we determine $\boldsymbol{\lambda} = (\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, .., \boldsymbol{\lambda_m})$. Let $\mu_s^u = \mu_s - \sum_{k=1,k\neq u}^{m} \lambda_{sk}$ be the available processing rate at a data center s as perceived by front-end proxy u. Hence, the problem of determining the best reply strategy reduces to finding the optimal job distribution for a system with one front-end proxy u ($\forall u \in U$), n GDCs with rates μ_s^u, ($\forall s \in S$). We can now express the above problem in the following optimization model, denoted by *Best-reply*$_u$.

$$\min_{\boldsymbol{\lambda_u}} \Psi_u(\boldsymbol{\lambda}) \tag{6.22}$$

subject to the following constraints.

$$\lambda_{su} \geq 0, \qquad \forall s \in S \tag{6.23}$$

$$\sum_{s=1}^{n} \lambda_{su} = L_u \tag{6.24}$$

$$\sum_{u=1}^{m} \lambda_{su} < \mu_s, \qquad \forall s \in S \tag{6.25}$$

The decision variables involved in this optimization problem are $\boldsymbol{\lambda_u} = (\lambda_{1u}, \lambda_{2u}, .., \lambda_{nu})$, as the strategies of other players are assumed to be fixed. As our optimization framework has the objective of minimizing the CIC, we sort the data centers in ascending order of C_s, where C_s is defined as

$$C_s = \theta_s + \beta d_{su} \tag{6.26}$$

The following theorem defines the best reply strategy of player u *i.e.*, the solution to the *Best-reply*$_u$.

Theorem 6.1. *Assuming that data centers are sorted based on C_s, the solution $\boldsymbol{\lambda_u}$ for Best-reply$_u$ is given by*

$$\lambda_{su} = \begin{cases} \mu_s^u - \sqrt{\frac{\beta \mu_s^u}{\alpha - C_s}} & \text{if } 1 \leq s < q_u \\ 0 \quad \text{if} & q_u \leq s \leq n \end{cases} \tag{6.27}$$

where q_u is the smallest integer satisfying

$$\sum_{i=1}^{q_u} \sqrt{\frac{\beta \mu_s^u}{\alpha - C_i}} \leq \sum_{i=1}^{q_u} \mu_i^u - L_u \tag{6.28}$$

The above constraint ensures that the demand is served from a cheaper data center first and the expensive one is used only when all the cheaper data centers are full. α is a Lagrangian multiplier whose value is given by

$$\alpha = \theta_{q_u} + \beta(\frac{1}{\mu_{q_u}^u} + d_{su}) \tag{6.29}$$

Proof.

Feasibility: Of the three constraints of the optimization framework, we observe that stability (Eq. 6.25) is always satisfied by the Nash equilibrium solution because of Eq. (6.21) and the fact that total compute capacity of data center is greater than the cumulative client demand. Hence, we need to consider, Eq. (6.23) and Eq. (6.24) as the constraints for our optimization framework.

We first prove that $\Psi_u(\boldsymbol{\lambda})$ is a convex function in $\boldsymbol{\lambda_u}$ and that the feasible solution set formed by Eq. (6.24) and Eq. (6.23) is convex.

It can be easily shown from Eq. (6.15) that $\frac{\partial \Psi_u(\boldsymbol{\lambda})}{\partial \lambda_{su}} \geqslant 0$ and $\frac{\partial^2 \Psi_u(\boldsymbol{\lambda})}{\partial (\lambda_{su})^2} \geqslant 0$ for $s = 1, 2, .., n$. Hence the Hessian of $\Psi_u(\boldsymbol{\lambda})$ is positive implying that $\Psi_u(\boldsymbol{\lambda})$ is a convex function of $\boldsymbol{\lambda_u}$. All the constraints are linear and hence they define a convex polyhedron.

Hence *Best-reply*$_u$ is an optimization problem with a goal of minimizing a convex function over a convex feasible region. The first order KKT conditions are necessary and sufficient conditions for optimality.

Let $\alpha \geqslant 0, \kappa_s \geqslant 0, s = 1, 2, .., n$ denote the Lagrangian multipliers. The Lagrangian is given by

$$L(\lambda_{1u}, \lambda_{2u}, .., \lambda_{nu}, \alpha, \kappa_1, \kappa_2, .., \kappa_n)$$
$$= \sum_{s=1}^{n} \Big(\theta_s \lambda_{su} + \beta \lambda_{su} (\frac{1}{\mu_s^u - \lambda_{su}} + d_{su})\Big) - \alpha\Big(\sum_{s=1}^{n} \lambda_{su} - L_u\Big) - \sum_{s=1}^{n} \kappa_s \lambda_{su} \quad (6.30)$$

The KKT conditions imply that $\lambda_{su}, s = 1, 2, ..n$ is the optimal solution to *Best-reply*$_u$ if and only if there exists $\alpha \geqslant 0, \kappa_s \geqslant 0, s = 1, 2, ..n$ such that,

$$\frac{\partial L}{\partial \lambda_{su}} = 0, \quad (6.31)$$

$$\frac{\partial L}{\partial \alpha} = 0, \quad (6.32)$$

$$\kappa_s \lambda_{su} = 0, \ \kappa_s \geqslant 0, \ \lambda_{su} \geqslant 0, \ s = 1, 2, .., n \quad (6.33)$$

Solving Eq.(6.31),Eq.(6.32) and Eq.(6.33), we get

$$\theta_s + \beta\Big(\frac{\mu_s^u}{(\mu_s^u - \lambda_{su})^2} + d_{su}\Big) - \alpha - \kappa_s = 0 \tag{6.34}$$

$$\sum_{s=1}^{n} \lambda_{su} = L_u \tag{6.35}$$

$$\kappa_s \lambda_{su} = 0,\ \kappa_s \geqslant 0,\ \lambda_{su} \geqslant 0,\ s = 1, 2, .., n \tag{6.36}$$

These are equivalent to

$$\alpha = \theta_s + \beta\Big(\frac{\mu_s^u}{(\mu_s^u - \lambda_{su})^2} + d_{su}\Big),\ \text{if } \lambda_{su} > 0,\quad 1 \leqslant s \leqslant n \tag{6.37}$$

$$\alpha \leqslant \theta_s + \beta\Big(\frac{\mu_s^u}{(\mu_s^u - \lambda_{su})^2} + d_{su}\Big),\ \text{if } \lambda_{su} = 0,\quad 1 \leqslant s \leqslant n \tag{6.38}$$

$$\sum_{s=1}^{n} \lambda_{su} = L_u,\quad \lambda_{su} \geqslant 0,\quad 1 \leqslant s \leqslant n \tag{6.39}$$

From Eq.(6.37), we get the value of λ_{su} as

$$\lambda_{su} = \mu_s^u - \sqrt{\frac{\beta\mu_s^u}{\alpha - \theta_s - \beta d_{su}}}\ \text{ if } \lambda_{su} > 0,\quad 1 \leqslant s \leqslant n \tag{6.40}$$

Claim.*Since our objective is to minimize the CIC, we sort the data centers based on the cost factor ($C_s = \theta_s + \beta d_{su}$), i.e., $C_1 \leqslant C_2 \leqslant .. \leqslant C_n$. Under the given assumption on ordering of data centers, we have the following order on load fraction:$\lambda_{1u} \leqslant \lambda_{2u} \leqslant .., \leqslant \lambda_{nu}$. This implies that there may be a case in which no load has been assigned to a costlier data center while there it can be handled with a cheaper one. This mean that there exist an index $q_u,\ 1 \leqslant q_u \leqslant n$ such that*

$$\lambda_{su} > 0,\ s = 1, .., q_u - 1 \tag{6.41}$$

$$\lambda_{su} = 0,\ s = q_u, .., n \tag{6.42}$$

The Lagrangian multiplier α has to be chosen suitably to satisfy the conservation constraint Eq (6.24). Using Eq. (6.40) and Eq. (6.24), we get

$$\sum_{i=1}^{q_u-1} \sqrt{\frac{\beta\mu_s^u}{\alpha - \theta_s - \beta d_{su}}} = \sum_{i=1}^{q_u-1} \mu_i^u - L_u \tag{6.43}$$

Using Eq. (6.26) the above equation becomes

$$\sum_{i=1}^{q_u-1} \sqrt{\frac{\beta\mu_s^u}{\alpha - C_i}} = \sum_{i=1}^{q_u-1} \mu_i^u - L_u \tag{6.44}$$

where q_u is the minimum index which satisfies

$$\sum_{i=1}^{q_u} \sqrt{\frac{\beta\mu_s^u}{\alpha - C_i}} \leqslant \sum_{i=1}^{q_u} \mu_i^u - L_u \tag{6.45}$$

From the above discussion, we know that one or more λ_{su} is positive, due to Eq. (6.23). Hence at the optimal q_u, we have $\lambda_{q_u u} = 0$. Using this in Eq. (6.40) gives

$$\alpha = \theta_{q_u} + \beta(\frac{1}{\mu_{q_u}^u} + d_{su}) \tag{6.46}$$

Based on the above theorem, we formulate Algorithm 1 for determining the best reply for a proxy u. Algorithm 1 provides the steps to find *Best-reply*$_u$. In Step 1, we sort the data centers based on the cost factor Eq. (6.26). Our aim is to assign greater load to cheaper data centers. In Steps 2 and 3, we initialize the value of p and t according to Eq. (6.28). The sum has been taken over all the data centers. In order to determine the first data center that satisfies Eq. (6.28), *while loop* in Step 4 is used. In particular, we assign load to n^{th} data center as 0, if the $(n-1)^{th}$data center processing rate is capable of satisfying Eq. (6.28). This loop continues till control condition $(p > t)$ is met and the corresponding λ_{nu} is set to 0, and n and p are updated accordingly. Finally in Step 5, we assign load to data centers according to Eq. (6.27).

Algorithm 1: *Best-reply*

Input: Total job arrival rate: L_u

Available processing rate at data centers: $\mu_1^u, \mu_2^u, .., \mu_n^u$

CIC of data centers: $C_1, C_2, ..C_n$

Output: Load balancing strategy:$\boldsymbol{\lambda_u} = (\lambda_{1u}, \lambda_{2u}, .., \lambda_{nu})$

1 Sort the data centers in the ascending order of CIC *i.e.*,
$(C_1 \leq C_2 \leq .. \leq C_n)$

2 $p \leftarrow \sum_{i=1}^{n} \mu_i^u - L_u$

3 $t \leftarrow \sum_{i=1}^{n} \sqrt{\frac{\beta \mu_i^u}{\alpha - C_i}}$

4 **while** $p > t$ **do**
- $\lambda_{nu} \leftarrow 0$
- $p \leftarrow p - \mu_n^u$
- $n \leftarrow n - 1$
- $t \leftarrow \sum_{i=1}^{n} \sqrt{\frac{\beta \mu_i^u}{\alpha - C_i}}$

5 **for** $i = 1, 2, .., n$ **do**
- $\lambda_{iu} \leftarrow \mu_s^u - \sqrt{\frac{\beta \mu_s^u}{\alpha - C_s}}$

Theorem 6.2. *The load balancing strategy* $\boldsymbol{\lambda = (\lambda_1, \lambda_2, ..., \lambda_m)}$*, given by the Best-reply algorithm is the best strategy for front-end proxy u and it solves the Best-reply*$_u$ *problem.*

Proof. *The while loop in step 4 finds the minimum index* q_u *for which*

$$\sum_{i=1}^{q_u} \sqrt{\frac{\beta \mu_s^u}{\alpha - C_i}} \leqslant \sum_{i=1}^{q_u} \mu_i^u - L_u \tag{6.47}$$

In the same loop, λ_{iu} *are set to zero for* $i = q_u, ..., n$*. In step 5,* λ_{iu} *is set equal to* $\mu_s^u - \sqrt{\frac{\beta \mu_s^u}{\alpha - C_s}}$ *for* $i = 1, ..., q_u - 1$*. These are in accordance to Theorem 1. Thus, the allocation* $\boldsymbol{\lambda_u} = (\lambda_{1u}, \lambda_{2u}, .., \lambda_{nu})$ *computed by Best-reply algorithm is the optimal solution of Best-reply*$_u$*.*

Remarks: (i) The execution time of this algorithm is $\mathcal{O}(nlgn)$. This is due to the sorting procedure in Step 1. (ii) In order to execute this algorithm each front-end proxy needs to know all the necessary information such as electricity price, expected delay and available processing capacity at each data center location and estimated demand [89]. In most of the cases, all the proxies and data centers are owned by the same operator. It is possible that all data centers and proxy servers exchange the estimated load, capacity, and electricity prices in real-time.

6.4 A Distributed Load Balancing Algorithm

Based on the *Best-reply* algorithm discussed in the previous section, we design a greedy algorithm for computing the Nash equilibrium of the non-cooperative game. The proposed algorithm is given in Algorithm 2. In addition to the notation mentioned in Sections 6.2 and 6.3, let l denote the iteration index and u denote the front-end proxy index. Let $\boldsymbol{\lambda_u^l}$ denote the strategy of front-end proxy u at iteration l. Let Ψ_u^l be the CIC of client region u at iteration l, ε be the selected acceptance tolerance. We define *norm* as $\sum_{u=1}^{m} |\Psi_u^{(l-1)} - \Psi_u^l|$. The function $\mathbf{Send}(u,(p,q,r))$ denotes sending message (p,q,r) to front-end proxy u, where p is the current sum of $|\Psi_u^{(l-1)} - \Psi_u^l|$, q is the iteration number, and r is $CONTINUE/STOP$ tag. Similarly, $\mathbf{Recv}(u,(p,q,r))$ denotes receiving message (p,q,r) from the front-end proxy u.

In this algorithm, each front-end proxy computes its *Best-reply* strategy for every time slot using the current load balancing strategies of other front-end proxies and updates its strategy. In an iteration l of while loop, each front-end proxy server computes its *Best-reply* strategy. It then adds to the sum, the difference in its achieved operating cost compared to previous $(l-1)$ iteration. Then, it sends the sum to its neighbour in a round robin fashion. This continues for several iterations and finally stops when the difference in the total operating cost across all the front-end proxy servers, in successive iterations, is less than the *norm* stopping criterion. We

Algorithm 2: Nash distributed load balancing algorithm

Input: Total Arrival rate: L_u
Available processing rate at data centers: $\mu_1^u, \mu_2^u, \ldots, \mu_n^u$
Cost factors of data center: $C_1, C_2, \ldots, C_n$
Output: Load balancing strategy at equilibrium:$\boldsymbol{\lambda} = (\lambda_u, \forall u)$
Front-end proxy u, $u = 1, 2, \ldots, m$ executes:

1 Initialization:
  $\boldsymbol{\lambda_u^l} \leftarrow \mathbf{0}$
  $\Psi_u^0 \leftarrow 0$
  $l \leftarrow 0$
  $norm \leftarrow 1$
  $sum \leftarrow 0$
  $tag \leftarrow CONTINUE$
  $left = [(u-2) \bmod m] + 1$
  $right = [u \bmod m] + 1$
2 **while** (1) **do**
  if *(*$u = 1$*)* **then**
    if $l \neq 0$ **then**
      Recv$(left, (norm, l, tag))$
      if $norm < \varepsilon$ **then**
        Send$(right, (norm, l, STOP))$
        exit
    $sum \leftarrow 0$
    $l \leftarrow l + 1$
  else
    Recv$(left, (sum, l, tag))$
    if $tag = STOP$ **then**
      if $u \neq m$ **then**
        Send$(right, (sum, l, STOP))$
      exit
    for $s = 1, 2, \ldots, n$ **do**
      $\mu_s^u \leftarrow \mu_s - \sum_{k=1, k\neq u}^{m} \lambda_{sk}$
      $\boldsymbol{\lambda_u} \leftarrow$ *Best-reply*$(\mu_1^u, \ldots, \mu_n^u, C_1, \ldots, C_n, L_u)$
      Compute Ψ_u
      $sum \leftarrow sum + |\Psi_u^{(l-1)} - \Psi_u^l|$
      Send$(right, (sum, l, CONTINUE))$

assume that the front-end proxies synchronously update their *Best-reply* strategies in a round robin manner.

The execution of this algorithm is restarted periodically when the data center system parameters (*e.g.*, electricity price, demand) change or a failure occurs. Once the equilibrium is reached, the front-end proxies continue to use the same strategy and the system remains in equilibrium until a new execution is initiated.

Discussion: We examine the message complexity of Algorithm 2. We assume that all the front-end proxy servers (collocated client regions) are logically connected in a ring topology. In every iteration, each front-end proxy server gathers the available capacity at each data center, which requires $\mathcal{O}(n)$ messages. Each front-end proxy server also needs to share its updated sum with the adjacent node, which requires $\mathcal{O}(1)$ messages. Thus, in each iteration all the front-end proxy servers require $\mathcal{O}(mn+m)$ messages, where n and m are the number of data centers and front-end proxy servers, respectively. The asymptotic message complexity is $\mathcal{O}(mn)$.

6.5 Numerical Results

In this section, we compare and analyze the performance of the proposed algorithm with different existing strategies. The proposed algorithm is abbreviated as NCG. In addition, we also implement for comparison two other models in literature: Proportional scheme (PS) and Global optimal scheme (GOS).

- Proportional scheme(PS): It is distributed and decentralized approach, where each front-end proxy allocates load in proportion to the processing rate at each data center. It can be noted that proportional scheme does not take into account either the operating cost or the communication delay between front-end proxy and data center.

- Global optimal Scheme(GOS): This scheme minimizes the CIC defined in

Section 6.3. The load values (λ) are obtained by solving the following non-linear optimization problem:

$$\min_{\boldsymbol{\lambda}} \sum_{u} \Psi_u(\boldsymbol{\lambda}) \tag{6.48}$$

subject to the following constraints.

$$\lambda_{su} \geq 0, \qquad \forall s \in S, \forall u \in U \tag{6.49}$$

$$\sum_{s=1}^{n} \lambda_{su} = L_u \qquad \forall u \in U \tag{6.50}$$

$$\sum_{u=1}^{m} \lambda_{su} < \mu_s, \qquad \forall s \in S \tag{6.51}$$

GOS is evaluated (at a centralized location) using the optimization tool *fmincon* of MATLAB. The parameters used with *fmincon* are presented in Table 6.1. We use default values for other parameters such as choice of optimization algorithm, optimality tolerance, step tolerance, and checkgradient [90]. GOS provides optimal solution, but does not provide fairness to users.

Parameter	Value
MaxFunctionEvaluations	10000
MaxIterations	5000
FunctionTolerance	10^{-6}
ConstraintTolerance	10^{-6}
OptimalityTolerance	10^{-6}
StepTolerance	10^{-10}
Other parameters	Default values

Table 6.1: fmincon parameters

The main performance metrics used in our numerical results are normalized cost, expected response time and the fairness index. The fairness index $F(\boldsymbol{x})$ is calculated as

$$F(\boldsymbol{x}) = \frac{[\sum_{u=1}^{m} x_u]^2}{m \sum_{u=1}^{m} x_u^2} \tag{6.52}$$

where x_u is the expected latency at client region u. The sum of expected cost across for all client regions C is calculated as

$$C(\boldsymbol{\lambda}) = \sum_{u=1}^{m} \Psi_u(\boldsymbol{\lambda}) \tag{6.53}$$

where λ is the strategy at the equilibrium. The normalized cost is obtained by normalization with respect to the maximum cost across all approaches.

6.5.1 System Setup

Data center locations: We considered data center locations in the USA based on power availability as mentioned in [53]. The locations are: Arizona, Illinois, Iowa, Mississippi, New Hampshire, Oklahoma, Oregon, Pennsylvania, South Carolina, and Utah. The service rate of server is 60 requests per second. Relative processing rate available across all the data centers is given in Table 6.2, where relative processing rate of a data center is calculated with respect to the fastest data center processing rate. The average electricity price across all the data center locations is given in Table 6.2 [21].

Client locations: We considered 12 states of the USA, where a large number of Internet users are located [67], *viz*, California, Florida, Georgia, Illinois, Michigan, New Jersey, New York, North Carolina, Ohio, Pennsylvania, Texas and Virginia. Demand from different client locations is kept proportional to the number of Internet users from that region [67].

DC Locations	Arizona	Illinois	Iowa	Mississippi	New Hampshire	Oklahoma	Oregon	Pennsylvania	South Carolina	Utah
Electricity Price (in cents/kWh)	5.54	6.34	5.20	6.03	12.33	4.62	6.38	6.78	5.55	5.58
Relative processing rate	0.4	1.0	0.2	0.6	0.2	0.6	0.4	1.0	0.4	1.0

Table 6.2: Average electricity price and processing rate across data center location

Client Locations	California	Florida	Georgia	Illinois	Michigan	New Jersey
Relative job arrival rate	0.17	0.11	0.03	0.11	0.04	0.04
Client Locations	New York	North Carolina	Ohio	Pennsylvania	Texas	Virginia
Relative job arrival rate	0.12	0.03	0.06	0.1	0.13	0.02

Table 6.3: Relative job arrival rate of each client location

Demand: We used trace of traffic from Wiki dump [68] to build the workload profile. Considering hourly wikipedia workload as an aggregate demand for every hour, we distribute demand across different client locations proportional to the number of Internet users at each location. The peak demand from the trace is of the order of thousand requests per second whereas, literature suggests that data centers serve requests to the tune of million requests per second [4]. Therefore we have upscaled the demand by a factor of 3000. The relative job arrival rate for each client location is given in Table 6.3. The propagation delay between data center and client location is considered proportional to the distance between them, and it increases by 10 ms for every 1000 km [16]. The energy consumption of an idle server at any given time could be 30% of the peak power [87, 91], P_{idle} and P_{peak} are taken to be 300W and 100W, respectively [92].

6.5.2 Results

We present and discuss the results obtained after evaluating NCG, PS and GOS models by varying the number of data centers, system workload and factor term

β (see Eq. 6.12). For each case, we plot the normalized value of cost wherein, the normalization is done with respect to maximum cost across all experiments. We evaluate the models for a period of one day with the system state changing every hour. The results presented here are average values obtained over this time horizon.

Effect of System size

We vary the number of data centers in the system from 6 to 10 and investigate its effect on the cost and fairness. We set demand as mentioned in the earlier section and β as 0.1. Fig. 6.2 shows the normalized cost for all the models. It can be observed that the cost obtained using NCG and GOS is almost the same because these models are aware of electricity price, propagation delay and data center processing rates and therefore perform load balancing in a cost-effective manner whereas, PS does not obtain optimum cost as it uniformly distributes the load without taking into consideration the electricity prices, propagation delays and data center processing rates. It can also be observed from Fig. 6.3 that NCG model achieves better fairness in average latency across all the client regions when compared to GOS whereas, the fairness of PS is approximately 1 across different number of data centers. The NCG approach has an additional advantage of being a decentralized load balancing scheme.

6.5.3 Impact of Demand

In order to understand the impact of demand on cost, we evaluate the proposed algorithm and other models on 10 data centers and 12 client regions. We vary the client demand from 0.2 to 2 times the original client demand obtained from [68]. It may be noted that demand is chosen such that it covers a broad range of data center utilization, where utilization is defined as the ratio of total arrival rate to aggregate processing rate of the system. Demand of 1 corresponds to a utilization of 44%. We

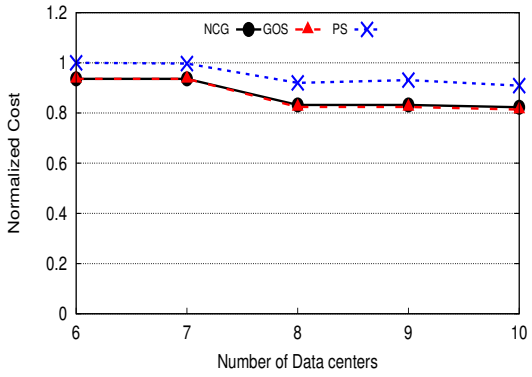


Figure 6.2: Impact of the number of data centers on the cost

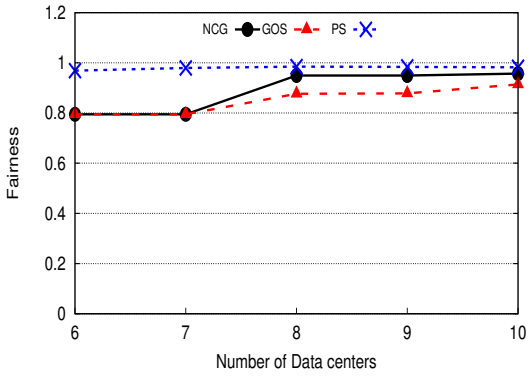


Figure 6.3: Impact of the number of data centers on the latency fairness

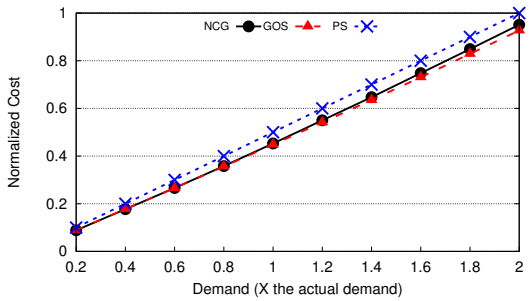


Figure 6.4: Impact of demand on the cost

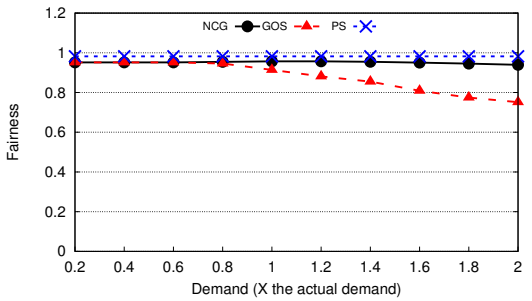


Figure 6.5: Impact of demand on the latency fairness

set β to 0.1. We show the results obtained in Fig. 6.4. It can be seen that with increase in demand NCG yields almost the same cost as the GOS approach, which means that NCG approach is as effective as GOS. From Fig. 6.5 it can be seen that NCG and PS maintain a fairness close to 1. Therefore the NCG scheme, apart from being decentralized, has the additional advantage of user optimality with respect to fairness in latency.

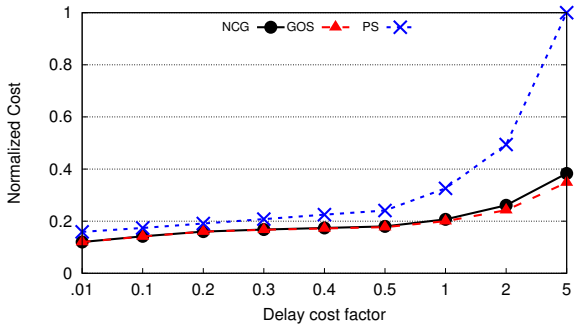


Figure 6.6: Impact of cost incurred due to delay on the overall cost

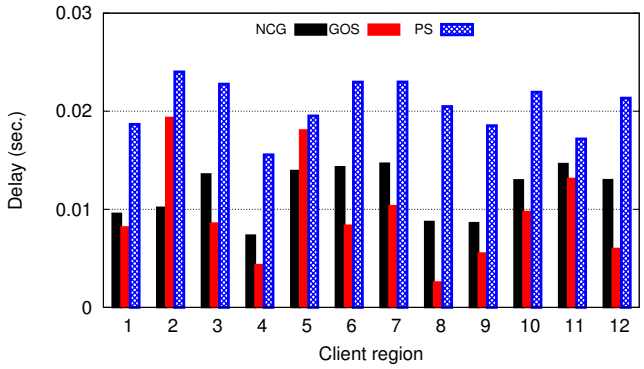


Figure 6.7: Average latency perceived across various regions

6.5.4 Impact of β

In order to understand the impact of β (delay cost factor) on cost and fairness index, we evaluate the proposed and existing schemes with 10 data centers and 12 client regions, and $\beta \in \{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 1, 2, 5\}$. It can be observed from Fig. 6.6 that across all β values our model approximates the cost of GOS. With increase in β the cost also increases for all models which is quite intuitive as increase in weight of latency factor yields a higher cost.

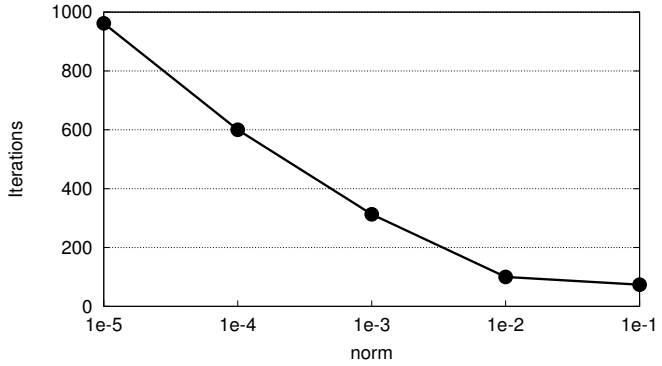


Figure 6.8: Impact of norm on convergence

6.5.5 Client Latency

In Fig. 6.7, we present the expected latency experienced by each client region. Though the PS scheme guarantees fairness in latency across all client regions, it has a higher expected latency. It can also be observed that with GOS scheme, there is a huge variation in the expected latency across all client regions, while NCG gives the lowest expected latency for each client region. In many cases, it can be observed that GOS is better and gives optimal solution that minimizes both the operating cost and the delay.

6.5.6 Convergence of NCG Algorithm

In order to determine the number of iterations required for NCG to converge, we consider two scenarios. First we determine the number of iterations required for norm $\in \{1e-5, 1e-4, 1e-3, 1e-2, 1e-1\}$, with 10 data centers and 12 clients regions. We set the demand as mentioned in Section 6.5.1 and β as 0.1. Fig. 6.8 shows the number of iterations required for the converge. From this, we see that the proposed algorithm converges after a reasonable number of iterations for the input size considered.

6.6 Summary

In this chapter, we formulated the load balancing problem in GDC as a non-cooperative game among front-end proxy servers (which are collocated with client regions). For the proposed game we characterized the structure of Nash equilibrium. Based on the Nash equilibrium structure, we derived a distributed load balancing algorithm for computing the same. We compared the performance of our non-cooperative game (NCG) with the existing proportional sharing and global optimal approaches. The main advantages of NCG algorithm is that it is decentralized, it has a low complexity (yet close to the optimal GOS), and it offers fairness in average latency across all the client regions.

In this chapter we assume that there is an efficient failure detection and recovery mechanism. The distributed load balancing algorithm is executed every time state of the system changes. Since the data center can fail at any time, probing to detect failures and thereby computing the fail-over load balancing policy is computationally expensive. In the next chapter, we propose a data center-initiated load balancing approach, which is efficient and quickly converges to the optimal load balancing policy.

Chapter 7

Distributed Failure Detection and Efficient Load Balancing in Fault-tolerant GDCs

7.1 Introduction

In the previous chapter, we addressed the problem of distributed load balancing in fault-tolerant GDCs to minimize the operating cost while ensuring the fairness in latency perceived across the clients. Once we have a model to provision spare capacity across the data centers to handle failures without increasing the TCO, we need a cost-aware load balancing strategy. After the failure, the front-end proxies need to locate the failure quickly and a new load balancing policy should be worked out that re-routes the requests to the available data centers. Keeping our philosophy of designing cost-effective fault-tolerant GDCs, the updated load balancing policy should also minimize the operating cost.

The problem of load balancing becomes challenging when it has to consider the failure of a data center (either partial or complete), since failure can happen at

anytime and with any frequency [6]. In fault-tolerant GDCs, failure detection could be carried out by central controller [15] or in a distributed manner by front-end proxy servers [33]. In the distributed approach, front-end proxies send keep-alive messages periodically for health monitoring. This could be computationally expensive in terms of message and time complexity. A cost-aware load balancing strategy working in the presence of failures, has to select a new data center for request re-routing considering renewable energy usage requirement, electricity cost, and QoS requirements. For a scalable load balancing system, we propose a data center-initiated, distributed load balancing strategy to satisfy post-failure QoS requirements while minimizing the operating cost. Our approach makes an early attempt to shift the load on a failed data center to mask the failure with a marginal increase in the operating cost. We model the problem of load balancing in fault-tolerant data centers using linear programming (LP) to optimize both the cost of energy consumption and to minimize the client latency even after failure. We propose a two-stage distributed scalable algorithm based on greedy method to solve the problem.

First, at a failed data center, we determine the surplus load and use *Shift Workload* algorithm to distribute the load across the remaining data centers, while minimising the operating cost (considering the energy consumption cost). This algorithm considers the queuing delay due to additional load on the remaining data centers (due to a failed data center). Second, we use *Request Re-routing* algorithm based on minimum cost network flow model, at the front-end proxy to minimize the latency in serving the client requests. We evaluate the proposed algorithm using real-world data to show its closeness to the optimal strategy, which makes it suitable for cost-aware online load balancing.

The rest of this chapter is organized as follows. The proposed system model and the formulation of the optimization problem are presented in Section 7.2. Section 7.3 discusses the two-stage distributed algorithm for load balancing. We also prove the

optimality and discuss the complexity of the algorithm. Few important results from a numerical evaluation of the proposed algorithm are presented in Section 7.4. Section 7.5 concludes the chapter.

7.2 Problem Formulation

We cast the problem of load balancing in fault-tolerant GDCs as two sub-problems, *workload shifting* and *request re-routing* problems. We use LP to model these problems. The solution to the *workload shifting* problem determines the amount of workload assigned (re-assigned) to each data center after the failure. Solving the *request re-routing* problem minimizes the latency for the client requests considering the increased workload on the remaining data centers.

Fig. 7.1 shows a schematic representation of the GDC architecture considered in this model. Each data center is powered by multiple renewable energy sources. S denotes the set of data center locations each housing m_s number of servers ($s \in S$). A set of front-end proxy servers U is associated with each client region generating a load L_u ($u \in U$). Each data center houses a resource manager, responsible to detect failures and to run the algorithms for workload shifting and request re-routing after the failure. The outputs from the algorithm are λ_s^{uf}: the number of requests from client region u served by data center s when f^{th} data center has failed, and m_s^f: the number of active servers at data center s when f^{th} data center has failed. The load balancing policy governed by λ_s^{uf} is pushed to the front-end proxy servers (as indicated by the dotted lines in Fig. 7.1) and the number of servers required to handle the load of the failed data center is updated due to m_s^f (as indicated by green lines in Fig. 7.1).

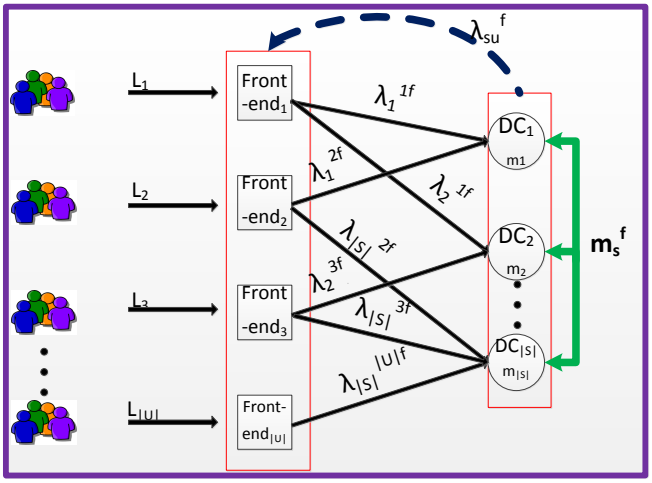


Figure 7.1: The system architecture used for load balancing

7.2.1 System Model

We first describe the assumptions used in the formulation and define various components used in the formulation. Table 7.1 summarises the notation used in this chapter.

Assumptions: The data centers are over-provisioned with sufficient compute capacity across various locations to accommodate single data center failure. Every data center is aware of the volume of workload being served across all the other data centers.

We now define the input parameters used in the formulation.

Demand: We define λ_s^{uf} as the number of requests mapped from client region u to data center s. Here $f = 0$ represents no data center failed and $f \in 1, 2..|S|$ represents the failure of the f^{th} data center.

Number of active servers: Let the processing rate of the server be μ requests per second. We assume that among m_s servers at a data center, m_s^f are activated after f^{th} data center has failed. A data center is modeled as an M/M/n queue. The number of servers required to satisfy workload l within the maximum tolerable

Input	Symbol	Description
Data center	S	Set of data center locations
	s	Index of data center
	M_s	Total number of servers available at data center s
	m_s^f	Number of active servers at data center s when data center f has failed
	p	Percentage of total servers failed at any data center
Client	U	Set of client regions
	u	Index of client region
	L_u	Total number of requests generated from user location u
Server	P_{idle}	Power consumed by server when idle
	P_{peak}	Power consumed by server when running at its peak
	μ	Service rate of server (in number of requests per second)
	γ_s^f	Average utilization of active servers at data center s when data center f has failed
Demand and workload distribution	L'	Total workload served by failed data center before failure
	L	Total workload required to be distributed
	L_g	Workload required to be served by green energy sources
	L_r	Workload which can be served by any energy source
	λ_s^{uf}	The number of requests from client region u served by data center s when data center f has failed
	$\boldsymbol{\lambda^f}$	Vector formed using λ_s^{uf}, which denotes the load balancing strategy
	Φ_{si}^{uf}	The number of requests from client region u being served at data center s being served by consuming energy of type i when data center f has failed
	Φ_{si}^f	The cumulative workload at data center s by consuming energy of type i when data center f has failed
	$\boldsymbol{\Phi_s^f}$	Vector of Φ_{si}^f for data center s when data center f has failed.
	$\boldsymbol{\Phi^f}$	Vector of $\boldsymbol{\Phi_s^f}$, which signifies the total workload distribution among different data centers and energy sources
Energy	i	Index of energy type
	θ_{si}	Electricity price per kWh at data center s for energy of type i
	$\boldsymbol{\theta}$	Vector of different electricity prices across all data centers
	P_{si}^f	Total power of type i consumed at data center s when data center f has failed
	Θ_s^f	The total cost of energy consumed at data center s when data center f has failed
	ρ	The minimum green energy usage bound
	E_s	PUE at data center s
	P_{si}^{max}	Maximum available power of type i at data center s
Delay	W_{su}	Propagation delay between data center s and client region u
	T	Maximum tolerable queuing delay

Table 7.1: Summary of notation used in the chapter

queuing delay T is given by the equation

$$\frac{1}{m_s^f \mu - l} = T \qquad \forall s, f \tag{7.1}$$

$$l = f(m_s^f) = m_s^f \mu - \frac{1}{T} \qquad \forall s, f \tag{7.2}$$

In Eq. (7.2), the function $f(m_s^f)$ gives the workload that can be served using m_s^f servers while meeting the queuing delay constraint. Similarly, we define m_s^f by

$$m_s^f = \frac{1}{\mu}(\frac{1}{T} + l) \qquad \forall s, f \tag{7.3}$$

Server utilization: We define γ_s^f to be the average utilization of active servers at the s^{th} data center after f^{th} data center failed, given by

$$\gamma_s^f = \frac{\sum_u \lambda_s^{uf}}{\mu m_s^f} \tag{7.4}$$

Power consumption model: Let P_{idle} be the average power drawn in idle condition and P_{peak} be the power consumed when server is running at peak utilization. We express the power consumed at a data center s, given workload l, after f^{th} data center failed, as [16]:

$$P_s^f = m_s^f(P_{idle} + (E_s - 1)P_{peak}) + m_s^f(P_{peak} - P_{idle})\gamma_s^f \tag{7.5}$$

$$= c_1 m_s^f + c_2 l \tag{7.6}$$

where $c_1 = (P_{idle} + (E_s - 1)P_{peak}), c_2 = \frac{(P_{peak} - P_{idle})}{\mu}$, and E_s is the PUE of a data center at s. Substituting m_s^f and γ_s^f in Eq. (7.6) from Eq. (7.3) and Eq. (7.4) gives

$$P_s^f = \frac{c_1}{\mu}(\frac{1}{T} + l) + c_2 l$$

$$= (\frac{c_1}{\mu} + c_2)l + \frac{c_1}{T\mu} \tag{7.7}$$

Surplus workload: Let L' be the workload served by a failed data center f, before the failure. After the failure, let m_f^f be the number of servers available. The surplus

workload violating the queuing delay constraint at failed data center can be obtained as

$$L = L' - f(m_f^f) \qquad \forall f \tag{7.8}$$

Workload partitioning: We partition the workload into two components: (i) *green workload*, L_g which is served by servers powered with green energy and (ii) remaining workload, L_r served by other servers. Due to workload conservation rule, the following equation has to be satisfied.

$$L_g + L_r = L \tag{7.9}$$

Let ρ denote the minimum fraction of workload to be served by servers powered with green energy. The green energy usage constraint is represented by

$$L_g \geq \rho L \tag{7.10}$$

In our model, we consider 4 types of energy sources namely, solar, wind, PPA and brown (indexed in the same order). Φ_{si}^{uf} denotes the workload from client region u at data center s being served by consuming energy of type i, after f^{th} data center had failed.

The following equation makes sure that all the client demand is served.

$$\sum_{i=1}^{4} \Phi_{si}^{uf} = \lambda_s^{uf} \qquad \forall s, u \tag{7.11}$$

Φ_{si}^{f} denotes the cumulative workload at data center s being served by consuming energy of type i (*i.e.*, $\Phi_{si}^{f} = \sum_u \Phi_{si}^{uf}$). Therefore, in order to satisfy the green energy usage we have

$$\sum_{s} \sum_{i=1}^{3} \Phi_{si}^{f} \geq L_g \qquad \forall s \tag{7.12}$$

Energy cost: Let θ_{si} be the cost of energy of type i consumed at a data center s and $\boldsymbol{\theta}$ be the vector of different energy prices across all data centers. The cost of

energy consumed at data center s after f^{th} data center has failed can be expressed as

$$\Theta_s^f = \sum_i \theta_{si} P_{si}^f \qquad \forall s, f \tag{7.13}$$

where P_{si}^f is the energy of type i is consumed at data center s when data center f has failed.

7.2.2 Optimization Model

As mentioned earlier, the load balancing problem is formulated as two sub-problems, both of which are modeled using LP. We use the cost factors discussed previously along with the necessary constraints in the formulation for the two sub-problems.

Workload shifting

In this problem, the load on a failed data center is shifted to alternate data centers such that the total operating cost is minimized. The input parameters are: L, the aggregate demand and ρ, the minimum green energy usage requirement. The decision variable is Φ_{si}^f, the number of requests that data center s will serve consuming energy of type i after failure of f^{th} data center. The workload shifting problem (denoted by **P1**) is formally defined as

$$\textbf{P1:} \quad \textbf{minimize} \;\; \Psi = \sum_s \Theta_s^f \tag{7.14}$$

subject to,

$$\sum_{s,i} \Phi_{si}^f = L \tag{7.15}$$

$$\sum_s \sum_{i=1}^{3} \Phi_{si}^f \geq \rho L \tag{7.16}$$

$$\Phi_{si}^f \geq 0 \qquad \forall s, i, f \tag{7.17}$$

Request re-routing

In this stage, the front-end proxy tries to minimize the client latency. This problem is modeled with an objective to minimize the total network latency subject to client demand and data center capacity constraints. The input parameters are: $L_u-$the total number of requests generated from user location u, $W_{su}-$the RTT between data center s and client region u, and $\Phi_s-$the total workload to be served by data center s in presence of any data center failure f (where $\Phi_s = \sum_{i=1}^{3} \Phi_{si}^{f}$). The decision variable is λ_s^{uf}, the number of requests to be served by each data center. The request re-routing problem (denoted by **P2**), is formally defined as

$$\textbf{P2:} \quad \textbf{minimize} \sum_{s,u} W_{su}\lambda_s^{uf} \tag{7.18}$$

subject to,

$$\sum_{u} \lambda_s^{uf} = \Phi_s \qquad \forall s \tag{7.19}$$

$$\sum_{s} \lambda_s^{uf} = L_u \qquad \forall u \tag{7.20}$$

$$\lambda_s^{uf} \geq 0 \qquad \forall s, u \tag{7.21}$$

7.3 Distributed Load Balancing Algorithms

In this section, we present online algorithms to solve the two problems **P1** and **P2** defined previously. For problem **P1**, we propose a heuristic *Shift Workload Algorithm*. For problem **P2**, we show that it can be casted as minimum cost flow problem, which can be solved by any existing strongly polynomial-time algorithms, like the one in [93].

Algorithm 3 gives an overview of the approach taken to solve both **P1** and **P2**. The input parameters for these algorithms include: surplus workload (L),

electricity prices across different data centers (θ), green energy usage bound (ρ), propagation delay matrix (PD) between client regions and data center, and the workload being served by different data centers using various energy options($\mathcal{S}$). Line 1 solves the *Workload Shifting* problem, to determine the distribution of surplus load across remaining data centers with minimal operating cost while satisfying the queuing delay and green energy usage constraints. Line 2 solves the *Request Re-routing* problem to distribute the client requests among the data centers based on propagation delay and the current workload (as determined by the output from Line 1). The updated load balancing strategy is pushed to the front-end proxy servers and the number of servers to be switched on is passed on to the active data centers.

Algorithm 3: *Load balancing in the event of a failure*

Input:

L: Surplus workload that violates QoS at failed data center

$\boldsymbol{\theta} = (\theta_{11}, \theta_{12}, .., \theta_{n4})$: Price of different types of energy sources at data centers.

ρ: Minimum green energy usage bound

PD: Propagation delay matrix

Output:

Load balancing strategy:$\boldsymbol{\lambda^f} = (\lambda_1^{1f}, \lambda_1^{2f}, .., \lambda_n^{mf})$

In the event of a failure during time slot t,

1 $\boldsymbol{\Phi^f} \leftarrow$ ShiftWorkload(L,$\boldsymbol{\theta}$,ρ,$\mathcal{S}$)

2 $\boldsymbol{\lambda^f} \leftarrow$ Request Rerouting

3 Update the new load balancing strategy $\boldsymbol{\lambda^f}$ to all front end proxies and data centers

7.3.1 Shift Workload Algorithm

First, we present an online algorithm to solve **P1** defined in Section 7.2.2 taking into account green energy usage bound, real-time electricity prices, and QoS requirements. Algorithm 4 given below presents the complete algorithm used to solve P1.

Let us define the following functions. $f(M_s)$: the workload that can be served using M_s servers while satisfying the queuing delay constraint (from Eq. (7.2)), $g(P_{si}^f)$: the workload that can be served using the available energy P_{si}^f (from Eq. (7.7)), $h(\Phi_{si}^f)$: the energy required to serve the workload Φ_{si}^f (from Eq. (7.7)), and $q(s, \mathbf{\Phi_s^f})$: the costliest renewable energy source being used at data center s.

First, the percentage of failure p is estimated by the resource manager at f^{th} data center. Next, the workload that needs to be migrated to other data centers is estimated using Eq. (7.2). Finally, the data centers that would handle the load of failed data center are determined based on the cost of power consumption. The excess workload L is divided as L_g and L_r, where L_g is assigned to servers powered by green sources (to meet green energy usage bound). In Algorithm 4, line 3 determines the value of L_r which is the workload assigned to servers powered by brown energy before failure at f. Line 6 determines L_g, using $q(s, \mathbf{\Phi_s^f})$ (uses decreasing order of the cost of green energy sources). Line 7 distributes L_g across different sites powered by green sources, while Line 8 distributes L_r across the data centers considering cheaper electricity price.

7.3.2 Request Re-routing Algorithm

We model the request re-routing problem as a minimum cost network flow problem. Fig. 7.2 depicts the network flow graph with S and T denoting the source and sink respectively. A set of front-end proxy servers, denoted by $\mathrm{F}_i, i = 1, 2, \ldots |U|$, distribute the requests across data centers, denoted by $\mathrm{DC}_j, j = 1, 2, \ldots |S|$. The

Algorithm 4: Shift Workload (L,$f, \boldsymbol{\theta}$,$\rho, \mathcal{S}$)

Input:

L: Surplus workload that violates QoS at failed data center

$\boldsymbol{\theta} = (\theta_{11}, \theta_{12}, .., \theta_{n4})$: Price of different types of energy sources at data centers.

ρ: Minimum green energy usage bound

$\mathcal{S}$: Current system state which includes the volume of workload being serverd by different data centers using various energy options.

Output:

$\boldsymbol{\Phi^f} = (\Phi_{11}, \Phi_{12}, .., \Phi_{n4})$

$\boldsymbol{\nu} \leftarrow \boldsymbol{\theta(1:n, 1:3)}$

$M_f \leftarrow (1-p)M_s$

`/* workload served by brown energy before failure */`

$L_r \leftarrow \min[L, \Phi_{f4}]$

$L \leftarrow L - L_r$

$L_g \leftarrow 0$

`/* determine workload (`L_g`) served by green sources, required to be shifted to mask failure and keep green energy usage requirement intact */`

while $L > 0$ **do**

- $i \leftarrow q(f, \boldsymbol{\Phi_f^f})$
- $\delta \leftarrow \min[L, \Phi_{fi}^f]$
- $L \leftarrow L - \delta$
- $L_g \leftarrow L_g + \delta$
- $\Phi_{fi}^f \leftarrow \Phi_{fi}^f - \delta$

`/* spreads the load` L_g `across data centers with minimum green energy cost */`

while $L_g > 0$ **do**

- $s, i \leftarrow$ arg-min$[\boldsymbol{\nu}]$
- $\delta_{si} \leftarrow \min[L_g, f(M_s) - \sum_i \Phi_{si}\ ,\ g(P_{si}^{max})]$
- $L_g \leftarrow L_g - \delta_{si}$
- $\Phi_{si} \leftarrow \Phi_{si} + \delta_{si}$
- $P_{si}^{max} \leftarrow P_{si}^{max} - h(\delta_{si})$

`/* spreads the remaining load` L_r `across data centers with minimum energy cost */`

while $L_r > 0$ **do**

- $s, i \leftarrow$ arg-min$[\boldsymbol{\theta}]$
- $\delta_{si} \leftarrow \min[L_r, f(M_s) - \sum_i \Phi_{si}\ ,\ g(P_{si}^{max})]$
- $L_r \leftarrow L_r - \delta_{si}$
- $\Phi_{si} \leftarrow \Phi_{si} + \delta_{si}$
- $P_{si}^{max} \leftarrow P_{si}^{max} - h(\delta_{si})$

edge between S and F_i is weighted with capacity L_i, which also corresponds to load at front-end F_i, and a cost of zero. For the sake of simplicity, we assume the capacity of edge between F_i and DC_j to be L_i and cost to be W_{ij}, which is based on network latency between data center j and the front-end proxy i (serving client region i). Similarly, the edge between DC_j and T is assigned a capacity of Φ_j (obtained from Algorithm 4) which is the total workload that can be served at data center j and the cost is zero. This capacitated minimum cost network flow problem can be efficiently solved with an algorithm like the one proposed in [93].

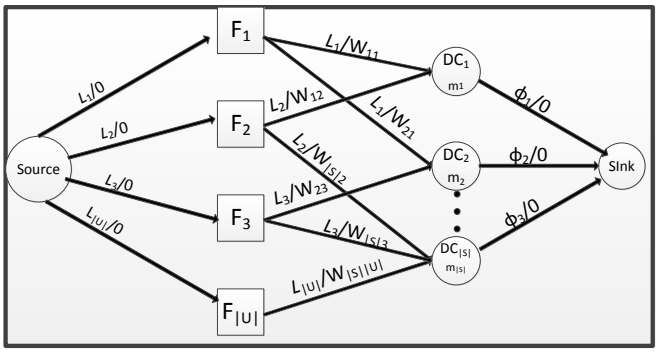


Figure 7.2: Illustration of min-cost network flow model for P2

Next, we prove that the algorithm presented in 7.3.1 will give an optimal solution for P1. Let L denote the surplus workload due to data center failure, of which L_g and L_r denote the demand to be served by green energy sources and brown energy sources, respectively.

Lemma 7.1. *All the demand previously served by a failed data center is allocated to servers in the remaining data centers.*

Proof. Let us consider a request $l \in L$ that could not be served by any data center, which implies that no compute capacity is available in remaining data center to satisfy l within the delay constraint, which could not be the case as spare capacity provisioning is already done. The other reason could be that there is insufficient

green energy to satisfy the green energy bound, which again could not happen as renewable energy installation is already ensured to be sufficient to handle post-failure scenario. This gives a contradiction and hence the statement of the lemma is proved.

Lemma 7.2. *At the end of Algorithm 4, the cost incurred in serving the total workload even after the failure is the lowest.*

Proof. As discussed in the formulation, we split the workload of failed data center into two components L_g and L_r. Our greedy algorithm requires maximum 2 iterations at each data center, one to find cheaper green energy sources and the other to find any other cheaper energy source to serve the workload components L_g and L_r. As a part of the execution, we maintain two sorted lists of data centers, one sorted in increasing order of the cost of green energy source and other one in increasing order of the cost of any available energy source.

For the sake of simplicity, we focus on one iteration that satisfies the workload L_r (henceforth, denoted by L). Let L_i denote the number of requests served so far by the algorithm, where i is the index of data center (from the list sorted in increasing order of energy cost). For any L_i served in an iteration of allocating workload to $DC_j, j = 1, 2, \ldots |S|$, let the cost incurred be C_i. Our argument is that the optimal algorithm would also allocate the demand to data center in ascending order of energy cost while serving the workload. This is because, the objective of any optimal algorithm would also be to minimize the cost of energy consumption. Therefore, the cost incurred to serve the total workload in the greedy solution is same as that of the optimal solution. This proves the statement of the lemma.

7.3.3 Time Complexity Analysis

Time complexity of Shift Workload Algorithm: Let n be the number of data centers and i be the number of energy sources. In Algorithm 4, line 6 has the running time $\mathcal{O}(ni)$, whereas lines 7 and 8 each have the worst case running time of $\mathcal{O}((ni)^2)$. Therefore, the overall worst case running time of the algorithm is $\mathcal{O}((ni)^2)$. In most of the practical scenarios $i \leq 6$ and hence, the worst case running time reduces to $\mathcal{O}(n^2)$.

Time complexity of Request re-routing: We consider a network flow graph with number of nodes $v = n + m + 2$ and number of edges $e = nm + m + n$, where n is the number of data centers and m is the number of client regions. The best known algorithm to solve the minimum cost network flow problem has the worst case running time of $\mathcal{O}(e \log v(e + v \log v))$ [93]. By formulating the problem using minimum cost network flow model, we have a polynomial-time algorithm for fault-tolerant load balancing in GDCs. For the global optimal solution (GOS), two linear optimization sub-problems need to be solved and the worst-case running complexity of the best known algorithm is $\mathcal{O}(n^{3.5}L)$, where the number of decision variables n can be encoded in L input bits. For **P1** and **P2**, the number of decision variables is ni and nm, respectively. Therefore, the combined worst case running time is $\mathcal{O}((nm)^{3.5}L)$. Hence, the asymptotic worst-case time complexity of the proposed algorithm is significantly better than the best known pseudo-polynomial time algorithm for solving the LP version.

7.4 Numerical Results

In this section we evaluate the proposed algorithm (abbreviated as FTLB) and present the results obtained. We compare FTLB with the global optimal solution (abbreviated as GOS) by solving the optimization problems **P1** and **P2**. The

proposed algorithm and optimization problems are solved using MATLAB and CPLEX on a server with Intel Xeon processor and 64 GB of RAM, running Ubuntu 14.04 64-bit OS. The performance metric considered is the normalized average energy cost with respect to the maximum energy cost across both the solutions for a given scenario. In order to quantify the quality of approximation (by FTLB), we define an approximation ratio as

$$R = \frac{\text{Energy cost by FTLB}}{\text{Energy cost by GOS}} \tag{7.22}$$

7.4.1 Experimental Setup

We evaluated the proposed algorithm based on real-world data for data center locations, traffic workload, renewable energy availability, and electricity prices as described below.

Data center characteristics: We considered six data center locations in the USA: California, Iowa, Arizona, Colorado, Illinois and Texas. Electricity prices at these locations are taken from US energy information website [21]. Other parameters used are shown in Table 7.2.

Parameter	**Value**
PUE	1.5
Server power (idle,peak)	100W,200W
Queuing delay (T)	.001
Server processing rate (μ)	30
M_{min}, M_{max}	200, 100000
γ^{max}	80%
Empirical Constant, ϵ	0

Table 7.2: Input parameters

Renewable energy availability: We collected meteorological data from NREL [31]

for year 2016. We used power generation models for wind and solar energy described in [59] and [79], respectively. At each site, we considered 20 wind turbines of capacity 1.5MW each and 10,000 solar panels of 120W each. The cost of generating wind and solar power is obtained by taking the installation cost of 1630 and 3100 $/kW, and life time of 20 and 25 yrs, respectively. We took quarterly average of client demand and renewable energy generated for every hour of the day. Therefore, we use 96 time slots each spanning an hour for the evaluation.

Demand: We used trace of traffic from Wiki dump [68] to build the workload profile. We took quarterly average of client demand for every hour of the day. Considering hourly wikipedia workload as an aggregate demand for every hour, we distribute demand across different client locations proportional to the number of Internet users at each location. The peak demand from the trace is of the order of thousand requests per second, whereas literature suggests that data centers serve requests to the tune of million requests per second [4]. Therefore we have upscaled the demand by a factor of 3000. We took quarterly average of client demand to generate workload across 96 time slots. For every hour we distribute the demand across different client locations proportional to the number of Internet users at each location [67]. We chose nine regions for the demand: Illinois, Tennessee, New York, Arizona, Massachusetts, California, Florida, Missouri, and Louisiana.

7.4.2 Results

We present the results obtained by varying the number of data centers and percentage of failure. In each scenario, we run the FTLB and GOS for various combinations of failure percentage (p) for every hour of the day. Therefore we have $n \times 96$ data points for each scenario (where n is the number of data centers).

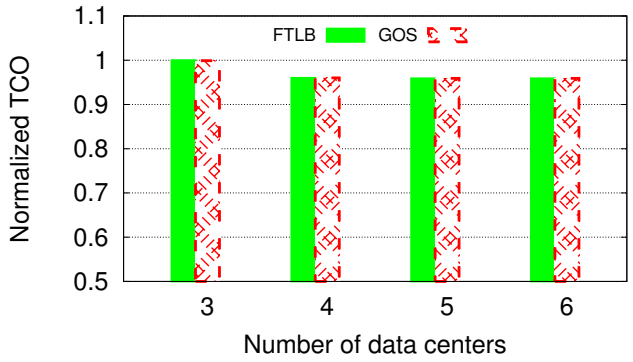


Figure 7.3: Energy cost with varying number of data center

Energy cost comparision

In this experiment we compare the energy cost for FTLB solution with the GOS. We increase the number of data centers from 3 to 6 with 9 client regions, a maximum queuing delay bound of 1ms, and $p = 0.5$. Fig. 7.3 shows the normalized average energy cost for both the algorithms. It can be seen that the cost with $FTLB$ exactly matches the value obtained by GOS. It can be observed that with increasing number of data centers, the energy cost decreases due to more options available to leverage demand multiplexing and electricity price variation.

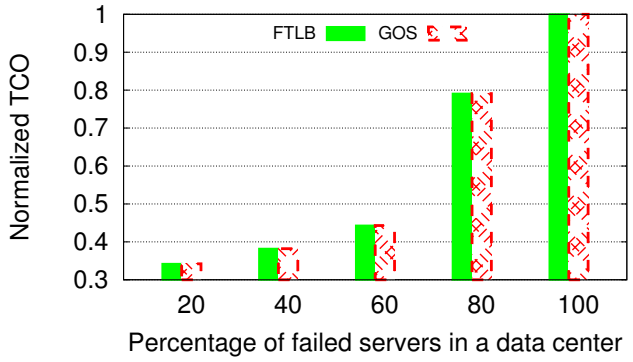


Figure 7.4: Energy cost with varying failure percentage

Impact of failure percentage

In order to understand the impact of failure percentage on energy cost, we evaluate the algorithms with 3 data centers, 9 client regions and set queuing delay bound to 1ms. We vary the percentage of failure from 10% to 100%. Fig. 7.4 shows the results obtained. It can be observed that with increase in failure percentage the energy consumption cost increases. This is because, the system is operating at optimal operating point before failure and after failure, the system has to use the costlier options. We see that across all the scenarios our algorithm closely approximates the GOS.

Impact of demand

To understand the impact of variation of demand on energy cost, we evaluate the algorithm with 3 data centers, 9 client regions, and with varying demand (kx which is the multiple of baseline demand x, where $k \in \{0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6\}$). We set the queuing delay bound to 1 ms. From Fig. 7.5 it can be observed that with increasing demand, energy cost also increases for both the approaches which is quite intuitive.

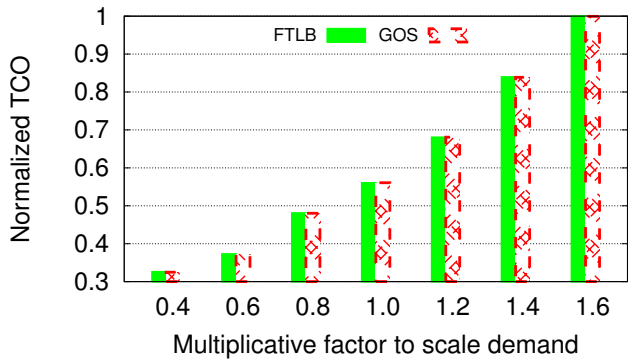


Figure 7.5: Comparing energy cost when demand varies

7.5 Conclusion

In this chapter, we addressed the problem of load balancing in fault-tolerant data centers powered by co-located green energy sources. We modeled the problem using linear programming in two stages. We proposed a distributed algorithm which first spreads the load of failed data center, while minimizing the operating cost (which includes brown and renewable energy cost) and then, re-routes requests to satisfy the delay and green energy usage constraints. Extensive evaluation using real world data shows that the proposed algorithm yields nearly the same results as the global optimal solution, yet it has lower complexity. We conclude that online load balancing algorithms should be more cost-aware to distribute the requests so that the operating cost is also minimized along with the latency.

Chapter 8

Summary and Future Directions

The work in this thesis addressed the research problems of spare capacity planning and load balancing in fault-tolerant GDC. Our objective was to help the operators design and manage GDCs that can tolerate the failure of a single data center without a significant increase in the TCO. We proposed optimization models for cost-aware distribution of spare capacity across the locations and also arrive at optimal load balancing algorithms that minimize the TCO while satisfying the delay constraints.

First, we addressed the problem of cost-aware capacity provisioning for GDCs so that the failure of a single data center failure is masked. We proved that this problem is in NP-hard and proposed an MILP formulation to reduce the TCO. The proposed model is observed to be better than the existing ones, due to its ability to multiplex demand considering the spatio-temporal variation in the electricity prices and the demand. We then extended the model for fault-tolerant GDCs collocated with renewable energy sources. We evaluated the model considering factors such as, green energy availability, demand, delay, and failure percentage, where we observed that the proposed model achieves significant improvement in the TCO despite using costlier green energy. Further, we extended the optimization model for spare capacity provisioning in GDCs powered by both brown and renewable

energy sources, where the data center operators try to meet a target green energy bound with a marginal increase in cost. We showed that even though green energy is costlier, intelligent scheduling of requests and capacity provisioning could lower the operating cost.

Next, we proposed a non-cooperative game theoretic model for load balancing (among front-end proxy servers) with an objective of minimizing the operating cost and obtained the structure of Nash equilibrium. Based on this structure, we designed a distributed load balancing algorithm which provides better fairness to the clients (in terms of service latency) without increasing the operating cost. We compared the performance of the proposed algorithm with the existing approaches to demonstrate that the proposed algorithm approximates the global optimal solution. Finally, we devised a data center-initiated load balancing algorithm to detect the failure early and correspondingly update the load balancing policy quickly. Results showed that the proposed algorithm has a low computational complexity, yet exactly matches the cost obtained using global optimal solution.

Conclusions: From this work, we conclude that it is indeed possible to minimize the cost of running GDCs considering the spatio-temporal dynamics and it is possible to mask single data center failure with no additional cost using the proposed model. Our model leads to a lower cost in designing a fault-tolerant GDC, particularly when the electricity costs vary widely across the data center locations along with higher PUE values. The CACP model is cost effective when the latency requirement is not stringent and a data center does not operate at its peak utilization. Under heavy load, the CACP model can help the provider determine an optimal data center upgrade plan while minimizing the TCO. Next, we conclude that with a suitable model, green energy integration lowers the cost of designing fault-tolerant GDCs (despite green energy being costlier). Our model works well even with uncertainty in the available wind energy and achieves significant reduction in the cost as the

technology advances. We also show that online load balancing algorithms should be cost-aware in allocating the requests so that, the operating cost is also minimized along with the latency. We designed an algorithm that ensures delay fairness to the clients without increasing the operating cost. We expect that our work can help data center operators make informed decisions about capacity planning with green energy usage bounds, different renewable energy sources, spatio-temporal dynamics in electricity price and demand, and varying failure rates.

Future Directions: There are several possible directions in which the work in this thesis can be extended. We list a few immediate extensions of the work related to the problems of capacity provisioning and load balancing.

- Data centers owned by different operators lying in different time zones may form a federation, thereby pooling their resources to improve resource utilization and minimize the operating cost. In such a federation, the load balancing problem can be formulated as a cooperative game among the data center operators. Based on the Nash bargaining solution (NBS) which provides Pareto-optimal and fair solution, one could design an algorithm to compute the optimal load balancing problem policy.

- In this thesis, we considered workload of only short request-response type. However, cloud service operators receive requests for a long-time VM allocation. Modeling the capacity planning across different data centers, based on the expected arrival rate of workload and considering diverse QoS requirements for the application would be an interesting extension.

- We considered a simple queuing model i.e., $M/M/n$ for the server. A more realistic queuing model for production data centers could be used in the work. One can model capacity provisioning in distributed data centers, where the objective would be to keep the expected response time or blocking probability

below a predefined value.

- Only the failure of servers has been considered in our work. We can extend the model to consider the failure of power distribution network, network links, networking elements and a combination of these.

References

[1] "Distributed Virtual Data Center for Enterprise and Service Provider Cloud," http://www.cisco.com/c/en/us/products/collateral/routers/asr-9000-series-aggregation-services-routers/white_paper_c11-694882.html.

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

[3] S. U. Khan, A. A. Maciejewski, and H. J. Siegel, "Robust CDN replica placement techniques," in *Proc. of IEEE IPDPS*, 2009, pp. 1–8.

[4] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *Proc. of ACM SIGCOMM*, 2009, pp. 123–134.

[5] "Three Ways System Downtime Affects Companies and Four Methods to Minimize It," http://dynamic.globalscape.com/files/whitepaper_DevastatingDowntime.pdf.

[6] "New Study: Cost of Data Center Outages (2016)," http://datacenterfrontier.com/cost-of-data-center-outages/.

[7] D. C. Marinescu, *Cloud computing: Theory and Practice*. Morgan Kaufmann, 2013.

[8] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's not easy being green," in *Proc. of the ACM SIGCOMM*, 2012, pp. 211–222.

[9] K. T. Le, "Managing energy usage and cost through load distribution in multi-data-center services," Ph.D. dissertation, Rutgers University-Graduate School-New Brunswick, 2013.

[10] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch, "Autoscale: Dynamic, robust capacity management for multi-tier data centers," *ACM Transactions on Computer Systems (TOCS)*, vol. 30, no. 4, p. 14, 2012.

[11] W. Deng, F. Liu, H. Jin, B. Li, and D. Li, "Harnessing renewable energy in cloud datacenters: opportunities and challenges," *IEEE Network*, vol. 28, no. 1, pp. 48–55, January 2014.

[12] "How Clean is Your Cloud?" http://www.10gea.org/articles/disaster-recovery-cost-of-downtime.

[13] "Renewable energy," https://en.wikipedia.org/wiki/Renewable_energy.

[14] C. Ren, D. Wang, B. Urgaonkar, and A. Sivasubramaniam, "Carbon-aware energy capacity planning for datacenters," in *Proc. of IEEE MASCOTS*, 2012, pp. 391–400.

[15] A. Rahman, X. Liu, and F. Kong, "A survey on geographic load balancing based data center power management in the smart grid environment," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 214–233, 2014.

[16] A.-h. Mohsenian-rad and A. Leon-garcia, "Energy-information transmission tradeoff in green cloud computing," in *Proc. of IEEE GLOBECOM*, 2010.

[17] C. Gu, C. Liu, J. Zhang, H. Huang, and X. Jia, "Green scheduling for cloud data centers using renewable resources," in *Proc. of IEEE INFOCOM WKSHPS*, 2015, pp. 354–359.

[18] S. Chen, Y. Wang, and M. Pedram, "Concurrent placement, capacity provisioning, and request flow control for a distributed cloud infrastructure," in *Proc. of ACM DATE*, March 2014, pp. 1–6.

[19] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008.

[20] I. Goiri, K. Le, J. Guitart, J. Torres, and R. Bianchini, "Intelligent placement of datacenters for internet services," in *Proc. of IEEE ICDCS*, June 2011, pp. 131–142.

[21] "Electricity price in USA," http://www.eia.gov/.

[22] "Electricity price in European countries," http://ec.europa.eu/eurostat/statistics-explained/index.php/File:Half-yearly_electricity_and_gas_prices,_second_half_of_year,_2012\%E2\%80\%9314_(EUR_per_kWh)_YB15.png/.

[23] "Electricity price in Hong Kong," https://www.hkelectric.com/en/customer-services/billing-payment-electricity-tariffs/commercial-industrial-and-miscellaneous-tariff/commercial-industrial-and-miscellaneous-tariff-calculator.

[24] "International industrial energy prices," https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/487759/table_531.xls.

[25] V. Verter, "Uncapacitated and Capacitated Facility Location Problems," in *Foundations of Location Analysis*. Springer US, 2011, vol. 155, pp. 25–37.

[26] Apple, "Apple and the Environment," http://www.apple.com/environment/renewable-energy, 2013.

[27] V. Gao, Z. Zeng, X. Liu, and P. Kumar, "The answer is blowing in the wind: Analysis of powering internet data centers with wind energy," in *Proc. of IEEE INFOCOM*, April 2013, pp. 520–524.

[28] J. L. Berral, Í. Goiri, T. D. Nguyen, R. Gavalda, J. Torres, and R. Bianchini, "Building green cloud services at low cost," in *Proc. of ICDCS*, June 2014, pp. 449–460.

[29] "Google, Apple, Facebook race towards 100% renewable energy target," https://www.theguardian.com/sustainable-business/2016/dec/06/google-renewable-energy-target-solar-wind-power.

[30] "Adding clean and renewable energy to the grid," https://sustainability.fb.com/clean-and-renewable-energy/.

[31] "Renewable Resources Maps and Data," http://www.nrel.gov/midc/.

[32] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment," in *Proc. of IEEE INFOCOM*, 2010, pp. 1–9.

[33] Z. Naseh, "Disaster Recovery & Global Site Load Balancing for Distributed Data Center Applications," https://www.nanog.org/meetings/38/presentations/naseh.pdf.

[34] M. J. Osborne and A. Rubinstein, *A Course in Game Theory.* MIT press, 1994.

[35] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-theoretic, and Logical Foundations.* Cambridge University Press, 2008.

[36] K. Le, R. Bianchini, T. Nguyen, O. Bilgir, and M. Martonosi, "Capping the brown energy consumption of internet services at low cost," in *Proc. of IEEE Green Computing Conference and Workshops (IGCC)*, Aug 2010, pp. 3–14.

[37] D. Weldon, "Downtime numbers are downright distressing," http://www.fierceenterprisecommunications.com/story/downtime-numbers-are-downright-distressing/2013-07-25, 2013.

[38] "Help! My Data Center is Down!" http://www.availabilitydigest.com/public_articles/0704/data_center_outages-lessons.pdf, 2012.

[39] "Amazon Cloud Outage," http://www.datacenterknowledge.com/archives/2012/10/22/amazon-cloud-outage-affecting-many-sites/.

[40] R. Souza Couto, S. Secci, M. Mitre Campista, and L. Kosmalski Costa, "Network design requirements for disaster resilience in iaas clouds," *IEEE Communications Magazine*, vol. 52, no. 10, pp. 52–58, October 2014.

[41] " Sun, wind and sea: Apple details data center renewable energy initiatives," http://www.datacenterdynamics.com/content-tracks/power-cooling/sun-wind-and-sea-apple-details-data-center-renewable-energy-initiatives/98193.fullarticle.

[42] "Renewable Energy Certificate (REC) mechanism," https://www.recregistryindia.nic.in/index.php/general/publics/faqs.

[43] "Carbon offset," https://en.wikipedia.org/wiki/Carbon_offset.

[44] R. Bianchini, "Leveraging renewable energy in data centers: Present and future," in *Proc. of ACM International Symposium on High-Performance Parallel and Distributed Computing*, 2012, pp. 135–136.

[45] "2015 wind technologies market report," https://emp.lbl.gov/publications/2015-wind-technologies-market-report.

[46] "7 charts that show wind power is surging in the us and abroad," http://www.greentechmedia.com/articles/read/7-Charts-That-Show-Wind-Power-is-Surging-in-the-US-and-Abroad.

[47] "Solar electricity cost," http://solarcellcentral.com/cost_page.html.

[48] J. Weinman, "Cloudonomics: a rigorous approach to cloud benefit quantification," *J. Software Technol*, vol. 14, no. 4, pp. 10–18, 2011.

[49] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proc. of IEEE INFOCOM*, 2013, pp. 854–862.

[50] C. Kim and H. Kameda, "An algorithm for optimal static load balancing in distributed computer systems," *IEEE Transactions on Computers*, vol. 41, no. 3, pp. 381–384, 1992.

[51] M. Guo and L. Yang, *New Horizons of Parallel and Distributed Computing*. Springer US, 2006.

[52] K. Lu, R. Subrata, and A. Y. Zomaya, "Towards decentralized load balancing in a computational grid environment," in *Proc. of International Conference on Grid and Pervasive Computing*, 2006, pp. 466–477.

[53] M. Wardat, M. Al-Ayyoub, Y. Jararweh, and A. Khreishah, "To build or not to build? addressing the expansion strategies of cloud providers," in *Proc. of FiCloud*, Aug 2014, pp. 477–482.

[54] A. Gumaste, P. Gokhale, T. Das, M. Purohit, and P. Agrawal, "Using global content balancing to solve the broadband penetration problem in the developing world: case study, india," *IEEE Communications Magazine*, vol. 50, no. 5, 2012.

[55] N. Buchbinder, N. Jain, and I. Menache, "Online job-migration for reducing the electricity bill in the cloud," in *NETWORKING 2011*. Springer, 2011, pp. 172–185.

[56] Y. Zhang, Y. Wang, and X. Wang, "Capping the electricity cost of cloud-scale data centers with impacts on power markets," in *Proc. of ACM International Symposium on High Performance Distributed Computing*, 2011, pp. 271–272.

[57] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *Proc. of IEEE INFOCOM*, 2012, pp. 1431–1439.

[58] E. Kayaaslan, B. B. Cambazoglu, R. Blanco, F. P. Junqueira, and C. Aykanat, "Energy-price-driven query processing in multi-center web search engines," in *Proc. of ACM SIGIR*, 2011, pp. 983–992.

[59] Y. Zhang, Y. Wang, and X. Wang, "Greenware: Greening cloud-scale data centers to maximize the use of renewable energy," in *Middleware 2011*. Springer, 2011, pp. 143–164.

[60] M. Ghamkhari and H. Mohsenian-Rad, "Data centers to offer ancillary services," in *Proc. of IEEE SmartGridComm*, 2012, pp. 436–441.

[61] M. Lin, Z. Liu, A. Wierman, and L. L. Andrew, "Online algorithms for geographical load balancing," in *Proc. of IEEE Green Computing Conference and Workshops (IGCC)*, 2012, pp. 1–10.

[62] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. Andrew, "Greening geographical load balancing," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 2, pp. 657–671, 2015.

[63] "Amazon EC2 pricing," http://aws.amazon.com/ec2/pricing.

[64] M. Ghamkhari and H. Mohsenian-Rad, "Energy and performance management of green data centers: A profit maximization approach," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 1017–1025, 2013.

[65] J. Kleinberg and É. Tardos, *Algorithm Design*. Pearson Education India, 2006.

[66] B. Molina, C. E. Palau, and M. Esteve, "Modeling content delivery networks and their performance," *Computer Communications*, vol. 27, no. 15, pp. 1401–1411, 2004.

[67] "Internet World Stats," http://www.internetworldstats.com/unitedstates.htm.

[68] "Page view statistics for Wikimedia projects," http://dumps.wikimedia.org/other/pagecounts-raw/.

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

[70] Y. Li, H. Wang, J. Dong, J. Li, and S. Cheng, "Operating Cost Reduction for Distributed Internet Data Centers," in *Proc. of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2013, pp. 589–596.

[71] "Is PUE Still Above 2.0 for Most Data Centers? ," http://www.vertatique.com/no-one-can-agree-typical-pue.

[72] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, "Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services," in *Proc. of ACM Symposium on Operating Systems Principles*, 2013, pp. 292–308.

[73] M. Al-Ayyoub, M. Wardat, Y. Jararweh, and A. A. Khreishah, "Optimizing expansion strategies for ultrascale cloud computing data centers," *Simulation Modelling Practice and Theory*, vol. 58, pp. 15–29, 2015.

[74] "Renewable Energy Intermittency Explained: Challenges, Solutions, and Opportunities," https://blogs.scientificamerican.com/plugged-in/renewable-energy-intermittency-explained-challenges-solutions-and-opportunities/.

[75] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. Fathy, "Energy storage in datacenters: what, where, and how much?" in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1. ACM, 2012, pp. 187–198.

[76] J. L. Berral, Í. Goiri, T. D. Nguyen, R. Gavalda, J. Torres, and R. Bianchini, "Building green cloud services at low cost," in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on.* IEEE, 2014, pp. 449–460.

[77] M. Lin, A. Wierman, L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *Proc. of IEEE INFOCOM*, April 2011, pp. 1098–1106.

[78] Y. Guo and Y. Fang, "Electricity cost saving strategy in data centers by using energy storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1149–1160, 2013.

[79] "PVWatts Calculator," http://pvwatts.nrel.gov/pvwatts.php.

[80] "2013 Study on Data Center Outages," http://www.emersonnetworkpower.com/documentation/en-us/brands/liebert/documents/white\%20papers/2013_emerson_data_center_outages_sl-24679.pdf.

[81] D. Grosu and A. T. Chronopoulos, "Noncooperative load balancing in distributed systems," *Journal of parallel and distributed computing*, vol. 65, no. 9, pp. 1022–1034, 2005.

[82] S. Penmatsa and A. T. Chronopoulos, "Game-theoretic static load balancing for distributed systems," *Journal of Parallel and Distributed Computing*, vol. 71, no. 4, pp. 537–555, 2011.

[83] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, 2007, pp. 13–23.

[84] D. Grunwald, C. B. Morrey III, P. Levis, M. Neufeld, and K. I. Farkas, "Policies for dynamic clock scheduling," in *Proc. of the 4th USENIX Symposium on Operating System Design & Implementation*, 2000.

[85] E. Samadiani, Y. Joshi, and F. Mistree, "The thermal design of a next generation data center: a conceptual exposition," *Journal of Electronic Packaging*, vol. 130, no. 4, 2008.

[86] "A revolution in data center efficiency," http://multimedia.3m.com/mws/media/1127920O/2-phase-immersion-coolinga-revolution-in-data-center-efficiency.pdf.

[87] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *ACM SIGPLAN Notices*, vol. 44, no. 3, 2009, pp. 205–216.

[88] S. Tadelis, *Game Theory: An Introduction*. Princeton University Press, 2013.

[89] A. Gandhi, Y. Chen, D. Gmach, M. Arlitt, and M. Marwah, "Minimizing data center SLA violations and power consumption via hybrid resource provisioning," in *Proc. of IEEE Green Computing Conference and Workshops (IGCC)*, 2011, pp. 1–8.

[90] "fmincon," http://in.mathworks.com/help/optim/ug/choosing-the-algorithm.html.

[91] "Server Efficiency: Aligning Energy Use With Workloads," http://www.datacenterknowledge.com/archives/2012/06/12/server-efficiency-aligning-energy-use-with-workloads/.

[92] B. Subramaniam and W. Feng, "Enabling efficient power provisioning for enterprise applications," in *Proc. of 14th IEEE/ACM CCGrid*, 2014, pp. 71–80.

[93] J. B. Orlin, "A faster strongly polynomial minimum cost flow algorithm," *Operations Research*, vol. 41, no. 2, pp. 338–350, 1993.

Publications Related to Thesis

Published/Accepted

Journals

1. **Rakesh Tripathi**, S. Vignesh, Venkatesh Tamarapalli, "Optimizing Green Energy, Cost, and Availability in Distributed Data Centers", IEEE Communications Letters, vol. 21, no.3, Mar. 2017, pp.500-503.

2. **Rakesh Tripathi**, S. Vignesh, Venkatesh Tamarapalli, Deep Medhi, "Cost Efficient Design of Fault Tolerant Geo-Distributed Data Centers", IEEE Transactions on Network and Service Management,vol. 14, no.2, June 2017, pp.289-301.

3. **Rakesh Tripathi**, S. Vignesh, T. Venkatesh, A. T. Chronopoulos, and H. Siar, "Non-cooperative Power and Latency Aware Load Balancing in Distributed Data Centers", Journal of Parallel and Distributed Computing, Elsevier, Vol. 107, Sep. 2017, pp. 76-86.

Conference Proceedings

1. **Rakesh Tripathi**, S. Vignesh, and T. Venkatesh, "Cost-aware Capacity Provisioning for Fault-tolerant Geo-distributed Data Centers", Proc. of

8th International Conference on Communication Systems and Networks (COMSNETS), Bengaluru, India, Jan. 2016.

2. **Rakesh Tripathi**, S. Vignesh, and T. Venkatesh,"Minimizing Cost of Provisioning in Fault-tolerant Distributed Data Centers with Durability Constraints", Proc. of IEEE International Conference on Communications (ICC), Malaysia, May 23-27, 2016.

3. **Rakesh Tripathi**, S. Vignesh, and T. Venkatesh,"Towards Cost-Effective Capacity Provisioning for Fault-tolerant Green Distributed Data Centers", Proc. of IEEE International Conference on Advanced Networks and Telecommunication Systems (ANTS), Bengaluru, Nov. 2016.

Communicated

Journals

1. **Rakesh Tripathi**, S. Vignesh, and T. Venkatesh,"Provably Cost Optimal Load-balancing in Fault-tolerant Internet Data Centers", Future Generation Computer Systems, Elsevier

Brief Biography of the Author

Rakesh Tripathi was born in a town called Gomia (in Jharkhand state), India on 21st May, 1982. After completing his schooling in Gomia, he has received the B.Tech degree in Information Technology, from UPTU Lucknow, India, in 2004 and the M.Tech. degree in Information Technology, from the Tezpur University, India in 2006. After masters, he joined NERIST, Nirjuli as a lecturer in Department of Computer Science and Engineering. He worked in NERIST between July 2006 to August 2008. Further, he worked in NIT Raipur as Assistant Professor in the Department of Information Technology between August 2008 to July 2014. On July, 2014, he joined Department of Computer Science and Engineering to pursue his PhD research under the supervision of Dr. T. Venkatesh. His research interests include Data center networks, game theory in networks, distributed algorithms, performance modeling of communication Systems, etc.