GLOBAL COST DIVERSITY AWARE SCHEDULING ALGORITHM FOR HETEROGENEOUS DATA CENTERS

by

Ananth Narayan Sankaranarayanan

B. Eng., Visveshwaraya Technological University, 2002

P.G.D. (IT), International Institute of Information Technology (Bangalore), 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2011

APPROVAL

Name: Ananth Narayan Sankaranarayanan

Degree: Master of Science

Title of Thesis: Global Cost Diversity Aware Scheduling Algorithm for Heterogeneous Data Centers

Examining Committee:

Dr. Arthur Kirkpatrick
Computing Science, Simon Fraser University
Chair

Dr. Alexandra Fedorova
Computing Science, Simon Fraser University
Senior Supervisor

Dr. Anoop Sarkar
Computing Science, Simon Fraser University
Supervisor

Dr. Jiangchuan Liu
Computing Science, Simon Fraser University
Examiner

Date Approved: August 2, 2011

Partial Copyright Licence

Abstract

Internet based companies service user requests from multiple data centers located around the globe. These data centers often house heterogeneous computing infrastructures and draw electricity from the local electricity market. Electricity is also consumed for cooling the data centers, and the total costs often run into millions of dollars. Reducing operating cost, therefore, is an important and challenging problem.

In this work, we propose a novel solution which schedules user requests to geographically diverse data centers and exploits the server heterogeneity, data center power usage efficiency, and global electricity market diversity to reduce operating costs. We evaluate our solution using a real-world workload, electricity prices, and power efficiency values. We show that our scheduling algorithm achieves a cost savings up to 14% over a load balancing scheme that distributes requests evenly across data centers, and outperform existing solutions which do not exploit either electricity market diversity or data center hardware diversity.

To my parents and my sister.

"Bring forth what is true; Write it so its clear. Defend it to your last breath."

— Ludwig Boltzmann, quoting Faust

Acknowledgments

It is my pleasure to express my gratitude to the many people who have provided their support to me over the course of my masters studies.

First, I would like to thank my academic supervisor Prof. Alexandra Fedorova, without whose support and input, this work would not have been possible. Sasha's feedback has been key in providing both focus and direction to research work I have embarked upon. I am also greatly indebted to her for the feedback on various soft skills that I believe would be key to success and growth in my professional life.

I would like to thank my supervisor Prof. Anoop Sarkar, and the faculty under whom I undertook coursework which has helped me hone my skills as an engineer and as a researcher. Thanks to all the folks from FASNet - Lee Grenough, Ching Lin, Stephen Nix, James Peltier, Ching Tai Wong - who provided timely closure to the computing infrastructure issues.

It was a great pleasure to share the office with Sergey Blagodurov, Mohammad Dashti, Tyler Dwyer, Sergey Zhuravlev, and I will fondly remember all the discussions we have had on everything under the sun. I would also like to thank my friends Ajit Khosla, Baskaran Sankaran, Bitan Roy, Simran Sarai, Somsubhra Sharangi, Suvayu Ali, and Udit Pareek for all the help and support they offered.

Finally, I would like to thank my parents and my sister, to whom I dedicate this thesis, for the encouragement and support they provided.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Internet is becoming accessible to more and more people, and simultaneously, has become an important medium for the distribution of content, services, and software.
Organisations like Google, Yahoo, Facebook, and Amazon, have almost all of their services hosted on-line; companies like Microsoft are increasing the software and services available on cloud computing infrastructure. To fulfil the data and computation needs placed by ever-connected consumers, increasing numbers data centres are either being setup or leased by these companies across the globe. Companies also use co-location facilities, which are data centers where multiple companies place their server, networking, and storage equipment. Using a co-location facility removes the complexity of setup and maintenance of the building, hardware, and cooling infrastructure from the company and pushes it onto the co-location provider, thereby allowing the company to use the human resources thus freed in other aspects of business.

A data center houses a large number of servers and their associated components such as storage and communication systems. Typical data centers house hundreds, maybe even thousands of servers and serve as the backbone for a variety of Internet services. A data center can be broadly viewed as consisting of hardware infrastructure, software infrastructure, and a cooling infrastructure. The hardware infrastructure can be further broken down as consisting of the servers that provide computational power, storage equipment, and the

networking infrastructure that provides local and remote network access. Software infrastructure includes operating systems (OS), administrative software for data center monitoring and management, and cluster level infrastructure (or middleware) that provide transparent access of resources to application level software. Finally, the cooling infrastructure includes equipment that provides cooling for the facility and ensures the right operating environment.

1.1 Types of data centers

Based on the layout, data centers can be broadly classified into one of the 4 tiers listed in Table 1.1. This classification however, is not precise and commercial data centers typically fall between Tiers 3 and 4 [28]. A higher tier implies an improvement in resource availability and reliability, but is accompanied with an increase in power consumption.

Type	Description
Tier 1	Single path for power and cooling; no redundant components.
Tier 2	Redundancy added to Tier 1 thereby improving availability.
Tier 3	Multiple power and cooling distribution paths of which one is active.
Tier 4	Two active power and cooling paths and redundant components on each path.

Table 1.1: Data Center Classification

Many studies [45, 41, 34] in the recent years have the shown how the electricity cost of running these data centers has emerged as a serious concern for operators; Qureshi et al [42] have recently shown the annual electricity costs for a data center to be in order of several million dollars.

Consequently, reducing the annual electricity costs – by reducing the server power consumption – has been an active area of research. Reducing the cost of operation allows the data center operator to invest in newer hardware and supporting infrastructure (storage and communication systems) and consequently, provide improved services to their clients. Prior studies [14, 11, 13] have reported that servers can draw close to 60% of their peak power consumption when idle, and that the global electricity costs for data centers have been reported as running into the billions. Shah et al [45] estimate that the operational

costs for a data center reach up to three million US dollars annually; Koomey et al [34] in 2007 estimated the total annual electricity bill for data centers in the USA was in excess of two billion, and more than seven billion globally. In the case of co-location facilities, the electricity costs are passed on directly by the data center operators to their customers - companies who host their services in the co-location facility.

While newly setup data centers might be homogeneous in their design, hardware upgrades over time would result in the data center becoming heterogeneous – servers begin to differ in their power consumption and computation capability. Prior work has shown that heterogeneous computing provides improved energy efficiency [11], and this aspect of heterogeneity provides us with an opportunity for power and cost savings.

1.2 Cooling & Power Delivery

While on the one hand, power is consumed by the computing equipment, a data center typically has additional power consumption resulting from cooling and power delivery. Cooling is necessary to ensure that the heat generated by servers is removed from the data center so that the facility can be maintained within a recommended range of temperature and humidity. In addition, power is consumed by the equipment that is necessary to supply power throughout the data center. The power consumed by the cooling and power delivery equipment can be substantial, therefore, data centers use various efficiency measures to reduce these costs. For example, hot and cold aisle containment ensures that the warm air exhaust of servers does not mix with the cool air supplied by the air conditioning, thereby improving the cooling efficiency resulting in lower cooling costs. Companies are investing into various technologies and mechanisms to reduce cooling costs [31, 32].

The American Society of Heating, Refrigeration, and Air Conditioning Engineers (ASHRAE) [8] defines the Power Usage Efficiency (PUE) – a metric of the energy efficiency of the data center. PUE is the ratio of the power delivered to the data center to the power consumed by the computing equipment; if the PUE is 1 (the ideal case), then all of the power supplied is used by the Information Technology (IT) equipment. However, such is not the case and the PUE of a data center is always above 1, because cooling and power delivery require power.

1.3 Service Replication

Having the computing and storage infrastructure replicated across multiple data centers allows the companies that deliver software services to provide low latency service to end users. Further, the inherent replication of the service supports robustness and reliability, consequently providing high availability. The location of the data centers can be influenced by a variety of cost factors including labour, electricity, water, taxes etc. Further, the data centers must be located such that the end users of the services hosted in the data center do not experience long delays resulting from the geographical distance between the data center and the end user.

1.4 Leveraging local electricity prices

The data centers can be located geographically far apart from each other, and we expect them to buy electricity from local markets. There could be significant differences in the price of electricity across the various geographical regions. Further, electricity pricing is being done based on hourly consumption basis therefore, we have an opportunity to exploit this variation in price by intelligently dispatching the requests to the data centers where the electricity is cheaper.

1.5 Server Heterogeneity

As mentioned earlier, a data center contains a large number of servers. Initially, the data center could be homogeneous, that is, it consist of servers that exhibit similar power-performance profiles. However, due to hardware upgrades and server replacements, the data center is expected to become heterogeneous over the course of time. Heterogeneous computing has been shown to provide better performance and power consumption [10, 39]. Heterogeneity therefore offers yet another dimension that we can use to reduce the costs without significant loss in performance. Research has also shown that hybrid data centers, that is, data centers containing a mix of high and low performance machines have the potential to provide energy efficient performance [15] .

1.6 The Challenge

Consider a web application which is replicated across multiple data centers that are located in geographically diverse regions. Wikipedia, the popular encyclopedia is an example of such an application; Wikipedia content is served from two data centers, one in Europe and one in North America. Because the data centers are located across diverse geographical regions and timezones, and, electricity prices vary over the course of the day, we can leverage the variation to reduce cost for data center operators by scheduling computational tasks on servers located in the data center with the lower electricity price.

However, quality of service is an important performance metric. Servicing requests in the data center with lower electricity price could result in increased load on the data center, resulting in increased processing times on the servers, and end users experiencing higher latency which results in poor user experience. Therefore, while on the one hand, we need to reduce the costs for data centers, it is imperative that end user experience does not suffer.

In addition, the data centers are heterogeneous, that is, they consists of servers that exhibit differences in power consumption and performance and heterogeneity must be accounted for when scheduling requests. For example, assigning requests to servers that provide high service rate could result in higher power consumption, because, typically such servers also tend to consume more power. On the other hand, assigning requests to the servers that consume low power would result in lower power consumption, and consequently lower electricity costs, but would also result in higher end user latency because such servers provide low computational capacity.

Thus, we have the opportunity to reduce operational costs for data centers, but need to do so under performance constraints. Our contribution is a scheduling framework that leverages electricity price across timezones to schedule HTTP requests across heterogeneous data centers located in diverse geographical locations. While individual aspects – such as power/performance modelling and delay modelling have been studied – that we use in our problem definition, ours is the first work that investigates cost reduction via scheduling for heterogeneous data centers. We consider all monetary aspects of operational costs - computation costs, non-computation costs (an oft-neglected aspect), and bandwidth costs - and thus provide a complete framework for scheduling across multiple, heterogeneous data

centers. The most important aspect being that the cost savings are obtained without loss of service quality.

The rest of the document is organised as follows: we describe the problem and the solution in Chapter 2. Chapter 3 captures the methodology, experiments, and evaluation metrics and the results are discussed in Chapter 4. Related work is presented in Chapter 5 and we conclude in Chapter 6.

Chapter 2

Problem Description and Solution

The expenses incurred by a data center comprises of many components such as the site rent, operating staff salary, and utilities bills. Because data centers house large number of high performance servers and other power consuming devices, and the electricity cost for running these hardware has emerged as a significant expense in operating a data center.

Consider a web application replicated across multiple data centers, such that any one data center can service any of the client requests. The incoming requests can be serviced by either data center, and therefore, can be redirected by a load balancing front-end to one of the available data centers. The load balancing front-end receives client requests and executes a request scheduling algorithm to assign requests to the data centers. The front-end, therefore, is along the critical path networking link that connects the clients to the data centers; physically, the front-end could be placed either at one of the avilable data centers or, at a suitable location such that the requests do not suffer added networking latency. A round robin assignment provides a simple technique for scheduling requests to data centers. If the electricity price in the data centers differ, then, requests can be assigned to the data center with the least electricity price, thus providing an intuitive and greedy algorithm that can reduce electricity costs. Consequently, we can define two scheduling algorithms which are described below.

2.1 Even Distribution Scheduling

Even Distribution scheduling is a naïve algorithm where the requests are distributed evenly across all available data centers; timezone, electricity costs, and other factors are not considered. With the even distribution scheduling, data centers execute an equal number of requests. This scheduling algorithm provides us with a baseline to compare other scheduling algorithms that consider the electricity price.

Algorithm 1 Even Distribution Scheduling

$n = 1$
$datacenters = \{\text{set of data centers}\}$
$N = |datacenters|$
while $true$ **do**
  $assign(request, datacenters_n)$
  $n = (n+1)\%N$
end while

2.2 Least Electricity Price Scheduling

In the Least Electricity Price scheduling, requests are redirected to the data center located in the region where the current price of electricity is the least across all timezones and data centers. With such a algorithm, we can clearly reduce the electricity costs associated with servicing the requests. We name the data center at the location which currently has the lower electricity price as the 'primary data center'. Requests are dispatched to data centers in the increasing order of electricity price if the primary data center is operating at full capacity. The least Electricity price algorithm is an intuitive, greedy algorithm.

2.3 Local Scheduling

Further to the global load balancing, where requests are distributed between global data centers, we define two 'local' scheduling algorithms - algorithms used within a data center to schedule requests to servers.

Algorithm 2 Least Price Scheduling

```
datacenters = {set of data centers}
N = |datacenters|
sortOnPrice(datacenters)
while true do
  n = 1
  while n ≤ N do
    if spareCapacity(datacenter_n) then
      assign(request, datacenter_n)
    else
      n = (n + 1)
    end if
  end while
end while
```

1. Proportional Fair (PF) : In this algorithm, the requests are distributed across all servers in proportion of their service rates, where the service rate captures the maximum number of requests that a server can handle in unit time. Different server types execute different number of requests, however, all servers in the data center exhibit comparable utilisation; utilisation, in this case, being the ratio of the requests executed to the server's service rate.

2. Fastest Server First (FSF) : In this algorithm, requests are assigned to the servers in the decreasing order of their service rates, that is, requests are assigned to servers which provide a higher service rate, and progressively to the slower servers.

2.4 Global Cost Aware Scheduling

Thus far we have discussed two simple algorithms to schedule requests across multiple data centers. While the Least Price scheduling attempts to leverage electricity prices, both algorithms are agnostic of other aspects such as heterogeneity, data center efficiency, bandwidth etc. We propose a solution that leverages all these aspects to reduce data center costs while at the same time keeping the latency acceptable by means of intelligently scheduling requests across data centers. However, there is no free lunch and a variety of constraints

must be respected in the process of scheduling. In the rest of the section, we formulate an optimisation problem and, define and discuss the various constraints that are applicable.

2.4.1 Total Cost

We denote the total cost incurred at time interval t as $Cost(T)$, and assume that the total cost incurred will be minimised if we minimise the cost for each time interval. The cost of servicing requests can be broadly broken down into two components : the electricity costs, and the communication costs. Electricity costs capture the costs incurred due to consumption of electricity ($ECost(t)$) and communication costs ($CCost(t)$) captures the monetary cost of bandwidth consumption.

$$Cost(t) = ECost(t) + CCost(t) \tag{2.1}$$

With N data centers, we denote the cost incurred by a data center $n \in \{1,\ldots,N\}$, at time t as $Cost_n(t)$. Therefore, in order to reduce the cost, we

$$\text{minimise} \sum_{n=1}^{N} Cost_n(t) \tag{2.2}$$

2.4.2 Data Center Electricity Cost

We denote the electricity cost of a time interval t as $ECost(t)$. The electricity cost at a data center is a function of the power consumed and the electricity price. Let the unit electricity price at data center n at time t be $\mathscr{E}_n(t)$ and the power consumption during the same interval at the same data center be $W_n(t)$.

$$ECost_n(t) = \mathscr{E}_n(t) W_n(t) \tag{2.3}$$

The electricity prices for the current interval can be periodically updated at the global load balancing front end from real-time, local market predictions known as ‘spot price’. The power consumed by a data center includes the power required for servers, the power required for cooling, and power lost due to inefficiencies in the power delivery mechanism.

Given the power usage effectiveness (PUE) of a data center and the power consumed by the IT equipment, we can calculate the total power consumption of a data center as

$$W_n(t) = PUE \times TotalServerPower \tag{2.4}$$

2.4.3 Server Power Consumption

Consider the case of multiple servers which are capable of executing a given application; these servers do not consume a constant amount of power. Changing the processor frequency, for example, results in a change in power consumed by the processor, which influences the power consumed by the server. Dynamic Voltage and Frequency Scaling (DVFS) is a commonly used mechanism for power and performance management in processors.

Over the years, technology improvements have reduced the power consumption of computing components, while simultaneously providing better performance. Therefore, different servers are expected to show differences in their power consumption.

The power consumed by a data center depends on the power consumed by the servers that constitute it. In order to estimate the power consumed by the data center, we need to obtain the power consumed by individual servers, and as just discussed, servers differ in their power consumption, therefore we need a model that captures the power consumed by each server.

Let us assume that data center n has S_n heterogeneous servers, with H types of servers and an $S_n^h : h \in H$ number of each type such that $S_n = \sum S_n^h$. Each server type has a power profile composed of an active power consumption h^{active} and idle power consumption h^{idle}, and a capability profile μ^h, where μ^h is the service rate of server type h. Therefore, the total capability of a data center can be given as $N_C = \sum S^h \mu^h$, over all server types $h \in H$.

A server is considered to be fully utilised when the request rate equals its service rate; the same can be said for a data center when we have full utilisation of each server within it. The power consumed by the data center is the sum of active power profiles of all servers. Let us assume that a data center n is loaded to x_n of its total capacity. The number of requests being serviced by the data center is therefore given as $x_n \times N_C$. Let y^h be fraction of the requests being serviced assigned to server type h. Therefore, utilisation of server type

Symbol	Description
N	Number of data centers
$C_n(t)$	Cost at data center n at time t
$Cost(t)$	Total cost across all data centers
$\mathscr{E}_n(t)$	Unit electricity cost
$W_n(t)$	Power Consumption at data center n
S_n	Number of servers at data center n
H	Type of servers
S^h	Number of servers of type h
N_C	Processing capacity of a data center
h^{active}	Active state power consumption of server type h
h^{idle}	Idle state power consumption of server type h
x_n	Load of data center n
y^h	Load of server type h
μ^h	Processing capacity of server type h
μ_n	Average processing capacity of data center n
D_n	Delay incurred by data center n
L^{st}	The latency along the link from source s to destination t
b	Size of the HTTP request/response.

Table 2.1: List of symbols used

h can be given as

$$U_n^h(t) = \frac{N_C}{S^h \mu^h} \times x_n y^h \tag{2.5}$$

And the power consumption of each server type h can be given as the sum of active idle power.

$$W_n^h(t) = h^{active} \times U_n^h(t) + h^{idle} \times (1 - U_n^h(t)) \tag{2.6}$$

We provide additional discussion and experimental results validating this model in Section 4.1.

Summing over all server types, the total power consumption of the data center n is obtained as

$$W_n(t) = \sum_{h \in H} W_n^h(t) \tag{2.7}$$

For brevity, we drop the time suffix for the rest of our discussion.

2.4.4 Communication Costs

The bandwidth costs depends on the bandwidth consumed and the price of bandwidth along the networking link on which data is transferred. The data transferred depends on the size

(in bytes) of the HTTP requests that are redirected from the front-end to the data center, the size of the response sent from the data center to the user. The communication costs incurred is given as

$$CCost_n = (P_{FD_n} \times b^{req} + P_{D_nU} \times b^{res}) \times N_C x_n \tag{2.8}$$

where, P_{FD_n} represents the price of bandwidth along link from the front end to the data center, P^{D_nU} represents the price of the link from the data center to the source region of the request, b^{req} and b^{res} capture the size of a single HTTP request and HTTP response respectively. $N_C x_n$ is the number of requests that were serviced by data center n.

Bandwidth is made available to on demand and pricing follows a tiered model. Higher dedicated bandwidth consumption is less expensive compared to lower bandwidth consumption. However, if the consumption limit is violated, then the additional bandwidth is priced much higher. Bandwidth pricing can therefore be visualised as a stepped function.

For example, the bandwidth prices listed on Colocation.com show that while a 499 Mbps connection costs $45 per Mbps, additional bandwidth costs up to $150 per Mbps. Therefore, bandwidth consumption limits (B_{max}) must not be violated.

There are two links involved in the path of a request-response pair: a user to front-end link, and a front-end to data center link. The front-end receives user requests which are dispatched to one of the data centers; the data center then processes the request. Consequently, the client request and the server response face two different network delays: first, the network delay on the end-user to front-end link L^{UF}, and subsequently on the front-end to data center link L^{FD}. The front-end hides the complexities of network that constitutes the back-end, namely the server clusters and the data centers; all client requests are sent to the front-end and all server responses pass through the front-end. The clients cannot connect to the individual servers nor can the servers respond to the client by bypassing the front-end.Therefore, the total network latency experienced by the user is

$$L_{Total} = 2 \times (L^{UF} + L^{FD}) \tag{2.9}$$

The network latency between any two points depends on the distance between the two points and the bandwidth saturation on the networking link between the two points. We assume that the front-end is connected to each data center with a high speed, high capacity

link, therefore, the bandwidth saturation does not affect the latency, that is, the latency on a link stays constant. The front-end to data center latency faced by the request depends on which data center is chosen to service the request.

2.4.5 Application Response Time

The important quality metric of a scheduling algorithm is the delay for servicing a request, which is defined as sum of the waiting time in the server queue and the processing time. In a homogeneous server setup, all servers are assumed to have equal, high service rate, and the processing time component can be ignored. However, in a heterogeneous setup the processing time can be a significant component, specially for the slower machines – machines with comparatively lesser service rate. While there are several models present for analysing homogeneous server systems, modelling of heterogeneous server systems is complex and the service rate of each server type must be considered separately.

From queueing theory, the time spent by a request in the system is given as

$$D = \frac{1}{\mu - \lambda} \tag{2.10}$$

where, μ is the service rate of the system and λ is the request rate. In a heterogeneous server setup, server types differ in their service rates; for example, Yu et al model the behaviour of the server cluster by using the average service rate of the servers that constitute the cluster [53].

The request rate for a server depends on the number of requests currently assigned to it, and in the context of an HTTP based application, the service rate of a server is the request rate that it can support without resulting in client errors and timeouts. At any request rate exceeding the service rate, client connections are either rejected or result in a timeout. If the request rate is below the service rate, then Equation 2.10 can be applied to obtain the time spent by the request in the system.

$$D^h = \frac{1}{\mu^h - \lambda^h} \tag{2.11}$$

where h is the server type, μ^h is the service rate of server type h and λ^h is the request rate on server type h.

Equation 2.11 suggests that an increase in the request rate will result in an increase in the time spent by each request in the system, which is intuitively correct. We note that, for a work conserving system (that is, where no jobs are dropped after they are allocated to servers) the allocation of requests to a server can never exceed the servers processing capacity. We enforce this as the following constraint:

$$\mu^h > \lambda^h \tag{2.12}$$

and the time spent in the system D^h must not exceed a delay bound, D_{Bound}. Work conservation in the context of HTTP based client-server application implies that clients do not encounter server side errors and connection timeouts.

2.4.6 Load Distribution Constraints

Let the request rate seen at the front end be λ. If there are N data centers with equal capacities N_C, and they are loaded to x_n of their capacity by the load balancing algorithm, then the load at each data center is $\lambda_n = N_C \times x_n$. The individual loads must add up to the total incoming load (seen at the front end) as the following.

$$\lambda = \sum_{n=1}^{N} \lambda_n = \sum_{n=1}^{N} N_C x_n \tag{2.13}$$

Inside each data center we have H server pools, each servicing a fraction, y^h, of the incoming load (λ_n). Therefore, the requests served by each server pool must add up to the total incoming load for that data center.

2.4.7 Putting it together

Summarising the discussion thus far, we can state the problem as :

$$\begin{aligned}
&\text{minimise} \quad \sum_{n=1}^{N} Cost_n(t) && \text{(P)}\\
&\text{subject to } \lambda = \sum_{n=1}^{N} \lambda_n && \text{(1a)}\\
&\lambda_n y^h < \mu^h && \text{(1b)}\\
&0 \leq y^h \leq 1 && \text{(1c)}\\
&(b^{req} + b^{res}) \times N_C x_n \leq B_{max} && \text{(1d)}\\
&D^h \leq D_{Bound} && \text{(1e)}\\
&L_{Total} \leq L_{Bound} && \text{(1f)}
\end{aligned}$$

(2.14)

That is, we minimise the power consumed while ensuring that the following constraints are not violated:

- All requests received must be serviced. The sum of the requests serviced by each data center must be equal to the number of requests received by the front end.
- Any given server can execute between 0 and 100% utilisation.
- The number of requests executed by each server must not exceed its service rate.
- The data transferred on a given networking link must not exceed the bandwidth constraints.
- The time spent by a request in the system must not exceed the processing delay bound set on the system. This constraint allows us to specify response time bounds on the system.

2.4.8 Time Series Analysis

The Global Cost Diversity Aware (GCDA) scheduling algorithm schedules the incoming requests among all available data centers such that the cost incurred is minimised. However,

solving the linear optimisation requires knowledge of the number of requests that need to be serviced in the time interval. This information however is not available upfront in a real time setup. Consequently, in order to obtain the number of requests seen in the next interval of time, we do a time series analysis, where the number of requests executed in the previous time intervals are used to predict the number of requests expected in the next time interval. The number of requests executed forms a time series, and time series analysis techniques can be used to forecast the number of requests expected in the next time interval - we use an implementation of an exponential smoothing model provided by Hyndman et al [30]. The observations of the request serviced per minute are scalar measurements, recorded sequentially with fixed time intervals, and consequently form a uni-variate time series.

Predictions made by any forecasting method however, will not provide values which match exactly with the actual number of requests seen - the predictions could be either lower or higher than the actual number of requests serviced. In order to avoid any response time violations, we err towards maintaining performance by providing the time series analysis with a larger number of requests than actually serviced. Consequently, the predictions made by the forecast are marginally higher than the actual number of requests serviced. For the time series analysis, the requests serviced in the previous five minutes are used to make the prediction for the next one minute; the requests are weighted equally.

2.4.9 Scheduling

The Global Cost Diversity Aware (GCDA) scheduling algorithm attempts to obtain a schedule that assigns requests to servers across multiple data centers such that the resulting cost is minimised, without loss of quality of service. The problem description therefore, requires models about server power consumption, information about server service rate, network bandwidth etc. In the preceding sections, we discussed the problem formulation and the various models and inputs.

The scheduling algorithm needs to be executed at regular intervals to schedule requests to servers; it is executed on a load balancing front end device, which can be located at any of the data centers. We assume that the scheduling algorithm is provided with the current

electricity price and power usage effectiveness of each data center, and this information is updated at regular intervals - typically once an hour. We also assume that the scheduling interval is longer than the time taken to solve the optimisation problem.

Algorithm 3 Global Cost Diversity Aware Scheduling

```
datacenters = {set of data centers}
N = |datacenters|
server = {set of all servers}
S = |datacenters|
while true do
    electricity_prices ← getCurrentPrices(datacenters)
    pue ← getCurrentPUE(datacenters)
    requests ← forecast();
    schedule ← getschedule(requests, electricity_prices, pue)
    while currentTimeInterval do
        assign(schedule, request)
    end while
end while
```

The time series analysis requires the request rates from the first few minutes to be able to make forecasts. Therefore, we use the request rate seen in the first five minutes as the input for making forecasts. Therefore, for the first few minutes we use the Even Distribution scheduling, with the Proportional Fair local scheduling. Solving the optimisation problem provides us with a schedule which captures the number of requests that must be executed by each server, which is used to schedule requests to servers in a weighted round robin fashion.

Chapter 3

Methodology Overview

Our evaluation involves simulation of the proposed scheduling algorithms using data centers located in two different timezones, and using real world values for the workload trace, electricity prices, bandwidth prices, and power usage efficiency. In this section, we describe the experimental methodology for the evaluation.

3.1 Power and Performance

In Section 2.4.3, we presented a model for server power consumption, where we assumed a linear relationship between the request rate and the total power consumed. We show by experiments, that this assumption is valid. In order to measure the power consumption and the service rates of individual servers, we used a controlled workload while simultaneously measuring the power consumption and the number of client errors encountered.

The client load was generated using the `siege` [33] application which issued valid HTTP requests to the system under test (SUT). The siege application was provided with a list of valid URLs to issue to the SUT and the number of virtual clients to simulate. As the name suggests, each virtual client signifies an independent client requesting pages from the web application. Increasing the number of virtual clients results in more requests being serviced by the web server – an increase in the load on the web server.

The system under test (SUT) was a web server that was installed with the Apache web server and the Mediawiki web application. Mediawiki requires a database back-end. The

SUT hosted Mediawiki and the MySQL database was hosted on a different server, therefore, the utilisation and power consumption are due to the Apache web server running the Mediawiki application. The SUT was connected to an external Extech 83038 power meter, which was used to measure the power consumption of the server while executing the workload. The power measurements, therefore, captured server power measurements and not the measurement of the processor alone.

In order to measure the service rate of the SUT, we measured the number of client errors encountered as the request rate was increased, and the request rate at which a client error were encountered was taken to be the service rate of the SUT. At any request rate above the service rate, client side errors increased.

3.2 Workload

We use the request trace from Wikipedia [49] for our workload. The request trace provides us with the timestamps and the URI of the pages requested but is anonymised; it does not contain any information on the user or the IP address of the client from which the URI was requested. The workload inherently captures the daily variation in the number of requests seen over time. Requests typically show a diurnal pattern, where the load is high in some parts of the day and low during others.

3.3 Data Center Setup

We consider a heterogeneous setup, with each data center consisting of three types of servers where each server type exhibits a different service rate and energy consumption footprint, and an equal number of servers of each type. The requests in the trace show a diurnal pattern, where the load is high in some parts of the day and low during other periods of time. Consequently, we can expect to see variation in the utilisation of the servers over time. Wikipedia uses hosting facilities in Florida, USA and The Netherlands and we base our data center setup on the same regions for obtaining the hourly electricity prices. For one specific experimental configuration, the high service rate machines are assumed to

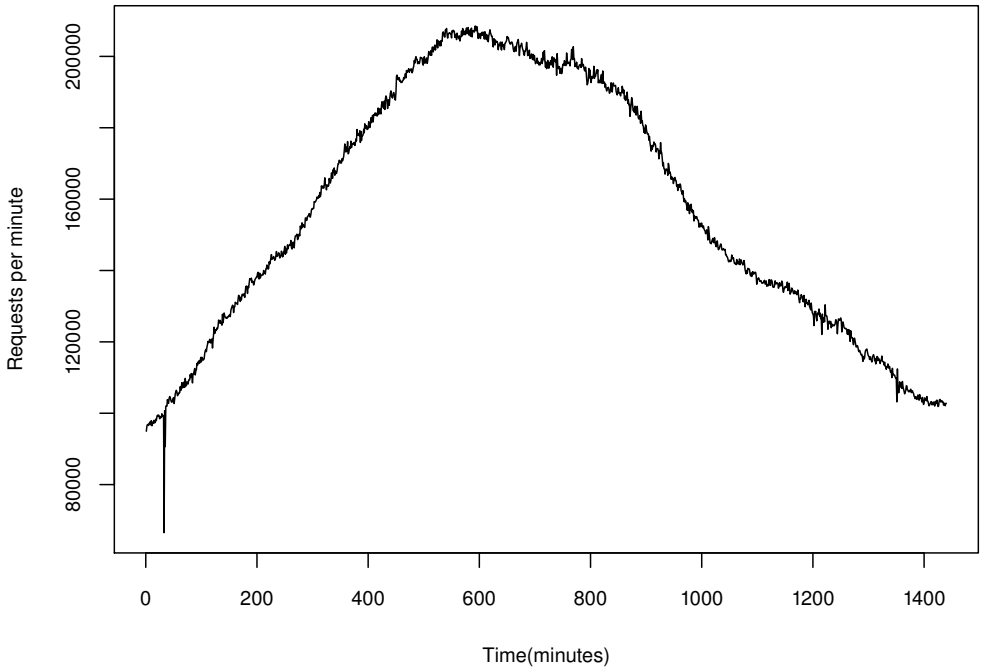


Figure 3.1: Workload pattern

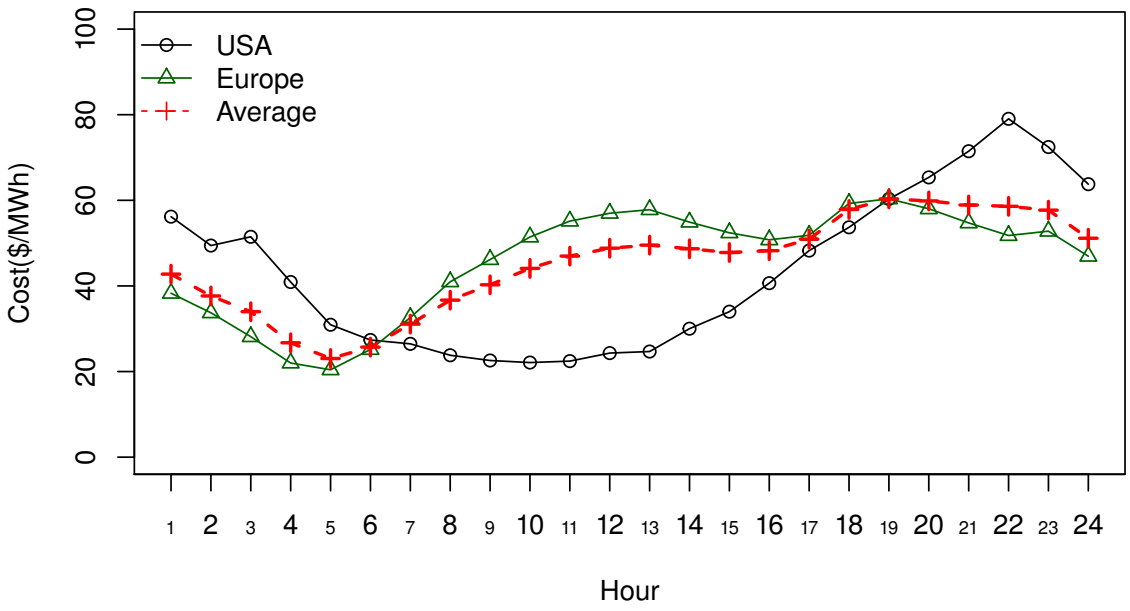


Figure 3.2: Time shifted, hourly electricity prices

be in one data center location and the low service rate machines are assumed to be in the second data center.

The scheduling algorithm is executed at a load balancing front-end - which could be either be built using off the shelf hardware and software components, or a vendor supported commercially available device (with or without customised hardware and software components). Commercially available load balancers support local load balancing - where requests are assigned to servers within a server cluster - and global load balancing - where requests are load balanced across data centers.

3.4 Hourly Electricity Prices

We obtain per-hour electricity prices from The USA [1] and The Netherlands [2] for an entire week. Analysing the data, we observed that across the seven days, the electricity price for the same hour of the day can show substantial fluctuation. Therefore, we averaged the hourly prices for the seven days for our evaluation.

Figure 3.2 shows the hourly variation in electricity prices, measured in Dollars per Mega Watt hour ($/MWh). The prices are corrected for timezone differences. The line representing Europe (marked with 'o') shows the variation in Europe region from 00:00 to 23:59 Central European Summer Time (CEST) for a Wednesday. Prices from USA Eastern Standard Time (EST) (marked with '△') are timezone corrected to use 6 hours from Tuesday evening (18:00 to 23:59) and 18 hours of Wednesday(00:00 to 17:59; the line marked with '+' plots the average price. We can see that prices in Europe are higher for 13 hours and the prices in the USA are higher for the remaining 11 hours in a span of 24 hours. For the experiments that are designed to be corner-case evaluations, we do not use the diurnal electricity price variations, but use these values to obtain prices for these evaluation scenarios. For example, in the low capacity-low price data center evaluation scenario, we use the lower electricity prices from the available data instead of assuming random electricity prices.

3.4.1 PUE Values

PUE values of individual data center are difficult to obtain as data center owners do not appear to make such information public, however Google has provided PUE information per quarter [22] and also over the duration of one day for one of their data centers. PUE seen over one day of operation of a data center is not a static but can vary over the course of the day, similar to electricity prices. The PUE values are available for a 24 hour period, and were suitably time-shifted as mentioned in the previous section.

3.5 Communication Costs

In order to measure the communication costs, we need the cost of bandwidth from the source regions to the destination regions. The bandwidth consumption depends on amount of data transferred on the networking link - which can be determined by the bytes transferred for each request-response pair and the number of requests. We assume a page size of 3KB and requests are 128 bytes, giving us a total of 3200 bytes, which is also the mean size of articles in Wikipedia. Bandwidth prices are obtained from [4].

3.6 Experiments & evaluation metrics

We evaluate the scheduling algorithms with the inputs - electricity price, PUE, workload trace - described in the previous sections and compare them on cost incurred and response time. The Even Distribution scheduling described earlier is a naïve scheduling algorithm and provides us with a baseline to compare the rest of the algorithms against. Being a naive algorithm, this approach does not leverage either the price of electricity or consider any of the constraints that we set out in the problem definition.

The Least Price scheduling provides us with a greedy algorithm that takes advantage the lower electricity prices to provide reduction in the total cost. However, being a greedy algorithm, this approach does not consider any response time or bandwidth constraints that are set. Being greedy, the least price scheduling will inherently scheduling requests to the the data center where the electricity price is low, possibly resulting in the data center operating at full capacity.

The Global Cost Diversity Aware scheduling algorithm leverages the electricity price and data center efficiency while considering the networking bandwidth, server capacity, and processing time constraints. Consequently, we expect this algorithm to provide the least cost, while respecting all of the constraints placed. Most importantly, we do not expect to observe any violations of the response time constraints set with the GCDA algorithm.

Between the local scheduling algorithms, the Proportional Fair algorithm assigns requests across all server types. Consequently, it assigns requests to the servers that exhibit lower power consumption and can take advantage of the same. The Fastest Server First scheduling assigns requests to the servers in the order of service rate, therefore, it assigns requests to the low power servers only when the high power servers are executing at full capacity. Consequently, we expect that this scheduling algorithm exhibits higher cost compared to the Proportional Fair scheduling.

We compare the scheduling algorithms on two aspects, cost and response time as discussed below.

- Cost: The objective of our algorithm is cost reduction, therefore, we compare the algorithms on the total cost incurred for the completion of the workload.

- Application Response Time: The performance aspect to measure is the response time exhibited by the requests. More importantly, it is necessary to measure the percentage of requests that complete within the response time specified.

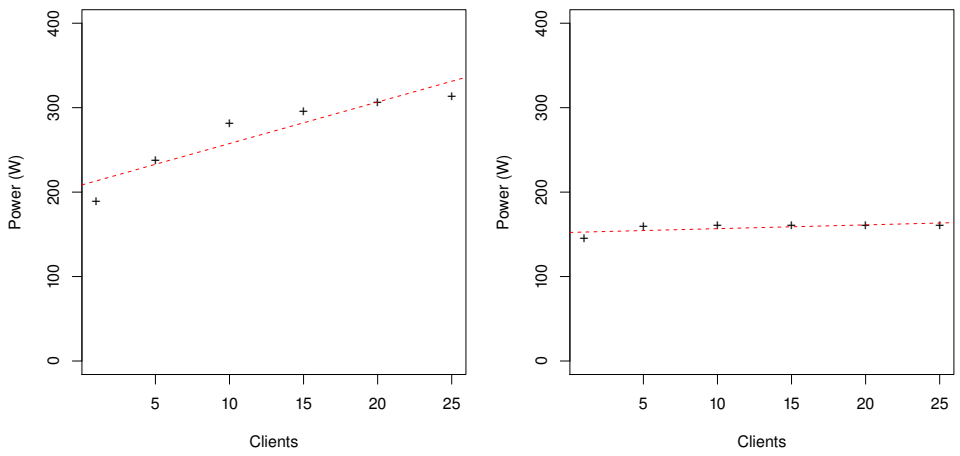
Chapter 4

Results

In this chapter we provide the results of our experiments. We first provide the results from measuring the power and performance of individual servers, which we subsequently use in the GCDA scheduling algorithm to estimate the power consumption of servers as the load, and consequently the utilisation varies. Next, we present the results of the experiments where we execute the scheduling algorithms presented earlier and compare the algorithms on cost incurred and the response time behaviour. We mentioned earlier that while on the one hand a scheduling algorithm could result in lower power consumption and consequently costs, it could result in response time constraint violations. We execute the GCDA algorithm with increasing response time constraints and also a special case, where no response time constraint is placed.

4.1 Power Models

We first discuss the power and performance models that we obtained by executing an HTTP workload using the Mediawiki application with an increasing number of clients. The plots in Figure 4.1 captures the power consumption of an two different servers: the first (Figure 4.1a), is a 12-core, AMD® Opteron™ processor based server and the second, a 4-core, Intel® Xeon™ processor (Figure 4.1b). The 12-core AMD Opteron server exhibits a higher power consumption compared to the 4-core Intel Xeon server – its idle power consumption is higher than the active power consumption of the Xeon server – because it is



(a) Power Consumption (12 Core AMD Opteron)

(b) Power Consumption (4 Core Intel Xeon)

Figure 4.1: Power consumption patterns of two servers (running the Mediawiki application) with an increasing number of client connections

comparatively more complex. Besides having more cores (12 versus 4), it also supports non-uniform memory access (NUMA), has multiple memory controllers etc, which result in the increased system power consumption.

In the plots, the abcissa axis captures the number of concurrent virtual clients, and the ordinate axis captures the power consumption, measured in Watts. An increase in the number of clients results in increased load on the system under test (SUT). As a result, the utilisation and consequently the power consumption of the SUT also increase as shown in figures 4.1a and 4.1b. The plots are limited to thirty concurrent clients, because at higher virtual client counts, client errors were observed. When the utilisation reaches 100%, the power consumption also peaks, and an increase in the number of virtual clients does not result in an increase either in utilisation or power consumption.

The regression line (the dashed line in the plot), and the regression coefficient show a strong linear correlation between the request rate and power consumption. The power consumption plots shown here are consistent with the prior results which have undertaken similar measurements [29, 44, 14].

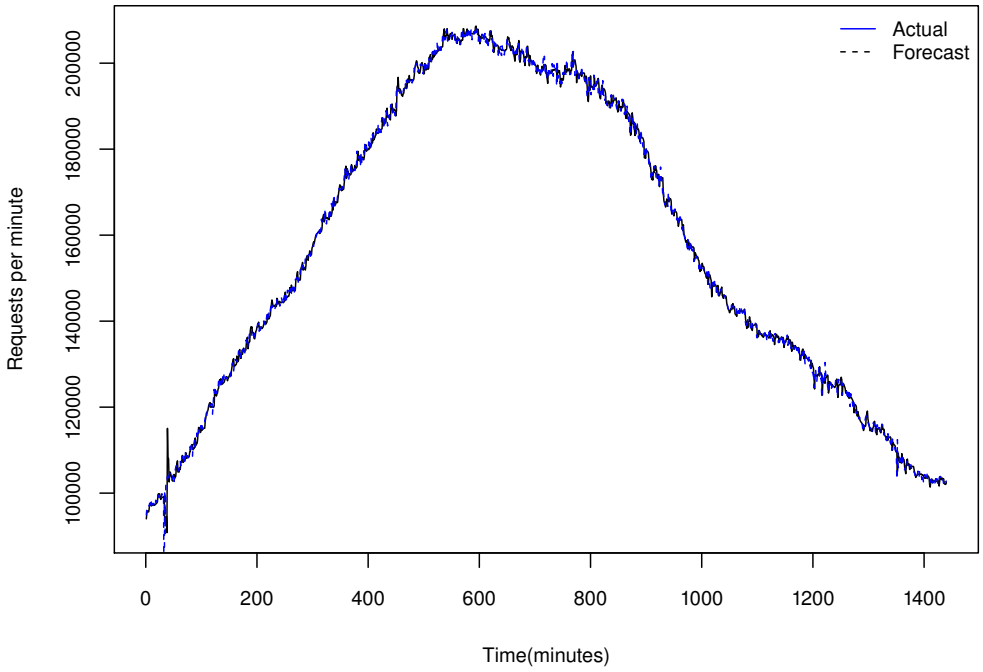


Figure 4.2: The actual workload trace pattern and predicted request rates

4.2 Workload

In Section 2.4.8, we discussed the time series analysis and forecasting used in our solution. Prior to providing the input to the time series modelling, we increase the number of requests serviced by 1%. We do not make any predictions for the first five minutes [40], and use these observations for making predictions starting from the 6^{th} minute. In Figure 3.1, we showed the actual pattern of the workload over time; Figure 4.2 captures the request pattern seen on Wikipedia for one day. The x axis captures the time in minutes and the y axis captures the request rate measured in requests per minute. The dashed line captures the actual values obtained from the trace, and the unbroken line captures the predicted values. Despite correcting the input provided to the time series model by 1% as explained above, the two lines overlap closely showing that the predictions obtained from the time series analysis closely matches the actual values.

4.3 High Capacity Servers

We evaluate the algorithms with two different server setups: In the first case, we assume heterogeneous servers, all of which provide high service rate, whose power and performance profiles are comparable and obtained from measurements made on 64 bit, x86 instruction set architecture, multi-core processor based servers (Section 4.1). Such a setup is similar to typical data centers; for example, Wikipedia uses servers with high frequency processors and high RAM capacity in its hosting facilities [5]. The second setup, discussed in detail in the subsequent section, consists of a servers that differ substantially in their power consumption and service rate.

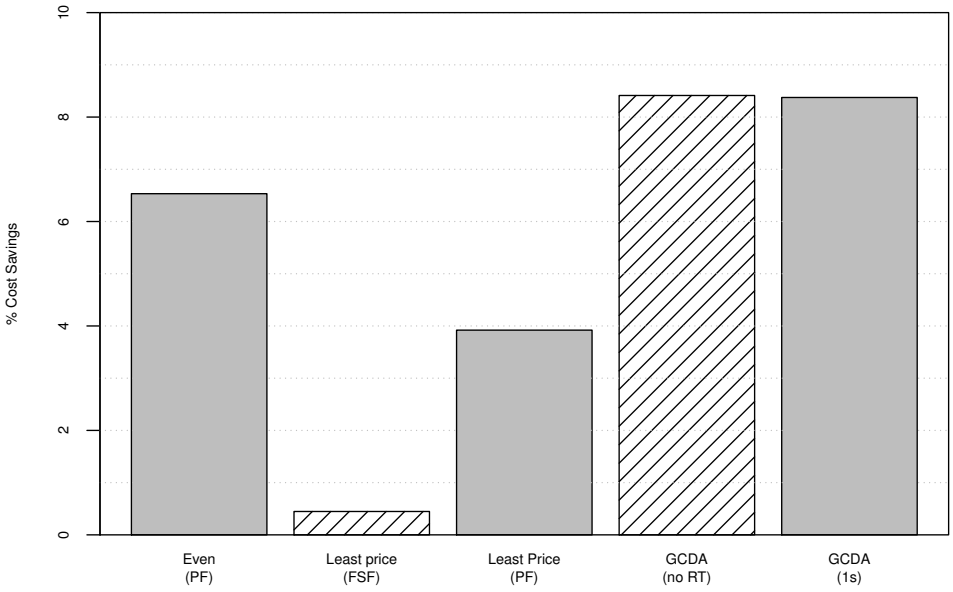
4.3.1 Costs and Power Consumption

In this section, we discuss the costs and power consumption resulting from the algorithms. Figure 4.3a shows the cost incurred across the various scheduling algorithms. The cost incurred under each scheduling algorithm is normalised to the naïve Even Distribution scheduling algorithm, with the Fastest Server First local scheduling algorithm. The y axis captures the cost savings (shown as a percentage), and the scheduling algorithms are presented along the x axis. Recollect that the Fastest Server First local scheduling assigns requests to servers in the descending order of the service rates, whereas the Proportional Fair local scheduling assigns requests to servers in the ratio of their service rates. The power consumed, and the resultant electricity cost discussed below do not take into account the power consumed by the load balancing front end itself.

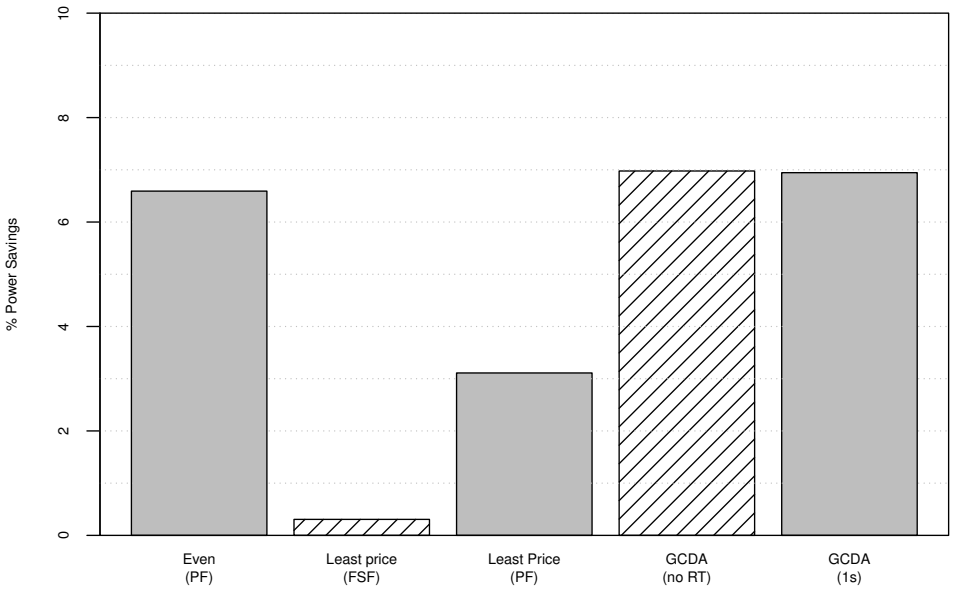
The greedy Least Price scheduling algorithm exhibits a marginal reduction in the cost compared to the Even Distribution. Between the two local scheduling algorithms, the Proportional Fair algorithm exhibits lesser electricity price because requests are spread across all server types and lesser power is consumed to service requests assigned to the low-power servers. The GCDA algorithm shows 8.5% reduction in costs compared to the Even Distribution algorithm; it also exhibits 7% lower power consumption compared to the Even Distribution algorithm, as seen in Figure 4.3b.

The distribution of requests across server types, shown in Figure 4.4, provides us some

insight on the power consumption. In the plot, the y axis captures the percentage of requests, while the scheduling algorithms are listed along the x axis. For each scheduling algorithm, the servers are listed in the descending order of their service rates. In the Fastest Server First local scheduling scenario, requests are first assigned to the servers that provide higher service rate. However, these servers also consume higher power, resulting in higher aggregate power consumption. The Least Electricity Price algorithm assigns requests to data centers in the increasing order of electricity price and assigns requests to a data center till is processing capacity is reached before assigning requests to the next data center. Because the data center is assumed to contain servers of all types, requests are assigned to all server types. Therefore, the requests distribution with the Least Electricity Price scheduling shows that the servers that provide lesser service rate execute more requests compared to the Even Distribution scheduling. With the GCDA algorithm, a majority of requests (up to 80%) are serviced by the servers that consume lesser power, providing us with reduced power consumption.



(a) Cost Savings



(b) Power Savings

Figure 4.3: Cost and power savings seen with different scheduling algorithms, when using a server setup consisting of high capacity servers.

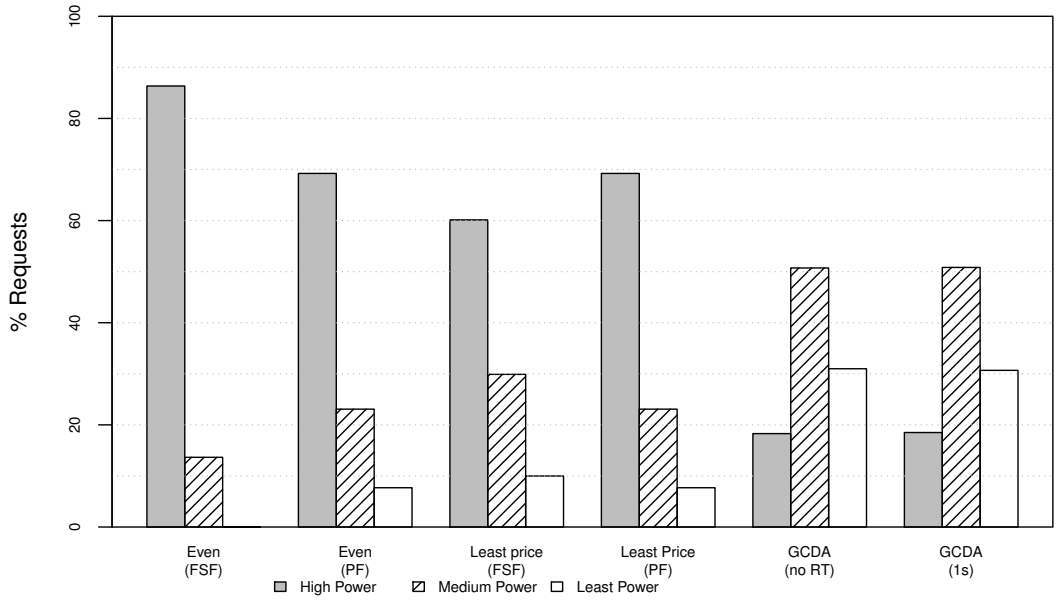


Figure 4.4: Distribution of requests to servers. Each bar captures the percentage of requests executed by a the particular type of server. Servers are listed in descending order of maximum power consumption.

4.3.2 Response Time

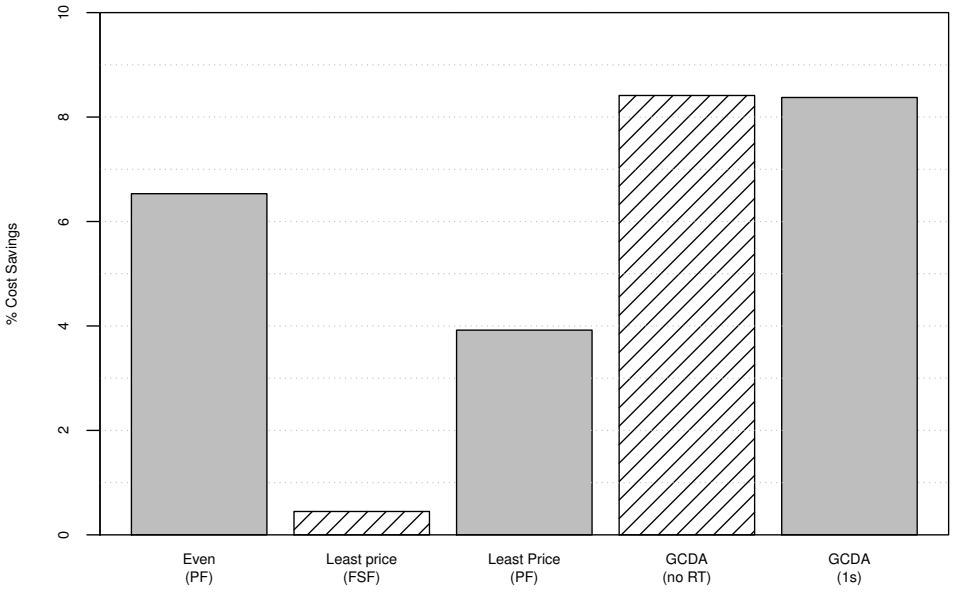
Response time exhibited by the algorithms provides us with a metric of performance. Because the load on each server is expected to vary over time due to the nature of the workload that we use, we expect to observe requests experiencing different response times. However, we are particularly interested in the number of requests that violate the response time bound that we set. Table 4.1 captures the distribution of response times. We bin the response time experienced for each request into '1 second' bins, except for the last bin which captures response times of 5 seconds and higher. Each bin captures the percentage of requests that fall under it. We observe that with the Even Distribution and the Least Price, the response time values are spread over the various bins and in some cases the response times in excess of 5 seconds are observed. With the GCDA algorithm however, we do not see any response time violations when the constraint is set. When we do not set any response time constraint, the service rate constraint on each server must be respected, that is the request rate on a given server can be equal to one less than its service rate.

Algorithm	1s	2s	3s	4s	5s
Even Distribution (FSF)	48.7	0.0	0.0	0.0	51.2
Even Distribution (PF)	100.0	0.0	0.0	0.0	0.0
Least Price (FSF)	20.1	0.1	0.0	0.0	79.7
Least Price (PF)	21.3	19.6	19.67	19.67	19.67
GCDA (no constraint)	0.4	0.0	0.0	0.0	99.5
GCDA (1s)	100	0.0	0.0	0.0	0.0

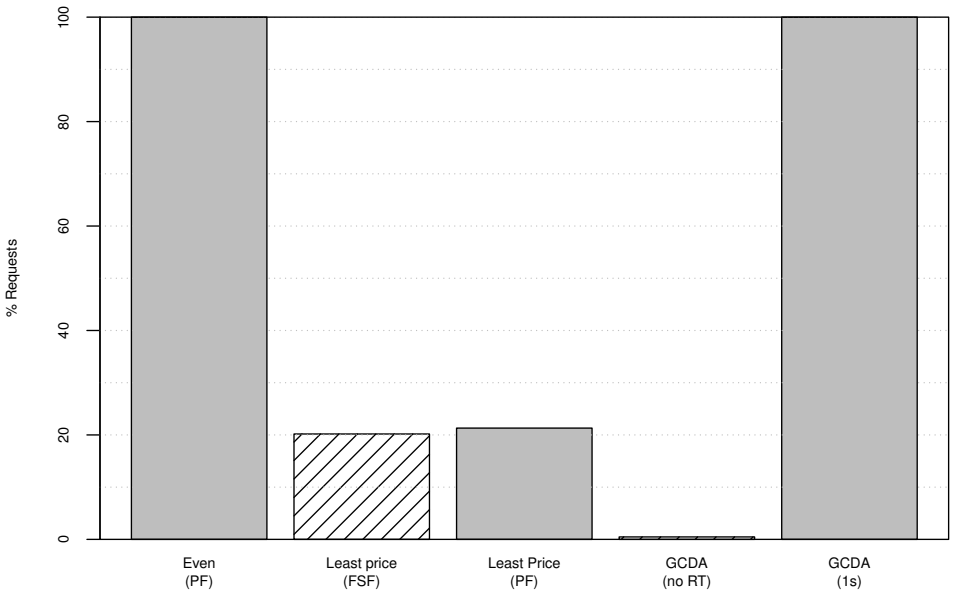
Table 4.1: Table capturing the number of requests under each 1-second response time bin.

The Fastest Server First algorithm assigns requests to the servers that provide higher service rate (fast servers) before assigning requests to the servers with lower service rate. The downside of this algorithm is that at high request rates, such a schedule results in the faster servers executing at near or full service rate, therefore, the requests spend longer duration waiting to be serviced. The Proportional Fair local scheduling algorithm however, spreads the requests across all the server types, in proportion to the service rate. Therefore, instead of resulting in high load on a small number of servers, we see low load on comparatively more servers, and it results in fewer violations.

Figure 4.5 provides us with a combined picture of cost savings and response time behaviour - the first plot captures the cost savings, while the second plot captures the percentage of requests that complete within the time specified- we use a response time of 1 second. With the GCDA algorithm and the Even Distribution algorithm (with the Proportional Fair local scheduling) all requests complete within the response time specified; except in the case when no constraints are placed on the GCDA algorithm, where all requests experience a response time of 5 seconds or higher.



(a) Cost



(b) Response Time

Figure 4.5: Cost savings and percentage of requests that do not violate response time constraints for each scheduling algorithm.

4.4 Mix of low & high performance servers

Chun et al [15] showed that hybrid data centers - data centers containing a mix of high and low performance servers have the potential to provide energy efficiency without substantial loss of performance. Taking this result into consideration, we evaluate our algorithm with a data center setup that consists of high service rate and low service rate servers. However, because we do not have low power computing platforms at our disposal for the purpose of characterisation, we reuse the power-performance measurements provided by Krioukov et al [35].

Servers that provide high service rate typically exhibit high power consumption and those that consume low power provide low service rate. This data center configuration is significantly different from our previous setup, where, though the servers were heterogeneous, the differences in the power consumption and service rates were less pronounced.

4.4.1 Cost Savings

Figure 4.6 captures the cost and power savings observed with the second data center configuration. In these plots, the cost and power savings are normalised to the Even Distribution scheduling algorithm with Fastest Server First local scheduling. The ordinate axis captures the percentage cost savings; the scheduling algorithms are shown along the abcissa. The Even Distribution scheduling algorithm again results in the highest cost while the greedy Least Price scheduling algorithm exhibits marginally higher cost savings. Using the GCDA algorithm, however, we obtain between 14.7% to 15% cost savings compared to the Even Distribution.

Power consumption is shown in Figure 4.6b and here too, the Proportional Fair local scheduling algorithm results in lesser power consumption and cost compared to the Fastest Scheduling First local scheduling algorithm – savings obtained by scheduling to servers that consume lesser power. The GCDA algorithm also provides from 12.2% to 12.4% power savings compared to the Even Distribution scheduling algorithm.

Because the configuration consists of servers that provide low service rate, the total number of requests serviced by such servers significantly lesser than that serviced by the high service rate servers. In the data center setup discussed in the previous section, such

Algorithm	1s	2s	3s	4s	5s
Even Distribution (FSF)	96.6	0.8	0.1	0.0	2.0
Even Distribution (PF)	100.0	0.0	0.0	0.0	0.0
Least Price (FSF)	22.5	0.1	0.0	0.0	77.2
Least Price (PF)	20.0	19.98	19.98	19.98	19.99
GCDA (no constraint)	0.5	0.0	0.0	0.0	99.5
GCDA (1s)	100	0.0	0.0	0.0	0.0

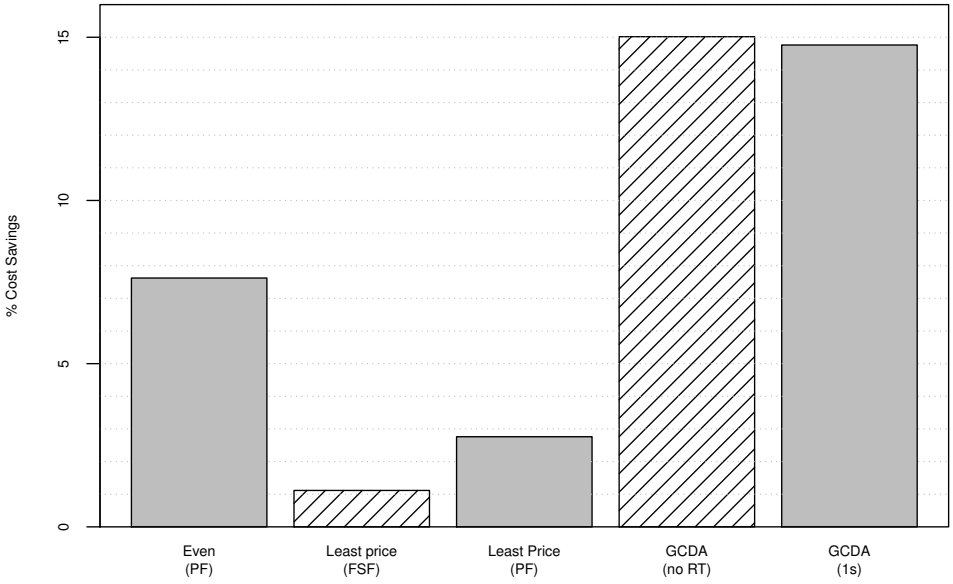
Table 4.2: Table capturing the percentage of requests falling within each response time bin.

was not the case because the servers were comparable in their service rates.

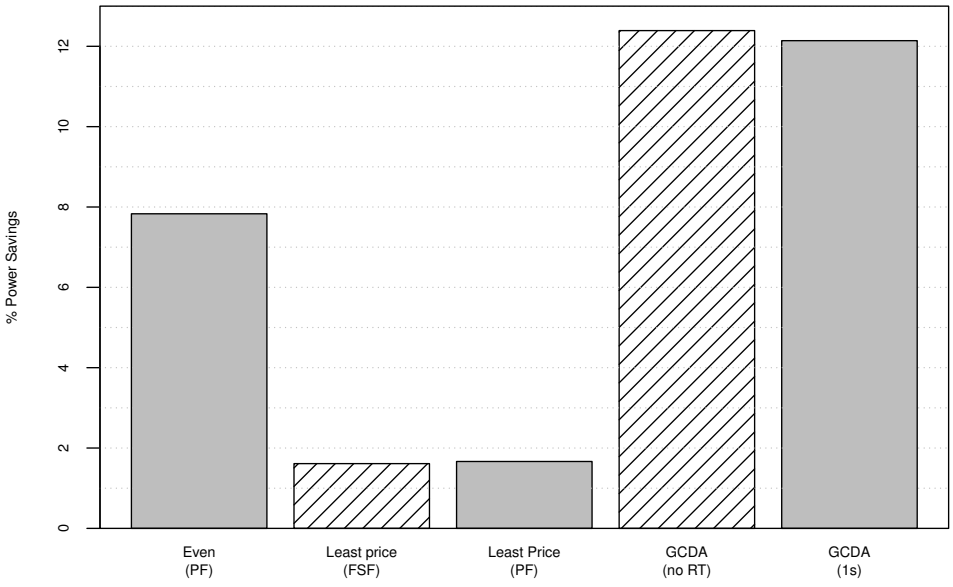
4.4.2 Response Time

Table 4.2 captures the distribution (similar to Table 4.1) where the response time is binned into 1 second intervals, and each bin captures the percentage of requests in the bin. In this configuration too, we used 1 second as the target response time. From the table, we observe that the Even Distribution and the Least Price scheduling algorithms show a range of response times and also result in violations - between 20% and 22% requests are serviced within 1 second with the Least Electricity Price scheduling algorithm. The GCDA algorithm performs consistently well and no performance violations are observed.

Figure 4.8 captures both the cost savings and the percentage of requests that were serviced within the response time specified and provides a unified picture of the performance of the various algorithms.



(a) Cost Savings



(b) Power Consumption

Figure 4.6: Cost and power savings with a server setup consisting of high and low performance servers. High performance servers also consume higher power compared to low performance servers.

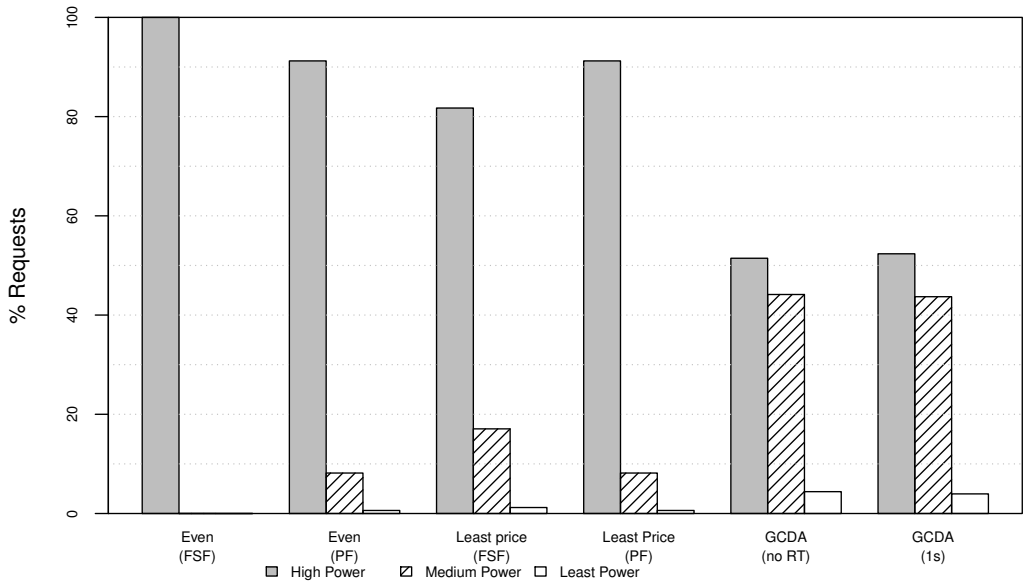
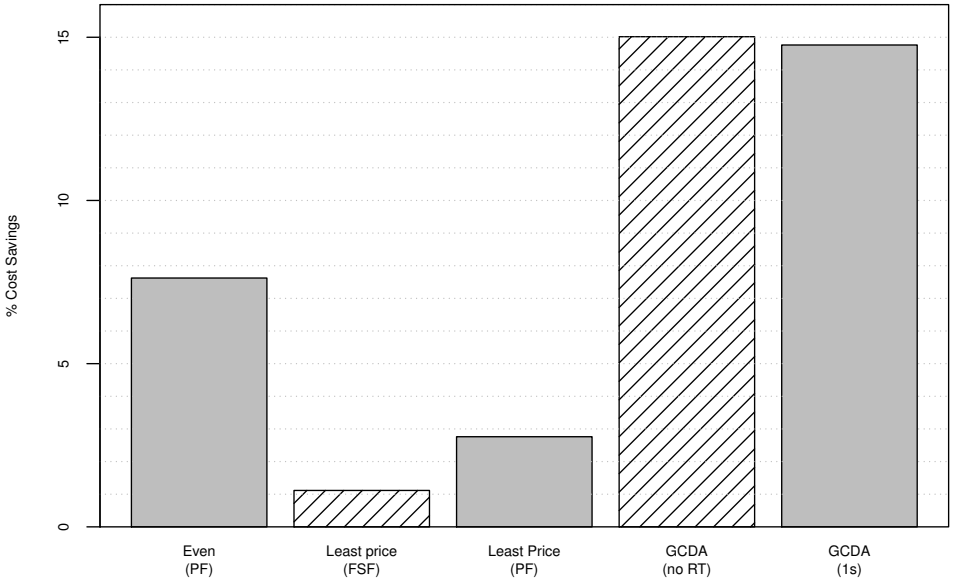
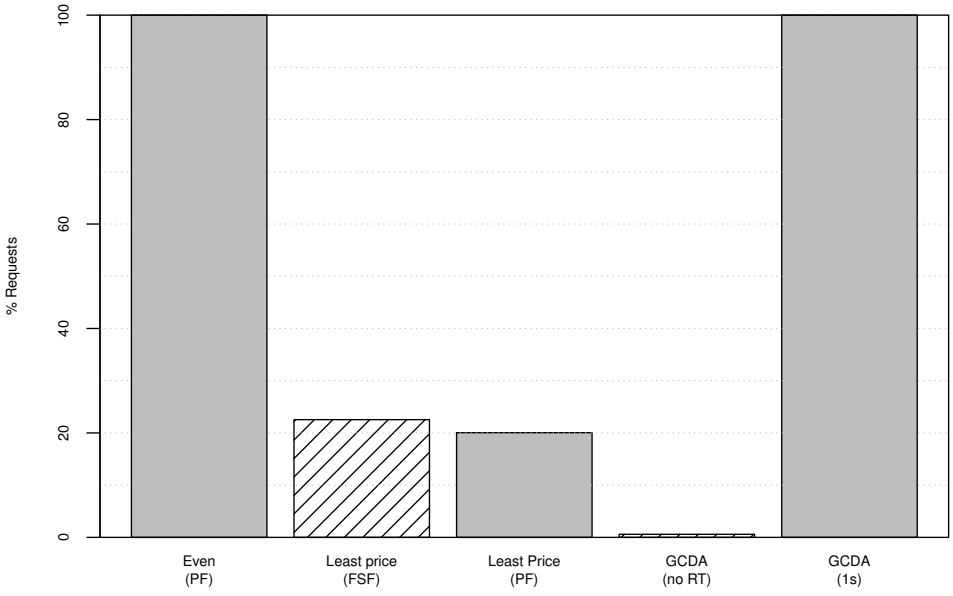


Figure 4.7: Distribution of requests to each server type. Servers are listed in the descending order of service rate.



(a) Cost Savings



(b) Response Time

Figure 4.8: Cost savings and percentage of requests not violating response time constraint for each scheduling algorithm.

4.5 Behaviour of the GCDA Algorithm

We observe from the results presented in the previous sections that the GCDA algorithm provides cost savings irrespective of the data center configuration. More importantly, the cost savings are made without loss of quality of service, because we do not violate the response time constraints. The cost savings however is higher (closer to 15%) when the data center configuration includes a mix of high performance and low performance servers; the cost savings is close to 9% with a servers that provide comparable power and performance.

An observation is that the costs do not differ significantly with the different configurations of GCDA where we vary the response time constraint. We do however observe that the savings obtained when no response time constraint is specified is the highest. The savings are least with the 1 second response time and improves as the constraint is relaxed. This reduction in savings between the cases when response time bounds are specified can be explained by investigating the change in the request distribution as we relax the response time constraint. The tightest response time constraint we placed on the system was 500 milli-second and relaxed to a 'no response time' constraint. When we relax the response time constraint, more requests are scheduled on the lower service rate servers, which also consume lesser power, consequently reducing the total power consumption and the costs incurred. When we do not set any response time constraint on the system, we could expect all requests to be assigned to the servers that consume less power. However, we also have a request rate constraint which ensures that no server is assigned more requests than its service rate, because such a circumstance will result in client connection errors or timeouts.

Another observation on investigating the request distribution is the distribution of requests to server types across the scheduling algorithms. With the Even Distribution and Least Electricity Price scheduling algorithms, the servers that provide high service rate execute a significant percentage of the requests. However, with the GCDA algorithm, the servers that provide lesser service rate (and also consume less power) service comparitively higher number of requests. For example, in Figure 4.4 (which captures the request distribution with servers providing comparable service rate), the slower servers execute most of the requests with GCDA (close to 90%), while with the other scheduling algorithms, the slower

servers service between 10% to 40% of the requests. Using the slower servers to service requests significantly reduces the total power consumption. With the second setup, the faster server executes a higher percentage of requests irrespective of the algorithm used. However, with the GCDA algorithm, the slower servers handle a much higher percentage of requests compared to the other algorithms. For example, with the Even Distribution-Fastest Server First setup, the fast servers handle all requests. Consequently, the power consumption and the electricity costs are lesser with the GCDA algorithm, resulting in savings.

Table 4.3 and Table 4.4 captures the response time distribution seen with varying configurations of the GCDA algorithm. In each case, we bin the response time exhibited into 1 second intervals, and each bin captures the percentage of requests within the bin. Table 4.3 captures the response time distribution with the high capacity server configuration where the servers have comparable power performance profiles, while Table 4.4 captures the distribution for the configuration consisting a mix of high and low performance servers.

When no response time constraint is specified, most requests ($> 99\%$) exhibited a response time of 5 seconds or higher, and when a response time constraint is specified, all requests complete within the time specified. When the response time is relaxed requests take more time to complete; for example, when the response time constraint is relaxed from 1 to 2 seconds, 99.5% of requests which were serviced within 1 second now get serviced under 2 seconds. This behaviour is intuitively expected: when the response time constraint is relaxed, more requests can be assigned to the servers that provide low service rate.

Algorithm	1s	2s	3s	4s	5s
GCDA (no constraint)	.4	0.0	0.0	0.0	99.5
GCDA (500ms)	100.0	0.0	0.0	0.0	0.0
GCDA (1s)	100.0	0.0	0.0	0.0	0.0
GCDA (2s)	0.4	99.5	0.0	0.0	0.0
GCDA (3s)	0.4	0.0	99.5	0.0	0.0
GCDA (4s)	0.4	0.0	0.0	99.5	0.0

Table 4.3: Table capturing the number of requests under each 1 second response time bin. This table corresponds to the data center setup consisting of servers with comparable power-performance profiles.

Algorithm	1s	2s	3s	4s	5s
GCDA (no constraint)	.5	0.0	0.0	0.0	99.5
GCDA (500 ms)	100.0	0.0	0.0	0.0	0.0
GCDA (1s)	100.0	0.0	0.0	0.0	0.0
GCDA (2s)	0.5	99.46	0.0	0.0	0.0
GCDA (3s)	0.6	0.0	99.4	0.0	0.0
GCDA (4s)	0.5	0.0	0.0	99.4	0.0

Table 4.4: Table capturing the percentage of requests falling within each response time bin. This table corresponds to the data center setup consisting of high and low performance servers.

4.6 Hypothetical Scenario

In the results discussed thus far, we used the power and performance profiles of servers obtained by measurement. In this section, we discuss the results obtained when we retain the power profiles of the servers, but assume that the servers provide equal service rate. This is a hypothetical scenario where, due to our assumption, the low power servers provide high service rate. We conduct our experiments with both the data center configurations discussed earlier and compare the cost and response time behaviour of the algorithms. The results from our previous experiments indicate that the cost savings are higher when the server setup consists of servers that consume low power. The cost savings observed with the equal service rate scenario are in line with the previous results as shown in the figures below. The savings obtained with the high service rate data center setup is 15% and 35% with both high and low service rate servers.

With the capacities being equal, the servers that consume lesser power for the computation service more requests than the high power consuming servers. The server that consumes least power services close to 60% requests as shown in figures 4.11 and 4.12. The low and medium power server together account for between 96% to 97% requests while the remaining 3% to 4% requests are serviced by the servers that consume high power.

This request distribution - where most requests are serviced by the servers with low power consumption is expected. This hypothetical setup removes the heterogeneity aspect - the performance heterogeneity is not applicable because all servers provide equal service rates and leads us to a simple, alternative scheduling algorithm where requests are greedily

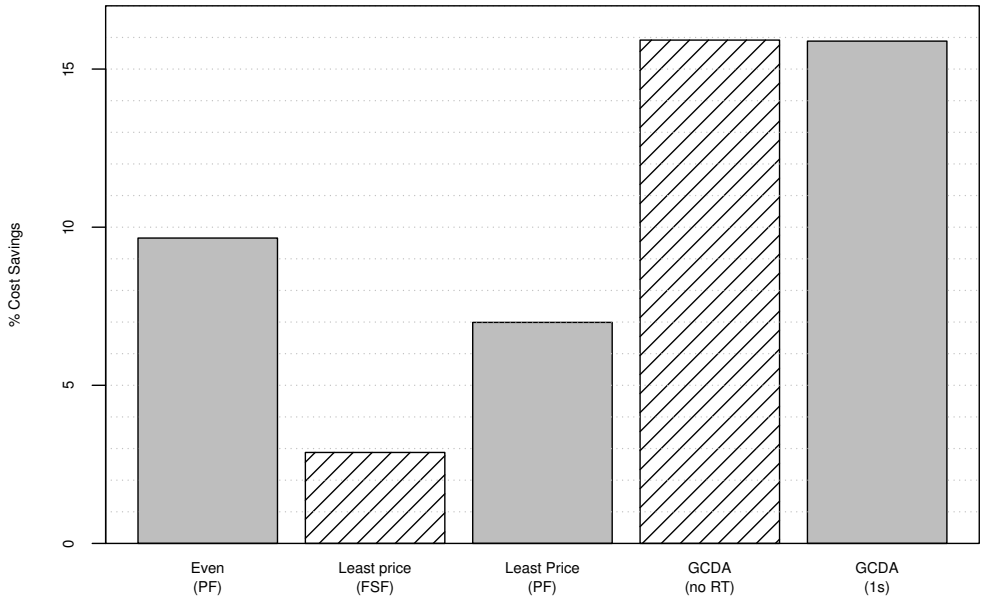


Figure 4.9: Cost savings with servers exhibiting comparable power consumption and assumed to have uniform service rate.

assigned to servers in the ascending order of server power consumption, that is, requests are first assigned - across both data centers - beginning with the servers that consume least power.

Such a schedule exhibits the highest power savings as shown in Figure 4.13 and Figure 4.14. These plots are similar to the power savings plots presented earlier, but also show the savings resulting from the greedy algorithm which schedules requests based on power consumption. The cost savings, in excess of 30% under both data center configurations, however comes at the price of poor response time: from 78.8% to 79.1% requests suffer a response time of 5 seconds or higher.

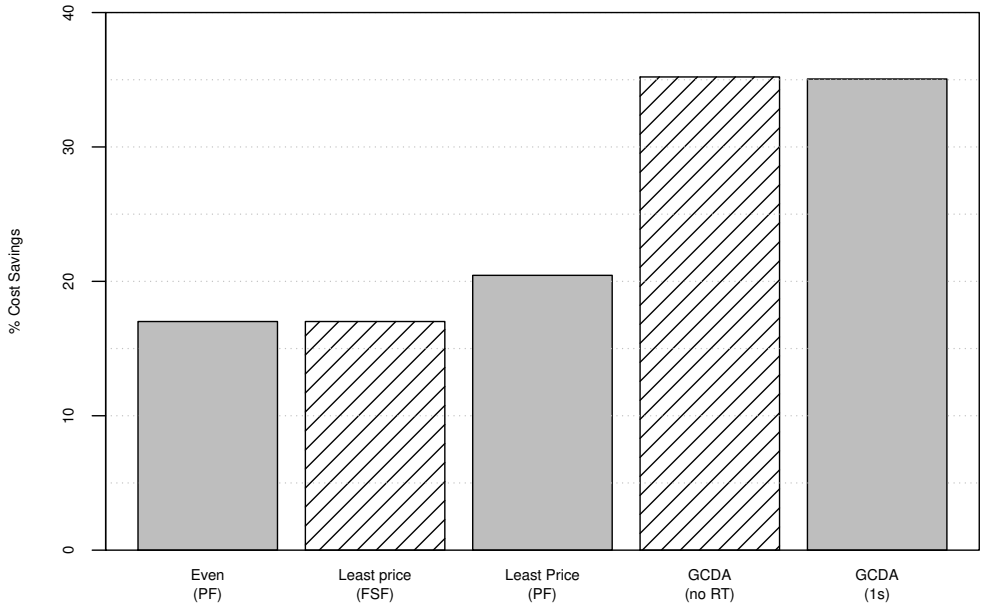


Figure 4.10: Cost savings with servers exhibiting widely differing power consumption but assumed to have uniform service rate.

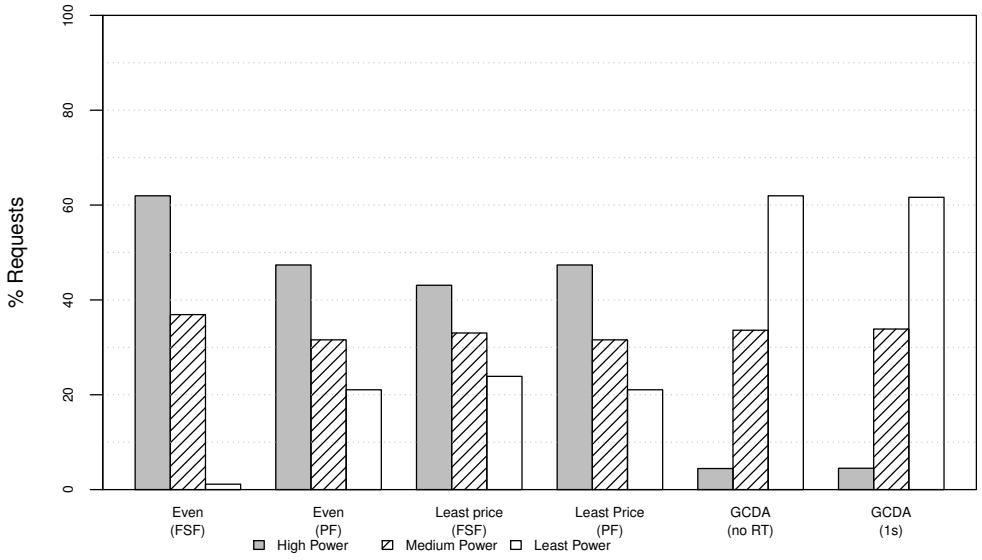


Figure 4.11: Distribution of requests to servers, with servers listed in the descending order of their power consumption. This plot corresponds to the data center setup with servers exhibiting comparable power profiles and assumed to have equal service rate.

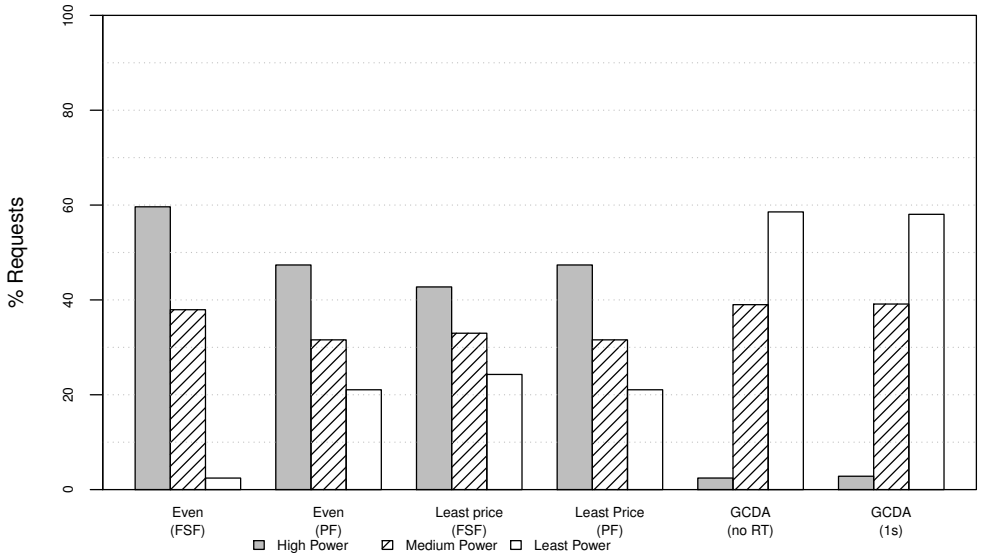
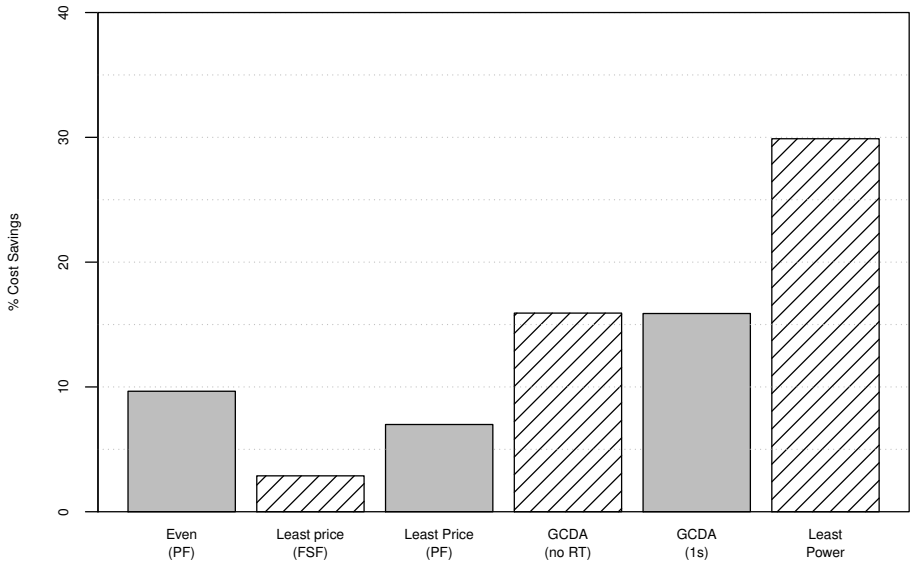
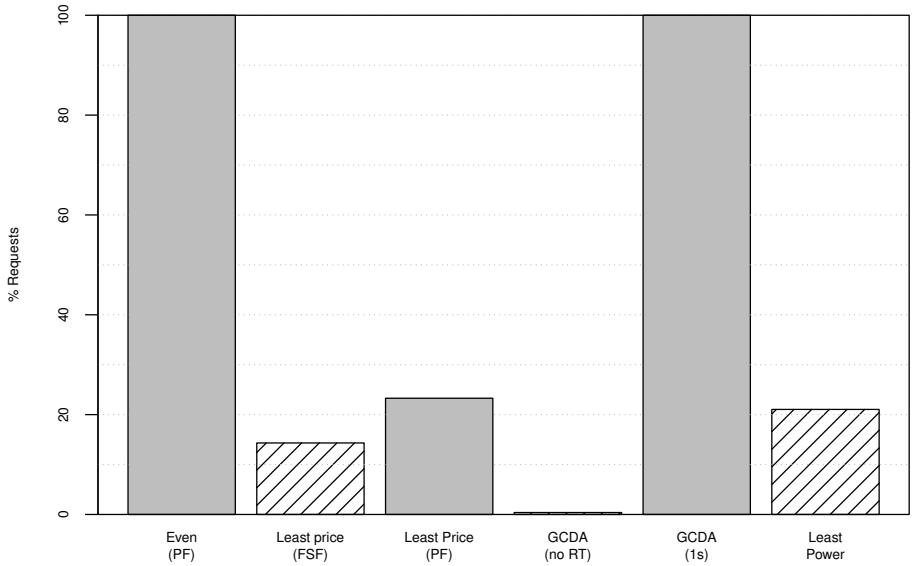


Figure 4.12: Distribution of requests to servers, with servers listed in the descending order of their power consumption. This plot corresponds to the data center setup with high and low power consumption servers; servers are assumed to have equal service rate.

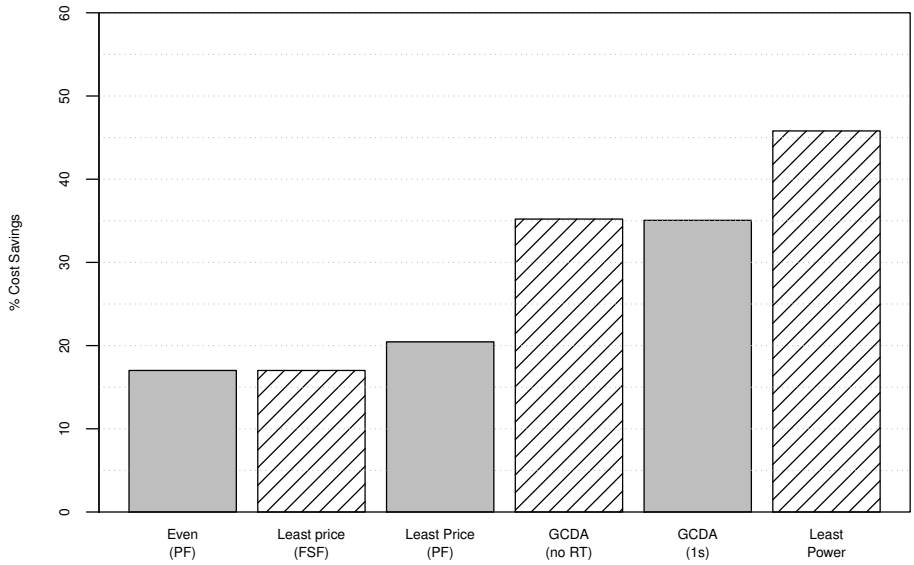


(a) Cost Savings

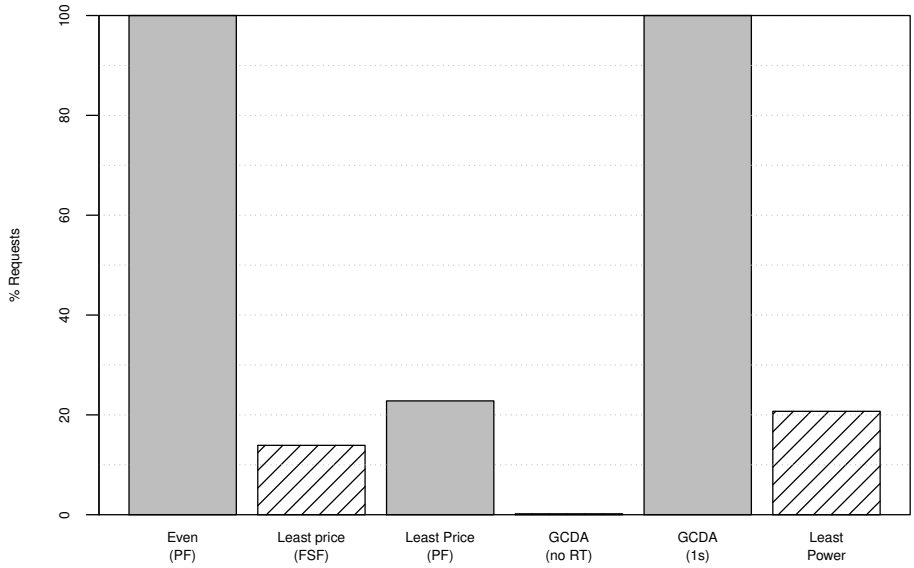


(b) Response Time

Figure 4.13: Cost savings and response time with server setup consisting of servers with comparable power consumption and compared to the greedy Least Power scheduling algorithm.



(a) Cost Savings



(b) Response Time

Figure 4.14: Cost savings for server setup consisting of high & low power servers and compared to greedy, Least Power scheduling algorithm.

4.7 High Service Rate - Low Price Data Centers

In the previous sections, we analysed the Global Cost Diversity Aware algorithm while varying server properties. Another evaluation scenario is to the case where one data center consists of servers that provide high service rate while the others consist of servers with low service rate. Further, the data center that consists of high service rate servers also exhibits the least electricity price. In this scenario, attempting to use the high service rate data center, which also exhibits lower price, would result in higher power consumption.

Comparing the total costs in this scenario, we observe that the GCDA algorithm provides 1% cost savings with first server setup where we had servers with comparable service rates and 3% savings with a mix of high and low performance servers - compared to the Even Distribution algorithm - which is lesser compared to our earlier evaluation scenarios. The response time distribution for both the server setups we have used so far is shown below (Tables 4.5 and 4.6). Here too, the GCDA algorithm does not result in any violations of response time constraints.

Algorithm	1s	2s	3s	4s	5s
Even Distribution (FSF)	68.3	0.2	0.0	0.0	31.3
Even Distribution (PF)	22.7	19.3	19.3	19.3	19.3
Least Price (FSF)	84.4	0.7	0.1	0.0	14.6
Least Price (PF)	20.0	19.98	19.98	19.98	19.99
GCDA (no constraint)	0.5	0.0	0.0	0.0	99.4
GCDA (1s)	100.0	0.0	0.0	0.0	0.0

Table 4.5: Distribution of response time with the server setup consisting of servers with high performance servers, where one data center consists of the highest capacity servers and also has the least electricity price.

4.8 Chapter Summary

In this chapter, we presented the results of using the Global Cost Diversity Aware scheduling algorithm and showed that the algorithm provides cost savings compared to both the naïve Even Distribution, and a greedy Least Electricity Price scheduling algorithm. We

Algorithm	1s	2s	3s	4s	5s
Even Distribution (FSF)	84.3	0.1	0.1	0.0	15.3
Even Distribution (PF)	100.0	0.0	0.0	0.0	0.0
Least Price (FSF)	43.3	0.3	0.1	0.0	56.1
Least Price (PF)	19.8	20.04	20.05	20.05	20.05
GCDA (no constraint)	0.4	0.0	0.0	0.0	99.5
GCDA (1s)	100.0	0.0	0.0	0.0	0.0

Table 4.6: Distribution of response time with the server setup consisting of high and low capacity servers, where one data center consists of the highest capacity servers and also has the least electricity price.

evaluated our algorithm with a variety of data center configurations and observed that the Global Cost Diversity Aware algorithm provided close to 15% savings in cost; more importantly, we did not observe any response time violations in any evaluation scenario. Our results also indicate that cost savings are higher when the data center consists of a mix of high and low performance servers.

Chapter 5

Related Work

5.1 Introduction

A data center can be viewed as a facility that houses computing components and non-computing components. The computing components are the server clusters which perform computational tasks and service requests. Non-computing components include Computer Room Air Conditioning (CRAC) units. Power consumption of the data center is the aggregate power consumed by all these components. Any computation would require power at the processor, for input/output operations, and for removing the heat generated during the computation. The cooling efficiency of a data center is the amount power required to remove the heat generated due to one watt of power consumed by the servers.

5.2 Power management for server clusters

There is a substantial body of current work that focuses on power management for individual machines [18, 46], homogeneous and heterogeneous server clusters, and cooling; the power savings can then be translated into cost savings. Reducing power consumption has focused on power managing servers using processor dynamic voltage and frequency scaling (DVFS) [48, 40], placing servers in sleep states [18] or a combination of both [12, 21, 16, 17, 29]. Processor power consumption is directly proportional to the frequency

of the processor and by varying the processor voltage and frequency based on its utilisation, the power consumption of the processor can be reduced [23]. Additional power savings can be obtained by placing the entire server in a sleep state - for example, suspending the processor to disk (hibernate mode) results in the system state being written to disk, and all components powered off. This state clearly consumes lesser power than a server that is powered on.

We discuss the related work in the area of power management for heterogeneous server clusters, followed by a discussion on solutions proposed to improve cooling efficiency for data centers. In the subsequent section, we discuss the related work in global scheduling for data centers, which includes current research that comes closest to our work.

Power management for heterogeneous server clusters is a particularly challenging problem. Heterogeneous servers have differences in their power and performance, thereby resulting in a huge number of possible configurations which could be used to service user requests. To handle this complexity, heterogeneous servers - servers that differ in their power and performance - need to be characterised in order to obtain a model that captures the power consumption of the server as the utilisation varies. These models are obtained by executing workloads or micro-benchmarks [27], while simultaneously measuring the power consumption, utilisation, or other metrics of interest. With models available for the heterogeneous servers in a cluster, the appropriate set of servers that can handle the current request rate can be kept active. Using the model for each server, Heath et al [27] formulate an optimisation problem, which is solved to obtain the best request distribution among servers for each load. The solution is calculated offline and then used for power management. With their approach, they show close to 29% reduction in power consumption.

Guerra et al [26] model the server behaviour using the frequencies of the processors. With each server assumed to be a queue, the incoming requests, are distributed amongst servers in the proportion of the frequencies. With this assumption, their solution attempts to calculate the frequency that each processor needs to be executing at in order to service the current load. Here too, the solutions are computed offline and used for setting the frequency of the servers that constitute the cluster. Horvath et al [29] provide a power management solution targeted for multi-tiered server clusters, where each tier is homogeneous, but servers across tiers are heterogeneous. Their approach involves characterising

servers using an approach similar to [27] and formulating an integer optimisation problem, the solution to which provides the number of servers that need to be available in each tier, and providing up to 23% power savings.

Other work has attempted to use machine learning techniques to power manage server clusters. Tesauro et al [48] use a reinforcement learning based approach to provide power management for a homogeneous blade server cluster. A neural network that is trained using the frequency of the processor and the response time measured is used to set the frequencies of processors of the servers that constitute the cluster, providing close to 20% savings in power.

The other aspect of data center power consumption is cooling and vendors and data center operators are using mechanisms to reduce the cost resulting from cooling. Cooling costs can be reduced using structural efficiencies, mechanical means and software means. Hot and cold aisle containment is a commonly used technique where the warm air exhaust from servers is isolated from the air conditioning vents, thus preventing mixing of warm and cool air. Mechanical devices are also used [31, 32] in order to improve the cooling efficiency of data centers. Intel claims that savings resulting from improved cooling efficiency could result in an annual cost reduction of 2.87 million USD for a 10 Mega Watt data center [32]. Moore et al [38] focused on temperature aware scheduling of tasks to servers within a data center. They used a fine grained model of the variation of server temperature and using computational fluid dynamics simulations, schedule tasks in order to reduce hot-spots, which result in cooling inefficiencies.

The current body of work that focuses on cluster power management and cooling efficiencies provide us with solutions that can be applied locally to a data center. Our work focuses on scheduling requests across globally diverse data center locations in an attempt to reduce costs. Local power management schemes and details of cooling efficiency measures are outside the scope of our work.

5.3 The case for heterogeneous data centers

Heterogeneity has been an active area of research. Current work on scheduling for heterogeneous processors has shown that such processors provide reduced power consumption

without substantial loss of performance. This result can be extended to data centers also. Chun et al [15] showed that data centers that contained a both low performance and high performance machines provided lower power consumption. There approach however, was a greedy approach, where tasks are progressively assigned starting from the low power servers to the high power servers. Unlike our work, this work however does not focus on cost optimisation but only makes a case about the energy efficiency of using low and high performance machines.

5.4 Cost reduction for global data centers

In our work, we attempt to leverage differences in electricity prices across timezones to minimise the electricity costs of data centers under service rate and response time constraints. The problem we consider is quite different from power management for server clusters which provide solutions that can be applied locally within a data center. Further, these solutions do not consider variables such as electricity prices and data center efficiency which are external to a server cluster power-performance profile. The work that come closest to ours are the by Qureshi et al [41] and Le at al [37].

Qureshi et al [41] investigated the cost savings accrued by leveraging the temporal and geographical variation in electricity costs and shifting computation across energy markets. This study makes a case for leveraging electricity prices and reducing costs and also demonstrates power savings that can be obtained by using heuristic methods of load balancing across multiple data centers. However, this study did not take any quality of service constraints into account. The Least Electricity Price Scheduling algorithm we implement is a greedy, heuristic method of leveraging the electricity prices to reduce cost.

Le et al [37] investigated load balancing across global data centers utilising renewable (green) and non-renewable (brown) energy costs, focusing on leveraging electricity generated using renewable sources compared to non-renewable sources. With a load balancing schedule that attempts to use green energy, the total cost however could increase, and they report a cost increase of close to 10% though the brown energy usage is reduced by close to 24%. Our work does not attempt to reduce the carbon footprint of data centers by selecting between different energy sources, instead, we concentrate on reducing the costs incurred by

the data center operators.

In our work, we consider the all the factors that result in cost for data centers - both the computation and non-computation costs. Further, we consider data centers which are heterogeneous as opposed to homogeneous, which increases the complexity of the problem. Prior work on global load balancing has focused on homogeneous data centers, where all machines in the data center exhibit the same power and performance profiles. The problem formulation we presented inherently captures the variation in power consumed as the request rate, and consequently the utilisation, changes. An important aspect of our solution is the focus on server heterogeneity. Prior work discussed does not take the server heterogeneity into consideration in the scheduling ; Qureshi et al [41] for example, assign requests to data centers as long as the data center has spare capacity. Heterogeneity inherently makes the scheduling problem difficult as the service rates for different server types are different and we account for this by explicitly considering the service rates of different server types (as discussed in the problem description).

Chapter 6

Conclusion and Future Work

Cost reduction for data centers is an extremely important problem, and given the increasing demands for data and computation, we can only expect that the problem gains more significance in the time to come. While on the one hand, the costs resulting from operating servers and other computation equipment is high, the costs of cooling data centers to ensure the right operating conditions, also cannot be ignored. The complexity of the problem is further increased due to the presence of heterogeneous servers in today's data centers.

Power management solutions have been proposed for heterogeneous server clusters. Our work takes into account computation and cooling costs for data center cost reduction, while also exploiting, electricity price variations, server heterogeneity, and data center efficiency. We evaluated our algorithm with different scenarios and observed that our algorithm provides cost savings solution provides up to 14% savings in electricity cost, with no loss of quality of service - no requests violate the response time constraints set.

Our solution requires servers to be characterised to obtain the service rates of servers, and their idle and active power consumption; characterisation is typically a manual process. A future extension for our work is to consider heterogeneous requests - where requests differ in their compute requirements. By considering the service rate of servers, we assume that the requests are homogeneous in nature. However, a web application typically serves different types of requests which differ in their compute requirements, for example serving a static HTML page takes less processing compared to dynamically generated content.

Our results also indicate that our algorithm is able to provide higher cost savings with

a mix of low and high performance servers. This is particularly interesting especially since companies have successfully prototyped web server clusters comprising of low power nodes [9, 25].

Bibliography

[1] Ameren : American Energy Exchange. www.ameren.com.

[2] APX-Endex : Energy exchange for The Netherlands. www.apxendex.com.

[3] Bandwidth prices. http://www.creativedata.net/index.cfm?webid=169.

[4] Colocation.com Tiered Bandwidth Prices. http://www.colocation.com/detailedPricingInternet.html.

[5] Wikimedia Servers Meta Website. en.wikipedia.org/wiki/meta:Wikimedia_servers.

[6] Wikimedia Traffic Analysis Report. http://stats.wikimedia.org/wikimedia/squids/SquidReportPageViewsPerCountryOverview.htm.

[7] AMD Corporation. ACP – The Truth About Power Consumption Starts Here. Whitepaper, AMD Corporation. http://multicore.amd.com/resources/43761C_ACP_WP.pdf.

[8] American Society of Heating, Refrigeration, and Air Conditioning Engineers. ASHRAE's Thermal Guidelines for Data Processing Environments. 2008.

[9] David G. Andersen, Jason Franklin, Michael Kaminsky, Amar Phanishayee, Lawrence Tan, and Vijay Vasudevan. FAWN: a fast array of wimpy nodes. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, pages 1–14, New York, NY, USA, 2009. ACM.

[10] Saisanthosh Balakrishnan, Ravi Rajwar, Mike Upton, and Konrad Lai. The impact of performance asymmetry in emerging multicore architectures. In *Proceedings of the 32nd annual international symposium on Computer Architecture*, ISCA '05, pages 506–517, Washington, DC, USA, 2005. IEEE Computer Society.

[11] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40:33–37, December 2007.

[12] Ricardo Bianchini and Ram Rajamony. Power and energy management for server systems. *Computer*, 37:68–74, November 2004.

[13] Pat Bohrer, Elmootazbellah N. Elnozahy, Tom Keller, Michael Kistler, Charles Lefurgy, Chandler McDowell, and Ram Rajamony. *The case for power management in web servers*, pages 261–289. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[14] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, Amin M. Vahdat, and Ronald P. Doyle. Managing energy and server resources in hosting centers. In *Proceedings of the 18th ACM symposium on Operating systems principles*, SOSP '01, pages 103–116, New York, NY, USA, 2001. ACM.

[15] Byung-Gon Chun, Gianluca Iannaccone, Giuseppe Iannaccone, Randy Katz, Gunho Lee, and Luca Niccolini. An energy case for hybrid datacenters. *SIGOPS Oper. Syst. Rev.*, 44:76–80, March 2010.

[16] Matthew Curtis-Maury, Ankur Shah, Filip Blagojevic, Dimitrios S. Nikolopoulos, Bronis R. de Supinski, and Martin Schulz. Prediction models for multi-dimensional power-performance optimization on many cores. In *PACT '08: Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 250–259, New York, NY, USA, 2008. ACM.

[17] E. N. Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy-efficient server clusters. In *Proceedings of the 2nd international conference on Power-aware computer systems*, PACS'02, pages 179–197, Berlin, Heidelberg, 2003. Springer-Verlag.

[18] Elmootazbellah N Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy conservation policies for web servers. In *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems - Volume 4*, USITS'03, pages 8–8, Berkeley, CA, USA, 2003. USENIX Association.

[19] Environmental Protection Agency. Report to Congress on Server and Data Energy Efficiency. 2007.

[20] Free Software Foundation. *GNU Linear Programming Kit (GLPK)*.

[21] Soraya Ghiasi, Tom Keller, and Freeman Rawson. Scheduling for heterogeneous processors in server systems. In *CF '05: Proceedings of the 2nd conference on Computing frontiers*, pages 199–210, New York, NY, USA, 2005. ACM.

[22] Google. Going Green at Google - Data Center Efficiency Measurements. 2010.

[23] Kinshuk Govil, Edwin Chan, and Hal Wasserman. Comparing algorithm for dynamic speed-setting of a low-power cpu. In *Proceedings of the 1st annual international conference on Mobile computing and networking*, MobiCom '95, pages 13–25, New York, NY, USA, 1995. ACM.

[24] Green Grid. Green Grid : Free Cooling Savings Calculator.

[25] Greenm3. Microsoft Research Builds Intel Atom Servers. `http://www.greenm3.com/gdcblog/2009/2/25/microsoft-research-builds-intel-atom-servers.html`.

[26] Raphael Guerra, Luciano Bertini, and J. C. B. Leite. Improving response time and energy efficiency in server clusters. In *Proceedings of the 8th Workshop de Tempo*, VIII Workshop de Tempo, 2006.

[27] Taliver Heath, Bruno Diniz, Enrique V. Carrera, Wagner Meira, Jr., and Ricardo Bianchini. Energy conservation in heterogeneous server clusters. In *Proceedings of the 10th ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPoPP '05, pages 186–195, New York, NY, USA, 2005. ACM.

[28] Urs Hoelzle and Luiz Andre Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st edition, 2009.

[29] Tibor Horvath and Kevin Skadron. Multi-mode energy management for multi-tier server clusters. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, PACT '08, pages 270–279, New York, NY, USA, 2008. ACM.

[30] Rob J. Hyndman, Anne B. Koehler, Ralph D. Snyder, and Simone Grose. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, 18:439–454, 2000.

[31] Intel Corporation. Reducing Data Center Energy Consumption with Wet Side Economizers. 2007.

[32] Intel Corporation. Reducing Data Center Cost with an Air Economizer. 2008.

[33] Jeffrey Fulmer. *Siege - HTTP Tester and Benchmarking utility*.

[34] Jonathan Koomey. Estimating total power consumption by servers in the US and the world. Technical report, Lawrence Berkeley National Laboratory, February 2007.

[35] Andrew Krioukov, Prashanth Mohan, Sara Alspaugh, Laura Keys, David Culler, and Randy Katz. NapSAC: Design and Implementation of a Power-Proportional Web Cluster. In *Proceedings of the 1st ACM SIGCOMM Workshop on Green Networking*, Aug 2010.

[36] Kien Le, Ricardo Bianchini, Thu D. Nguyen, Ozlem Bilgir, and Margaret Martonosi. Capping the brown energy consumption of internet services at low cost. In *Proceedings of the International Conference on Green Computing*, GREENCOMP '10, pages 3–14, Washington, DC, USA, 2010. IEEE Computer Society.

[37] Kien Le, Ozlem Bilgir, Ricardo Bianchini, Margaret Martonosi, and Thu D. Nguyen. Managing the cost, energy consumption, and carbon footprint of internet services. *SIGMETRICS Perform. Eval. Rev.*, 38(1):357–358, 2010.

[38] Justin Moore, Jeff Chase, Parthasarathy Ranganathan, and Ratnesh Sharma. Making scheduling "cool": temperature-aware workload placement in data centers. In *ATC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*. USENIX Association, 2005.

[39] Ripal Nathuji, Canturk Isci, and Eugene Gorbatov. Exploiting platform heterogeneity for power efficient data centers. In *ICAC '07: Proceedings of the 4th International Conference on Autonomic Computing*, page 5, Washington, DC, USA, 2007. IEEE Computer Society.

[40] Eduardo Pinheiro, Ricardo Bianchini, Enrique V. Carrera, and Taliver Heath. Load balancing and unbalancing for power and performance in cluster-based systems. Technical report, 2001.

[41] Asfandyar Qureshi. Plugging Into Energy Market Diversity. In *7th ACM Workshop on Hot Topics in Networks (HotNets)*, Calgary, Canada, October 2008.

[42] Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John Guttag, and Bruce Maggs. Cutting the electric bill for internet-scale systems. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 123–134, New York, NY, USA, 2009. ACM.

[43] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[44] Cosmin Rusu, Alexandre Ferreira, Claudio Scordino, and Aaron Watson. Energy-efficient real-time heterogeneous server clusters. In *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 418–428, Washington, DC, USA, 2006. IEEE Computer Society.

[45] A.J. Shah and N. Krishnan. Optimization of global data center thermal management workload for minimal environmental and economic burden. *Components and Packaging Technologies, IEEE Transactions on*, 31(1):39 –45, March 2008.

[46] Vivek Sharma, Arun Thomas, Tarek Abdelzaher, Kevin Skadron, and Zhijian Lu. Power-aware qos management in web servers. In *Proceedings of the 24th IEEE International Real-Time Systems Symposium*, RTSS '03, pages 63–, Washington, DC, USA, 2003. IEEE Computer Society.

[47] Swaminathan Sivasubramanian, Guillaume Pierre, and Maarten van Steen. Replicating web applications on-demand. In *Services Computing, 2004. (SCC 2004). Proceedings. 2004 IEEE International Conference on*, 2004.

[48] G. Tesauro, R. Das, H. Chan, J. O. Kephart, C. Lefurgy, D. W. Levine, and F. Rawson. 2008) managing power consumption and performance of computing systems using reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems*, NIPS 2008, 2008.

[49] Guido Urdaneta, Guillaume Pierre, and Maarten van Steen. Wikipedia workload analysis for decentralized hosting. *Computer Networks*, 53(11):1830–1845, 2009.

[50] Eric Jan van Baaren. WikiBench: A distributed, Wikipedia based web application benchmark.

[51] Verizon. Verizon IP Latency Statistics. http://www.verizonbusiness.com/about/network/latency/.

[52] Wikimedia Foundation. Wikipedia Statistics Website. `stats.wikimedia.org`.

[53] Shui Yu, Robin Doss, Theerasak Thapngam, and David Qian. A transformation model for heterogeneous servers. *High Performance Computing and Communications, 10th IEEE International Conference on*, pages 665–671, 2008.