Model-Based Design, Analysis, and Implementations for Power- and Energy-Efficient Computing Systems

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

bei der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation

von

Waqaas Munawar

aus Sialkot, Pakistan

Tag der mündlichen Prüfung:	30. Jan 2019
Erster Gutachter:	Prof. Dr. Hartmut Schmeck
Zweiter Gutachter:	Prof. Dr. Wolfgang Karl

Acknowledgements

First and foremost, I would like to thank Prof. Dr. Hartmut Schmeck for his insightful advice, unbureaucratic support, and unwavering patience that has led me here. I cannot thank him enough for the time – a priceless commodity – he spent reviewing my work and helping me polish it. It would not have been possible without him. Second, a heart-felt thanks to Prof. Dr. Jian-Jia Chen for his support and valuable guidance in my journey as a graduate student. I learned a lot from him.

I would also like to thank Prof. Dr. Wolfgang Karl for agreeing to be the second reviewer for my thesis and Prof. Dillman and Prof. Hein for agreeing to be on my defense committee.

Additional thanks go out to my many colleagues and collaborators that have accompanied me through the years, with whom I had lively discussions, and from whom I often received much needed-criticism and support in polishing my work.

Finally, I shall remain forever indebted to my parents, without whom I would be nowhere; and my family, for the constant support, love and smiles.

وقاص منوّر

Waqaas Munawar
Karlsruhe

List of Publications

This dissertation summarizes the results published in several conferences, journals, workshops as follows:

- Waqaas Munawar, Heba Khdr, Santiago Pagani, Muhammad Shafique, Jian-Jia Chen, and Jörg Henkel."Peak Power Management for scheduling real-time tasks on heterogeneous many-core systems", In: 20th IEEE International Conference on Parallel and Distributed Systems, ICPADS, Hsinchu, Taiwan, December 16-19. IEEE Computer Society, 2014, pp. 200-209. DOI: 10.1109/PADSW.2014.7097809.

- Waqaas Munawar, Jian-Jia Chen, and Minming Li. "Minimizing Environmental Footprints of Data Centers under Budget and Service Requirement Constraints". In: SMARTGREENS 2014 —Proceedings of the 3rd International Conference on Smart Grids and Green IT Systems, Barcelona, Spain, 3-4 April. 2014, pp. 222-232. DOI: 10.5220/0004934202220232.

- Waqaas Munawar and Jian-Jia Chen. "Peak power demand analysis and reduction by using battery buffers for monotonic controllers". In: 2013 23rd International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), Karlsruhe, Germany, September 9-11, 2013. IEEE, 2013, pp. 255-258. DOI: 10.1109/PATMOS.2013.6662185.

- Waqaas Munawar, Janmartin Jahn, Artiom Aleinikov, Jian-Jia Chen, Jörg Henkel, "An Empirical Feedback Provider for Multi Core Schedulers" in MARC Symposium 2011, Ettlingen, Germany, July 2011

Furthermore, following is a list of other co-authored works:

- Shengquan Wang, Waqaas Munawar, Jun Liu, Jian-Jia Chen, and Xue Liu. "Power-Saving Design for Server Farms with Response Time Percentile Guarantees". In: 2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium, Beijing, China, April 16-19. 2012, pp. 273-284. DOI: 10.1109/RTAS.2012.35.

- Che-Wei Chang, Jian-Jia Chen,Waqaas Munawar, Tei-Wei Kuo, and Heiko Falk. "Partitioned Scheduling for Real-time Tasks on Multiprocessor Embedded Systems with Programmable Shared Srams". In: Proceedings of the Tenth ACM International Conference on Embedded Software. EMSOFT'12. Tampere, Finland: ACM, 2012, pp. 153-162. ISBN: 978-1-4503-1425-1. DOI: 10.1145/2380356.2380384.

- Shenquan Wang, Waqaas Munawar, Xue Liu, Jian-Jia Chen, "Power-Saving Design in Server Farms for Multi-Tier Applications under Response Time Constraint", International Conference on Smart Grids and Green IT Systems (SMARTGREENS), Aachen, Germany, May 2013.
- Jia-Chun Lin, Fang-Yie Leu, Ying-Ping Chen, and Waqaas Munawar. "Impact of MapReduce Task Re-execution Policy on Job Completion Reliability and Job Completion Time". In: 2014 IEEE 28th International Conference on Advanced Information Networking and Applications. 2014, pp. 712-718. DOI:10.1109/AINA.2014.87
- Santiago Pagani, Heba Khdr,Waqaas Munawar, Jian-Jia Chen, Muhammad Shafique, Minming Li, and Jörg Henkel. "TSP: Thermal Safe Power - Efficient power budgeting for many-core systems in dark silicon". In: International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS. (Uttar Pradesh, India). Ed. by Radu Marculescu and Gabriela Nicolescu. ACM, Oct. 12, 2014, pp. 1-10. DOI: 10.1145/2656075.2656103.
- Waqaas Munawar, Muhammad Hamad Alizai, Olaf Landsiedel, and Klaus Wehrle. "Modular Remote Reprogramming of Sensor Nodes". In: International Journal for Sensor Networks 19.3/4 (Nov. 2015), pp. 251-265. ISSN:1748-1279. DOI:10.1504/ IJSNET.2015.072868.

Abstract

Modern computing systems are becoming increasingly complex. On one end of the spectrum, personal computers now commonly support multiple processing cores, and, on the other end, Internet services routinely employ thousands of servers in distributed locations to provide the desired service to its users. In such complex systems, concerns about energy usage and power consumption are increasingly important. Moreover, growing awareness of environmental issues has added to the overall complexity by introducing new variables to the problem. In this regard, the ability to abstractly focus on the relevant details allows model-based design to help significantly in the analysis and solution of such problems.

In this dissertation, we explore and analyze model-based design for energy and power considerations in computing systems. Although the presented techniques are more generally applicable, we focus their application on large-scale Internet services operating in U.S. electricity markets. Internet services are becoming increasingly popular in the ICT ecosystem of today. The physical infrastructure to support such services is commonly based on a group of cooperative data centers (DCs) operating in tandem. These DCs are geographically distributed to provide security and timing guarantees for their customers. To provide services to millions of customers, DCs employ hundreds of thousands of servers. These servers consume a large amount of energy that is traditionally produced by burning coal and employing other environmentally hazardous methods, such as nuclear and gas power generation plants. This large energy consumption results in significant and fast-growing financial and environmental costs. Consequently, for protection of local and global environments, governing bodies around the globe have begun to introduce legislation to encourage energy consumers, especially corporate entities, to increase the share of renewable energy (green energy) in their total energy consumption. However, in U.S. electricity markets, green energy is usually more expensive than energy generated from traditional sources like coal or petroleum.

We model the overall problem in three sub-areas and explore different approaches aimed at reducing the environmental foot print and operating costs of multi-site Internet services, while honoring the Quality of Service (QoS) constraints as contracted in service level agreements (SLAs).

Firstly, we model the load distribution among member DCs of a multi-site Internet service. The use of green energy is optimized considering different factors such as (a) geographically and temporally variable electricity prices, (b) the multitude of available energy sources to choose from at each DC, (c) the necessity to support more than one SLA, and, (d) the requirements to offer more than one service at each DC. Various approaches are presented for solving this problem and extensive simulations using Google's setup in North America are used to evaluate the presented approaches.

Secondly, we explore the area of shaving the peaks in the energy demand of large electricity consumers, such as DCs by using a battery-based energy storage system. Electrical demand of DCs is typically peaky based on the usage cycle of their customers. Resultant peaks in the electrical demand require development and maintenance of a costlier energy delivery mechanism, and are often met using expensive gas or diesel generators which often have a higher environmental impact. To shave the peak power demand, a battery can be used which is charged during low load and is discharged during the peak loads. Since the batteries are costly, we present a scheme to estimate the size of battery required for any variable electrical load. The electrical load is modeled using the concept of arrival curves from Network Calculus. Our analysis mechanism can help determine the appropriate battery size for a given load arrival curve to reduce the peak.

Thirdly, we present techniques to employ intra-DC scheduling to regulate the peak power usage of each DC. The model we develop is equally applicable to an individual server with multi-/many-core chips as well as a complete DC with an intermix of homogeneous and heterogeneous servers. We evaluate these approaches on single-core and multi-core chip processors and present the results.

Overall, our work demonstrates the value of model-based design for intelligent load distribution across DCs, storage integration, and per DC optimizations for efficient energy management to reduce operating costs and environmental footprint for multi-site Internet services.

Zusammenfassung

Moderne Computersysteme werden immer komplexer. Am einen Ende des Spektrums stehen die heutigen PCs die häufig mehrere Prozessorkerne besitzen, und am anderen Ende die Internet-Dienste, die aus Tausenden von Servern in verteilten Standorten bestehen, um den gewünschten Service für ihre Benutzer zu ermöglichen. In solch komplexen Systemen ist die Betrachtung der Energie- bzw. Leistungsverbrauchswerte ein zunehmend wichtiger Aspekt. Darüber hinaus hat das wachsende Umweltbewusstsein die Gesamtkomplexität des Problems durch Einführung neuer Variablen erhöht. In diesem Zusammenhang ermöglicht es der Ansatz des modellbasierten Entwurfs (Model-Based-Design), sich in der Analyse und Lösung solcher Probleme auf die relevanten Details zu konzentrieren.

In dieser Arbeit untersuchen und analysieren wir das Model-Based-Design in Hinblick auf Energieverbrauch und Energieeffizienz von Rechensystemen. Obwohl die vorgestellten Techniken allgemein anwendbar sind, konzentrieren wir uns auf ihre Anwendung auf groß angelegte Internet-Dienste die in den US-amerikanischen Strommärkte tätig sind. Internet-Dienste werden im heutigen ICT-Ökosystem immer beliebter. Die physikalische Infrastruktur, die solche Dienste unterstützt, basiert häufig auf einer Gruppe von kooperativen Rechenzentren (Data Centers - DCs), die im Tandem arbeiten. Diese Rechenzentren sind geographisch verteilt, um für ihre Kunden Sicherheit und angemessene Antwortzeiten zu gewährleisten. Um eine große Anzahl von Kunden bedienen zu können, setzen Rechenzentren Hunderttausende von Servern ein. Diese Server verbrauchen große Mengen an Energie, die traditionell durch Verbrennen von Kohle oder unter Verwendung anderer umweltgefährdender Methoden, wie Kern- und Gaskraftwerke, gewonnen wird. Dieser große Energieverbrauch verursacht bedeutende und schnell wachsende finanzielle und umwelttechnische Kosten. Folglich haben die Staatsregierungen rund um die Welt begonnen, Rahmenprogramme einzuführen, um den Energieverbrauch zu optimieren für den Schutz der lokalen und globalen Umwelt. Insbesondere Unternehmen sind aufgefordert, den Anteil der erneuerbaren Energie (grüne Energie) in ihrem Gesamtenergieverbrauch zu erhöhen. Jedoch auf den US-amerikanischen Strommärkten ist die grüne Energie in der Regel teurer als die traditionelle Energie, die aus Kohle oder Erdöl erzeugt wird.

Wir untergliedern das Gesamtproblem in drei Teilbereiche und untersuchen verschiedene Ansätze zur Verringerung der Umweltbelastung und Betriebskosten von verteilten Internet-Diensten unter Beibehaltung der Dienstgüte (QoS), wie in den Service Level Agreements (SLAs) vereinbart.

Zuerst modellieren wir die Lastverteilung zwischen den angeschlossenen Rechenzentren eines verteilten Internet-Diesntes. Die Verwendung grüner Energie wird unter Berücksichtigung verschiedener Faktoren optimiert, wie zum Beispiel (a) geographisch und zeitlich variabler Strompreise, (b) der Anzahl der zur Verfügung stehenden Energiequellen an jedem Rechenzentrum, (c) der Notwendigkeit, mehr als eine SLA zu unterstützen, und (d) das Erfordernis, mehr als einen Dienst an jedem Rechenzentrum anzubieten. Es werden verschiedene Ansätze zur Lösung dieses Problems vorgestellt und umfangreiche Simulationen des nord-amerikanischen Google-Setups verwendet, um die vorgestellten Ansätze zu evaluieren.

Zweitens erforschen wir die Möglichkeit der Spitzenabdeckung im Energiebedarf eines Rechenzentrums durch ein batteriebasiertes Energiespeichersystem. Der Strombedarf von Rechenzentren ist aufgrund der Verwendungszyklen seiner Kunden typischerweise spitzenlastig. Resultierende Spitzen im Strombedarf erfordern (a) die Entwicklung und Aufrechterhaltung teurer Energieabgabemechanismen, und werden (b) durch kostspielige Gas- oder Dieselgeneratoren gedeckt, die eine höhere Umweltbelastung verursachen. Um die Spitzen zu decken, kann eine Batterie verwendet werden, die bei niedriger Last aufgeladen und während der Spitzenlasten entladen wird. Da die Batterien teuer sind, präsentieren wir ein System, welches die Bestimmung der optimalen Größe der Batterie erlaubt. Die elektrische Last modellieren wir mit dem Konzept der Arrival-Curves aus der Network Calculus. Unser Analyse-Mechanismus kann dazu beitragen, die benötigte Batteriegröße für eine gegebene Load-Arrival-Curve zu bestimmen, um die Spitzenabdeckung zu reduzieren.

Drittens stellen wir Techniken zur verfügen, welche ermöglichen den Spitzenstromverbrauch jedes Rechenzentrums zu regulieren durch ein optimiertes Scheduling-Verfahren. Das Modell, das wir entwickeln, ist in gleicher Weise anwendbar auf einen einzelnen Server mit Mehrkern-/Vielkern-Prozessoren sowie auf ein komplettes Rechenzentrum mit homogenen und heterogenen Gruppen von Servern. Wir bewerten diese Ansätze auf Einzel- sowie Mehrkernprozessoren und präsentieren die Ergebnisse.

Insgesamt zeigt unsere Arbeit den Wert des modellbasierten Entwurfs für eine intelligente Lastverteilung auf Rechenzentren, Batterie-Integration und Optimierungen der Rechenzentren für ein effizientes Energie-Management im Hinblick auf die Senkung der Betriebskosten und der Umweltbelastung für verteilte Internet-Dienste.

Contents

1 Introduction

Over the past few decades, computing systems have grown increasingly complex. This holds true for both the processing chips and the compute architecture that surrounds them. Increasing internal complexity can be seen in the prevalence of multi-core processing chips with multiple levels of memory caches and multistage instruction pipelines. Analogously, the increasing complexity of the external architecture is reflected in the diversity of applications for which computers are now used. On one end of the spectrum are resource- and energy-constrained embedded devices such as nodes in a wireless sensor network, while on the other end, there are Internet services supporting millions of users through geographically distributed, collaborative data centers, each comprising of thousands of individual servers. The complexity of such systems makes them opaque to the users, forcing them to trust the system blindly. Similar hurdles are faced by designers also, who often have to resort to intuition and expert-opinion rather than real data when designing and optimizing such systems. In this context, attempts to optimize useful aspects of such complex systems can benefit from structured methods that allow designers to focus on essential attributes by abstracting out the irrelevant details. These essential attributes and their interplay constitute a model of a system.

Models can be of different types, depending on the kind of information they contain, the level of formality they use, how they are represented, and the level of abstraction. An important design choice is to select the most appropriate model for the current goal.

In this dissertation, we develop model-based design and analysis techniques to address the efficiency of large-scale computing systems. In particular, we focus on aspects, such as environmental impact, operating costs and temperature-related issues. Examples of large-scale computing systems are multi data center based Internet services offered by Google, Amazon, Microsoft and others.

Due to the geographically distributed architecture of such services, their constituent DCs often fall into service areas of different electricity markets. The electricity markets belonging to different regions show great variance among them, due to factors such as fuel availability, weather patterns and local legislation, among others. To keep the problem tractable, we use the data and trends from electricity markets of only one country, i.e., the United States. The techniques we develop and present in this thesis are, however, not limited to Internet services only and are more generally applicable. The other possible areas of application are smart grids and real-time systems, among others.

In this chapter, we motivate our work by discussing the energy needs and resulting environmental impact that multi data center based Internet services are creating globally. After this, we present a short overview of the techniques we developed to tackle these issues. Last, we document our contributions and present an overview of the entire work.

1.1 Motivation

Since the Industrial Revolution, the amount of green house gases (GHG) emitted annually due to fuel combustion has increased exponentially as shown in Figure 1.1. Compared to the pre-industrial era estimate of 280 ppm, the concentration of CO_2 in 2017 stood at 408 ppm, i.e., around 45% higher, with an average growth of 2 ppm/year in the last decade [76].

The negative effects of global warming on the environment are gradually being realized and awareness of the need to reduce GHG emissions is increasing. The trend can be seen in the new legislation and recommendations by governmental agencies all over the world [148, 141, 147]. Recently, to this effect, United Nations Framework Convention on Climate Change (UNFCCC) organized a conference in Paris with the aim of *"... stabilization of greenhouse gas concentrations in the atmosphere at a level that would prevent dangerous anthropogenic interference with the climate system."* [147]. UNFCCC boasts almost universal membership with 195 nations signatory to the convention. However, growing world energy demand is still playing a key role in the upward trend in GHG emissions, because it is satisfied chiefly by burning fossil fuels. It has been estimated that around three fourth of the global man-made (anthropogenic) GHG emissions are a result of burning fossil fuel, primarily coal, to produce electricity and heat [76], as shown in Figure 1.2.

One of the prime consumers of electricity is the so-called information and communication technology (ICT) sector [78, 83, 155]. Specifically, it is estimated that ICT products and services accounted for 4.6% of the global electricity consumption in 2012 [155]. This share increased from an estimated 3.9% in 2007, notwithstanding the fact that the global economy was going through a recession during this period.

Within the ICT sector, the major impact is generated by data centers. It has been reported that in 2005 in the United States, data centers consumed 56 TWhs. This

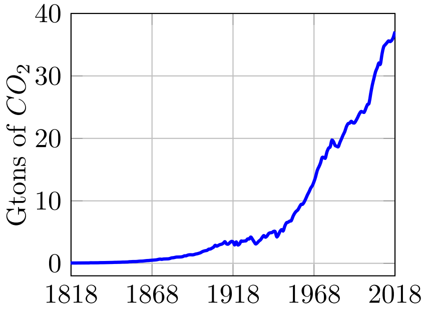


Figure 1.1: Trend in CO_2 emissions from fossil fuel combustion [19, 90]

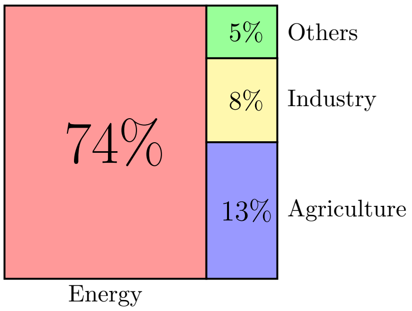


Figure 1.2: Global shares of anthropogenic GHG [76]

number increased by 53% to 85.6 TWhs in 2010, representing about 2.2% of total U.S. electricity use [83]. A similar trend is seen all over the world, as seen in Figure 1.3. Globally, the electricity usage for data centers has been increasing with a compounded annual growth rate of 4.4% with no signs of recession. Moreover, the amount of energy consumed for cooling down the servers is comparable to the energy consumed for actual computing, both categories shown as infrastructure and servers in Figure 1.3, respectively.

Owing to their massive electricity usage, data centers cause a significant environmental impact. For instance, for each of four and a half million registered users of Second Life, the average avatar consumes 1,752 kWh per year, or about two-thirds that of an actual person, globally averaged [28]. This, in terms of CO_2 emissions, is equivalent to 1.17 tonnes, i.e., the same as driving an average-size car for around 5500 km [28]. Similarly, each search at Google is estimated to produce about 0.2g of CO_2 [71]. Combining this with the estimation of the one billion daily search requests processed by Google [101] results in an estimate of 200 tons of GHG emissions due to Google searches everyday.

In view of these facts, it is imperative that the ICT sector, in general, and Internet services, in particular, strive to reduce their environmental footprint. In this dissertation we employ model-based design to develop schemes with the potential to efficiently minimize the environmental impact and reduce operating costs by regulating energy usage for the computing systems of interest. In the following sections we present a short overview of the developed approaches.

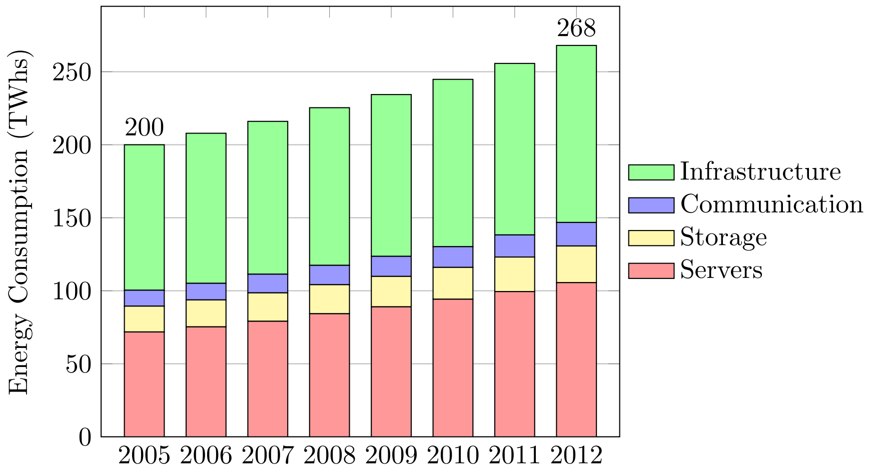


Figure 1.3: Worldwide electricity consumption breakdown for data centers [155]

1.2 Renewable Energy Utilization

One way to control GHG emissions is to use electricity obtained through renewable sources like wind and the sun, instead of consuming fossil fuels like coal, petroleum and natural gas. Renewable sources provide a virtually limitless clean supply of energy without releasing any GHG in the atmosphere. The “green” energy generation facilities are often co-located at sites where the energy is to be consumed and the energy that is produced is directly used to power the devices [8]. However, these “green” sources suffer from an inherent unreliability due to their dependence on the weather. For instance, the peak in production of the electricity - noon, in case of solar panels - might not correspond to the peak in consumption - 08:00 AM to 12:00 PM on weekdays for Internet services. Similarly, the peaks in the production of wind and water based renewable sources does not necessarily occur at the same time or with the same frequency as peaks in electricity demand. This can be overcome by over-designing the green energy setup such that, even at minimum production, it exceeds the peaks in demand. Considering most on-site green energy generation is through solar panels which have almost no output at cloudy days, this approach, however, can be overly expensive. The other way is to provide a back-up fossil fuel-based power (brown power) infrastructure to fall back upon. Because of this, the amortized cost of producing a unit of green energy is more than that of brown energy [163].

Table 1.1: Annual financial and environmental costs of Internet Services [127].

Company	**Servers**	**Electricity**	**Cost**	**CO_2 (Tons)**
eBay	16K	0.6×10^5 MWh	\$3.7M	0.43×10^7
Akamai	40K	1.7×10^5 MWh	\$10M	1.2×10^7
Rackspace	50K	2.0×10^5 MWh	\$12M	1.4×10^7
Microsoft	> 200K	$> 6 \times 10^5$ MWh	>\$36M	4.3×10^7
Google	> 500K	$> 6.3 \times 10^5$ MWh	>\$38M	4.5×10^7

An alternative approach followed by many data center operators in the U.S. is to outsource green energy production to specialist third parties, such as a wind farm operator [44]. The electricity produced in such a facility is contributed to the grid and sold at the same price as brown energy. However, for every unit of energy contributed, they are allowed to sell instruments, such as renewable energy credits (RECs). Other businesses must buy these RECs to show the reduction of their environmental impact. RECs are an additional regulatory source of income for renewable energy producers which offset higher production costs of renewable energy to bring it at par with brown energy [58]. In comparison, businesses in European Union have to buy carbon credits equivalent to their emissions thereby producing a regulatory addition in the price of brown energy and making it less desirable, as discussed further in Section 2.2.7.

The fundamental issue driving the regulation is that renewable energy is costlier to produce than brown energy. According to United States market survey, the cost of production of wind energy is expected to be in the range of 7 to 20 cents per kWh for the next 5 years. Similarly, solar energy per kWh is expected to range from 12 to 24 cents, depending upon the method of production. In comparison, brown energy typically costs $1{\sim}15$ cents per kWh [153, 132]. Here, we consider only the cost of production, and not the cost of delivering the energy to the end consumer, as we assume delivery expenses will add a constant offset on top of production costs, irrespective of the generation method. We also neglect the tax levied on brown energy production, as it imposed on production plants that are commissioned after 2022.

In addition to the environmental impact of massive energy consumption, data centers are faced with high electricity bills for their energy usage. Table 1.1 presents estimated costs and environmental impacts of some of the major Internet services. Reducing the environmental footprint also involves expenses in addition to normal operating costs. In view of this, the decision about when and how much green energy to buy is important and has potential to save operating costs, as well as reducing the environmental impact.

Moreover, typical Internet services are provided through a group of data centers which are distributed geographically. The reasons for this include natural business distribution, the need for high service availability and disaster tolerance, the sheer size of their computational infrastructure, and/or the desire to provide uniform access times to the infrastructure from widely-distributed client sites. Being geographically dispersed provides extra degrees of freedom for cost, as well as environmental impact optimization, by intelligently distributing the incoming requests among the member data centers. In this dissertation we explore these degrees of freedom to reduce brown energy usage while remaining within budget constraints.

1.3 Battery Inclusion for Demand Side Management

As seen in Table 1.1, the data centers have high electricity usage and, consequently, high electricity bills. The cost of electricity depends not only on the amount of electricity a customer uses, but also the time of day at which it is used as well as how quickly it is used. Demand for electricity is quite variable depending on the office-home cycle of the general population and seasonal cycles around the year, but varying the production of electricity on short notice can be expensive. The trends in the price of electricity, like other commodities, follow the dynamics of supply and demand; i.e., the price increases during the periods of high demand and decreases during the non-peak hours.

There are two main drawbacks of uneven demand in electricity. First, the supply must be provisioned for the worst-case power loads which might occur only occasionally. Failing to ensure this might make the generation system unstable, resulting in a catastrophic failure for the whole system. Therefore, the supply network as well as the generation setup must be over-provisioned. Consequently, additional costs are incurred which must be borne by the end consumers of electricity, mostly irrespective of their individual contribution in the peak loading.

Second, the peaks in demand are often met through those generators that can be powered up quickly. These consume natural gas or diesel as fuel, typically. Although they have a short response time, these back-up generators have a higher environmental impact because they are not highly efficient. Clearly, to maximize the return on investment, electricity producers use their most efficient generators for the base demand, and then, as the peak in demand occurs, the left-over, less efficient generators are looped in. Therefore, the runtime of the inefficient generators increases with increasing "peakiness" in demand. Randomness in electricity demand results in higher production costs, as well as higher environmental impact [121, 143].

To make the demand more uniform, power generation companies have introduced tariffs for bigger consumers that penalize peaks and encourage a "flatter" demand. This approach is called "demand side management" (DSM) [121]. In this model, incentives are provided to shape and/or shift the demand. For instance, the actual electricity bill not only depends on how much energy was consumed, but also how quickly it was consumed. If the demands are $(d_1, d_2, ..., d_n)$, then the total bill is of the form $c_1 \cdot \sum_i d_i + c_2 \cdot \max_i\{d_i\}$. Peaks are often measured in the time scale of 15 minutes to 1 hour. The peak penalty is also referred to as demand charge, and it can be more than 300 times the regular cost of energy [6, 49]. Consequently, the demand charge often exceeds the amount of all other charges combined [6].

In this dissertation we model this problem in its basic form for peak minimization using an energy storage element, such as a battery. We employ the framework of network calculus to derive the relationship among battery size and the maximum possible peak.

1.4 Intra Data Center Optimizations

Internet services are often provided by a group of geographically distributed data centers. Optimizations performed within a data center, such as decreasing power consumption or demand side management, can translate into a reduction of overall environmental impact, or financial savings, or both. For such optimizations, the internal architecture of the data center plays an important role.

Internally, each data center consists of thousands of servers. These servers can either be heterogeneous or homogeneous in nature. For a given workload, it can be decided for each server when and for how long to turn it on - dynamic power management (DPM) - or, a finer grained control through selecting an operating frequency for each server, among a set of discrete possible operating frequencies, i.e., dynamic voltage frequency scaling (DVFS). Given thousands of servers, selecting an optimal configuration for each is a difficult combinatorial problem. Moreover, workloads that data centers process are often bound by service level agreements (SLAs) that must be fulfilled. SLAs define contracted boundaries of time that must be respected for processing each request.

In essence, this problem is similar to another well-known problem of dark silicon [50, 112]. The dark silicon problem is described as follows. Processing chips are accumulating an ever-increasing amount of logic units (transistors) as per Moore's law. In contrast, neither on-chip power supply networks nor heat dissipation solutions are keeping up. Consequently, there will be more logic units on processing chips than can be safely turned on, simultaneously. This necessitates scheduling the use of logic units that takes these power-related constraints into consideration.. Considering a real-time workload for a heterogeneous many-core chip of dark silicon era, this problem is analogous to the one described above in the domain of a data center with thousands of heterogeneous servers in the presence of an SLA.

In our work, we model the second variant of the problem, i.e., scheduling a real-time workload for a many-core chip in dark silicon era, such that the peak power is minimized.

1.5 Thesis Contribution

In the scope of this dissertation, the above key concerns are addressed and we make the following contributions.

Optimizing Multiple Data Center Based Internet Services

We focus on decreasing the environmental impact and operating costs of complex computing systems. One of the application areas is distributed Internet services. In this regard, we present the following.

- We show that Internet services that are provided by multiple geographically distributed data centers, can exploit many degrees of freedom for cost savings and environmental impact reduction. Such opportunities stem from differences in electricity prices, time zones, climates, and access to green energy.
- We model the problem of maximizing green energy usage within a complete Internet service (such as Google or iTunes) without exceeding the allocated budget, as an optimization problem. For this multifaceted problem, we develop a holistic model that considers all important factors, such as energy consumption from the infrastructure for networking, computation, and cooling devices, latencies due to geographical distance, differing SLAs, time varying prices of electricity, varying weather conditions affecting green energy availability, and others as explained in detail in Chapter 2.

- As a solution to the above modeled problem, we present software optimization strategies to dynamically decide the distribution of incoming requests at the central hub of an Internet service provider among the constituent data centers.
- We evaluate our optimization strategies extensively with real-world workload traces from Wikipedia [150] and time varying electricity prices from different regions in the United States, obtained from NYISO [117].

Utilizing Energy Storage to Suppress Peak in Electrical Demand

Environmental impact can be reduced by making electrical demand more even over time. This can be achieved by introducing a storage element, such as a battery buffer, in the system. Here, an important question arises: when to store the energy and when to consume it. A false decision about charging and discharging in the presence of a battery buffer can be more damaging than a system without a battery buffer. In this regard, our contributions are as follows.

- We model the electrical loads by adopting the concept of arrival curves in Network Calculus [89]. That is, the electrical loads for the given time interval lengths are upper bounded by the given arrival curve.
- As a step toward peak minimization, we develop a mathematical technique to determine the maximum possible peak that can occur in a system whose control algorithm and battery state are known. The control algorithm decides when to charge and when to discharge the battery. Our analysis technique is limited only to *monotonic* controllers, as defined later in Chapter 4.
- We perform simulations to verify the efficacy of our analysis technique. For these, we use actual electrical load profiles and pricing data from the New York region of the United States [117].
- As an application of the developed technique, we present a case study of a large electricity consumer and determine the amount of savings that are potentially achievable if an appropriately-sized battery buffer is employed to flatten the peaks in demand.

Power Management Within Data Centers

As part of optimizations that can be helpful for power management within each individual data center, we model the similar problem of peak power reduction in the scope of many-core processing chips of the dark silicon era. Guaranteeing timing behavior of real-time tasks on such processing chips is non-trivial because of the existence of performance throttling mechanisms that step in automatically once the processing chip exceeds the temperature limits. Thus a way is needed to partition and schedule the tasks on processing cores in such a way that timing criteria are met and power consumption remains within the limit. This problem is, in fact, a general form of scheduling tasks (such as individual jobs of MapReduce [42]) in a data center such that a peak in the power consumption of the whole data center is minimized. In the scope of this dissertation, we concentrate on the general form of the problem, for which our contributions are as follows.

- We develop a peak power management scheme for many-core processors that execute task sets with real-time requirements. We develop a technique of introducing coordinated sleep cycles in the schedule of each core to minimize peak power consumption, without violating the hard real-time requirements of the tasks.
- In addition to the existing utilization-based schedulability tests, we introduce the concept of a sufficient test for schedulability, considering the peak power consumption of a task set with real-time requirements.
- We analyze and evaluate our techniques using power traces collected from the 48-core SCC [46] platform and gem5 architecture simulator in combination with McPAT [96].

1.6 Thesis Structure and Overview

The remainder of this thesis is organized into five chapters, each dedicated to its own facet of power management and optimization. First, Chapter 2 presents background knowledge that can help in following the topics discussed in the thesis. These include information about DCs, generic architecture of Internet services, energy market dynamics, energy storage systems and SLAs, among others. In Chapter 2, we also include a review of related literature.

In Chapter 3, we explore different aspects of the problem of reducing the environmental footprint of Internet services by regulating their energy usage. We model this problem as an optimization problem and develop algorithms to efficiently solve it. We present a comparative analysis of developed techniques based on a simulation of Google's setup in the United States. The contents of this chapter are developed upon our work as presented in [111]. We extend our previous work by adding more details about our assumptions and a discussion of the complexity of the problem.

In Chapter 4, we model the problem of minimizing the peak in electrical demand of a facility. We present an analysis technique to estimate the highest peak that can possibly occur and evaluate it using real-world data. We also present a case study in this chapter to highlight the benefits of developed technique. The contents of this chapter build upon our work in [110], where we simplify the system model without losing generality, improve earlier proofs, and extend the evaluation.

Chapter 5 presents our work on peak power management in the scope of many-core processors to be able to execute real-time tasks. We present and analyze algorithms for peak power minimization for heterogeneous and homogeneous tasks and processing cores. Evaluations are conducted on the 48-core SCC platform, among others. The contents of this chapter are based on our work as presented in [112]. Here we extend this to include a novel application of our results to reduce the power consumption on DC level.

In Chapter 6, we present a summary of our contributions in this dissertation. Finally, we provide an outlook into the possible directions for future work.

2 Background and Related Work

This chapter provides a brief overview about the background information regarding the important aspects that impact power and energy efficiency in data centers. Also presented is the overview of related work from the domain of energy efficiency and environmental footprint optimization for the Internet services.

2.1 Data Centers

A data center is a facility that is primarily responsible for storing, managing, processing and disseminating important data necessary for business operations of an organization. This facility houses an organization's 1) IT equipment, such as, servers, communication systems and storage devices; and 2) supporting infrastructure, such as, cooling system and backup power generators. From universities to e-commerce and from search engines to social networks, the data centers have become critical to continuity of daily operations of modern organizations. Consequently, the security and availability of data centers is considered very crucial. In this section, we present the relevant information about size, availability and power usage of a DC.

2.1.1 Size

Data centers vary greatly in size and services that they provide. A standard has been developed to measure the size and density of data centers at the scale of mini to mega [7] as shown in Table 2.1. Typically, small scale data centers are employed by small enterprises that may use these as email and web servers. Quite often, these data centers do not have any specialized infrastructure for cooling or power supply. In the middle of the range, there are data centers that have servers, storage devices and other hardware mounted into racks which are arranged into aisles, as shown in Figure 2.1. This arrangement is primarily followed to facilitate the containment of hot and cold air. These data centers have detailed designs for power supply and cooling, and, proportionate significant investment for such infrastructure. Often, to avail the economy of scale,

Figure 2.1: Server racks in Facebook data center in Lulea, Sweden.

Table 2.1: Data center standardized sizes. The focus of this dissertation is on small to large-scale DCs. [7]

Size metric	Rack yield	Compute space (m^2)
Mega	$\geq$ 9,001	$\geq$ 22,501
Massive	3,001–9,000	7,501–22,500
Large	801–3,000	2,001–7,500
Medium	201–800	501–2,000
Small	11–200	26–500
Mini	1–10	1–25

businesses share these facilities in the so-called co-location centers or carrier hotels. Lastly, at the top of the range are data centers with hundreds of thousands of servers with massive power supply and cooling infrastructures. These data centers are typically run by large Internet services such as Google, Amazon and Facebook. Mostly, these have homogeneous hardware and software. In the United States, most of the servers are housed in data centers with the floor area of around 25 thousand m^2 with the current trend of mergers and acquisitions resulting in increasing sizes [22]. As per the standard categorization [7], this floor area corresponds to the category of large scale data centers which form the main segment of the market and hence, the prime target for the power optimization. In this dissertation, we focus mainly on large-scale data centers as per scale shown in Table 2.1, although the presented techniques are applicable more generally.

Table 2.2: Data center availability: Tier classifications to consistently describe site-level infrastructure required to sustain data center operations. In tier IV, Class A continuous cooling requires backup power for the entire cooling plant. Source: [146]

	Tier I	**Tier II**	**Tier III**	**Tier IV**
Active capacity components to support the IT load	N	N+1	N+1	N After any failure
Distribution paths	1	1	1 Active and 1 Alternate	2 Simultaneously active
Concurrently maintainable	No	No	Yes	Yes
Fault tolerance	No	No	No	Yes
Compartmentalization	No	No	No	Yes
Continuous cooling	Load density dependent	Load density dependent	Load density dependent	Class A
Maximum allowed annual downtime (mins)	1729	1361	95	26

2.1.2 Availability

An important metric in building and operating a data center is its availability. Downtime caused by a data center can seriously effect business viability of its customers. Statistics, according to the U.S. National Archives and Records Administration (NARA), show that 93% of companies losing their data center for 10 days or more following a major breakdown have gone bankrupt during the following year [68, p.9].

A 4-tier classification system has been developed to classify the data centers according to their availability [146]. Tier I data centers normally have a single non-redundant power supply and cooling distribution path to serve the servers. Tier II specification requires the data center to have one extra redundancy for every component. This improves the availability to 99.741%, in comparison to Tier I's 99.671%. Specifications for Tier III mandates multiple inde-

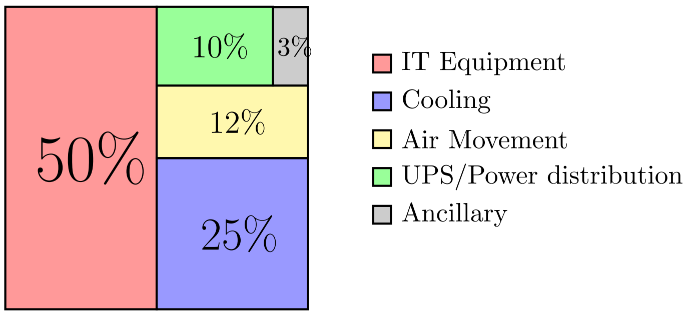


Figure 2.2: Typical breakdown of energy consumption within a data center with PUE of 2.0 [13].

pendent power and cooling paths to IT equipment, though only one is active. Moreover, the ability to concurrently allow site infrastructure maintenance and IT operation must be provided. This requires that each and every system or component that supports IT operations, including back up power and cooling system, must be able to be taken offline for scheduled maintenance without impact to the IT environment. The availability for a tier III system has to be guaranteed to be at least 99.982%. Tier IV requires that, in addition to IT equipment, all cooling equipment is powered by multiple independent paths. Fault tolerant site infrastructure is also required for electrical power storage and distribution facilities. This results in a typical 99.995% availability.

Most commercial data centers fall between Tier III and Tier IV as a trade-off between cost and reliability. The tier based classification system is summarized in Table 2.2.

2.1.3 Power Usage

The power efficiency of a data center is mostly commonly measured through a metric known as power usage effectiveness (PUE). It is defined as the ratio of the total power entering facility to the power being consumed by the computing devices. I.e.,

$$\text{PUE} = \frac{\text{Total Facility Energy}}{\text{IT Equipment Energy}} \tag{2.1}$$

The smaller the value of PUE, the better it is, with the ideal case of 1. According to a recent survey, [74], the industry wide average PUE value is around 1.7. This has been improving from 2.5 in 2007 to 1.8 in 2011. After 2011, it has not followed any significant trend and remained almost stable. Further improvements in this case will require significant investment and effort, with increasingly diminishing returns. However, it has been reported that Google has achieved a PUE of 1.14 combined across all its data centers [79]. Similar figures have been attributed to Microsoft and Facebook [69]. These numbers are significantly below the industry average owing to state-of-the-art practices, massive investment in infrastructure and relatively new facilities the big companies can afford. However, such excellent efficiency is still confined to a small set of data centers, and most of the small data centers have not improved much stabilizing at around 1.7 [13].

A number of factors contribute toward the non-IT energy consumption in data centers. Figure 2.2 provides an overview for this. As shown, cooling contributes the largest fraction, amounting to around 25% of the overall power used [13]. The next biggest contribution is caused by the fans used to regulate the air movement, consuming around 12% of the power used by the IT equipment. The other sources include the UPS units, power distribution units, humidifiers and the lighting.

In literature many schemes have been proposed to improve the PUE [115, 62, 122] of data centers. Since cooling the servers is the most costly operation in terms of energy usage, it offers the maximum returns for energy optimizations. The state-of-the-art in energy minimization for cooling is as follows. A strict segregation is maintained between the hot air exhausted by the servers and the cold air fed to the servers. For this, not only are the aisles carefully sealed, the internals of server carrying racks are specially designed to allow a single directional flow of air. Secondly, maintaining temperature of the input air at 25–30°C instead of the usual 18–20°C make it much easier to cool the data center efficiently. No evidence has been found of higher component failure due to the higher operating temperature of the IT equipment [13, p.89]. Similarly, for water based cooling systems high efficiency values have been reported though it might need specialized infrastructure not realizable through common off-shelf-components [100]. For both water based and air based cooling systems, low external temperatures can be leveraged to avoid the usage of chiller in the cooling system. This can lead to an energy savings of up to 75% [82, p.82] for energy used in cooling.

In summary, it is not only the computing equipment that consumes the energy in a DC. Emphasis must also be given to other factors that contribute toward this, and such factors must be quantified and catered for in a scheme aimed to optimize energy in DCs.

2.2 Electricity Market Dynamics

The electrical grid is a complex and huge system and has often been called the most complex machine ever built by humans. Typically, it consists of three subsystems. First one, the so-called primary system, comprises of a multitude of synchronized power generation plants, hundreds of kilometers of transmission lines and hundreds of thousands of distribution points. Second one, the so-called secondary system, consists of automatic control which is responsible for secure, stable and economic operation of the power system. Thirdly, the market based system for trading. A stable operation of electrical grid requires smooth interplay of all these components. Moreover, as the electricity is seen as one of the basic necessities of modern life, there is an expectation of high reliability and cost effectiveness associated with the electrical grid. Efforts to achieve these expectations increase the complexity of the system further. In this section, we present an overview of the relevant details of the electrical grid that are necessary to follow the techniques presented later in this dissertation. For an extensive discussion of the electrical grid, the reader is referred to [108, 136, 140].

2.2.1 Heterogeneity

All electricity generation and supply networks chiefly consist of the same three subcomponents as mentioned earlier, however, the economic and trading part of the system has been evolving continuously for over a century. Today, from a government-operated non-profit utility model to an open-for-all regulated market, electricity markets in different regions of the world exist in many different flavors. The differences originate out of a multitude of factors, such as, applicable legislation, governmental regulation, investors' interest, fuel availability and legacy factors. Growing share of non-reliable, increasingly distributed, renewable energy sources in the total power mix and different weather patterns add to the overall complexity as well.

To keep the problems tackled in this dissertation tractable in presence of all the mentioned factors, we limit our focus mainly to electricity markets operating in the United States. The motivation for this is as follows. As U.S. electricity markets fall under the jurisdiction of a single regulator, i.e., U.S. Federal Energy Regulatory Commission, they are more homogenized from the legal and operational point of views as compared to their European counterparts. On the other hand, they are quite distributed geographically as they include four distinct time-zones of Northern American continent. This makes them an easier target for optimization schemes that exploit time-varying loads and costs in the electrical grid.

For the rest of the dissertation, any unqualified statement about electricity markets is to be assumed to pertain to U.S. electricity markets.

2.2.2 Competitive Market

Electricity markets can be divided into two main segments: retail and wholesale. Retail markets involve sales of electricity to consumers, whereas, wholesale markets typically involve the sales of electricity among electric utilities and electricity traders before it is eventually sold to consumers. Much of these two markets have evolved to be competitive where prices reflect the factors driving supply and demand.

The wholesale market is governed by two different business models; cost-based pricing model and market-based pricing model. The cost-based pricing model is more orthodox and legacy oriented. As the electricity emerged as a commodity and the user base started growing, the electricity providers took the form of nonprofit municipal utilities. Such setups are mostly vertically integrated, in that, they are responsible for all the phases of electricity delivery, i.e., production, transmission and distribution. The selling price of electricity in such a setup is determined by the cost of production and operations, hence the cost-based pricing model. Another variant of this setup is that of a regulated monopoly. In this case, private investment is contracted for infrastructure development such as a new power generation plant or a transmission line. Cost-based rates are specified by the regulator to insure recovery of the costs associated with providing service as well as a fair return on initial investment. These are paid through increased tariff for the end customers.

In the market-based pricing model, market forces, under regulation of a public body, determine the price of electricity. The public regulation body, i.e., U.S. Federal Energy Regulatory Commission (FERC), sets the legal framework and appoints an independent operator for each regional market. These regional transmission operators (RTOs) are non-profit, regulator-approved entities that operate the transmission system and provide a trading platform with open and equal access to all eligible participants (in Europe, the regulatory body is called Agency for the Cooperation of Energy Regulators (ACER) and market operators are called transmission system operators–TSOs). The main responsibilities of transmission operators include 1) matching the demand and supply of electricity through market based mechanisms in their respective regions, 2) operating a competitive nondiscriminatory market where energy producers can offer their production and the load serving entities (LSEs) can buy and reserve the transmission network for their customers, and, 3) planning and insuring adequate generation for the future requirements considering relevant consumption trends.

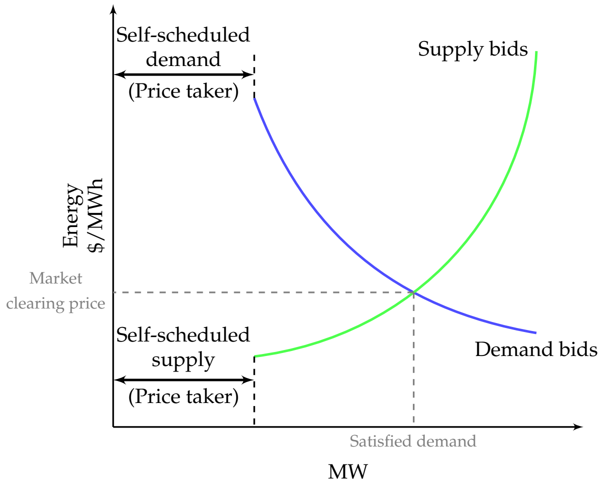


Figure 2.3: RTOs match the supply and demand in the market on the basis of clearing price.

The LSEs and generation units can independently enter into long term contracts with each other for the sale of electricity ranging up to years in advance. Other than this, the trading of electricity on the market occurs mainly in two phases; day-ahead unit commitment, or planning for the next day's energy dispatch, and economic dispatch, or real-time dispatch of the system. In the unit commitment stage, the RTOs typically decide a detailed schedule for electricity generators for each hour of the next 24-hour period. On the basis of this short term trading commitment, the generating units plan their operations a priori as they often need several hours lead time before they can be brought online. Also, other factors, such as physical characteristics of the generating units and weather forecast, are taken into account to guarantee that dispatch can meet the load reliably. In the second stage, the operators must decide in real-time the level at which each available resource from the unit commitment stage should be operated, given *actual* load and grid conditions, so that overall production costs are minimized. Actual conditions can vary from those forecast in the day-ahead commitment, and hence operators must adjust the dispatch accordingly. The real-time or spot market is much more volatile than the day-ahead market.

The operators decide real-time dispatch as per a supply curve which lists all generating units' bids of price at which they are willing to sell the energy as well as the maximum amount of energy they can sell. Sorting these bids from lower to higher and adding them together produces the mentioned supply curve (an example is presented later in Figure 2.6). Likewise, each LSE bids a volume range for a price they are willing to pay. The market clearing price is where the supply and demand curves cross. In other words, starting at the lowest price, the bids from generating units are matched to buyers until all demand bids are covered. The last generator bid accepted sets the market clearing price. This process is explained in Figure 2.3. In contrast to pay-as-you-bid system, a single clearing price system encourages price benefits for end customers. Moreover, because the last increment of demand set the clearing price, an explicit price signal to conserve electricity is established. For certain customers who can reduce their demand, a price incentive can be transparently seen.

As discussed, in a market based system, the supply and demand of electricity in a market decide the price of electricity at any given time. The demand for electricity is heavily influenced by the time of day, weather and other such local phenomena. This results in spatial and temporal variance in the price of electricity. For instance, Figure 2.4 shows the price variability in three cities in North America which lie in different time zones —New York, NY in Eastern Standard Time (EST), Oak Ridge, TN in Central Standard Time (CST) and Los Angeles, CA in Pacific Standard Time (PST). These price variations can be exploited as we discuss later in Chapter 3.

2.2.3 Cost of Energy

The cost of electricity that the end consumers pay, is dependent upon the tariff under which the electricity is sold. A number of factors affect the cost of electricity in an offered tariff, such as, the whole-sale price of electricity in the market, applicable taxation, RTOs' fees, amount of energy and power needed, among others. Typical retail tariffs for the sale of electricity can be divided into two main categories; static and dynamic pricing tariffs.

Static pricing tariffs are those under which the price of energy has been fixed a priori. Most common among these is the flat tariff commonly offered to residential consumers. Under this tariff, every unit of energy (kWh) costs the same throughout the budgeting period, e.g. a year, except for the periodic increment to adjust for the inflation. A variant of this tariff is time-of-use pricing tariff (TOU). In TOU, the cost of electricity varies depending upon the time at which the electricity is used. For example, there are two distinct prices of electricity, one applicable from 9 am to 5 pm of weekdays, and one for all other times. The price for higher load period is set higher to encourage the consumers to shift their load to periods of lower demand. Although, more and more utilities are shifting toward TOU pricing from the older flat pricing model, this simplistic scheme fails to accurately reflect the actual situation of supply and demand in the market, resulting in an economic inefficiency.

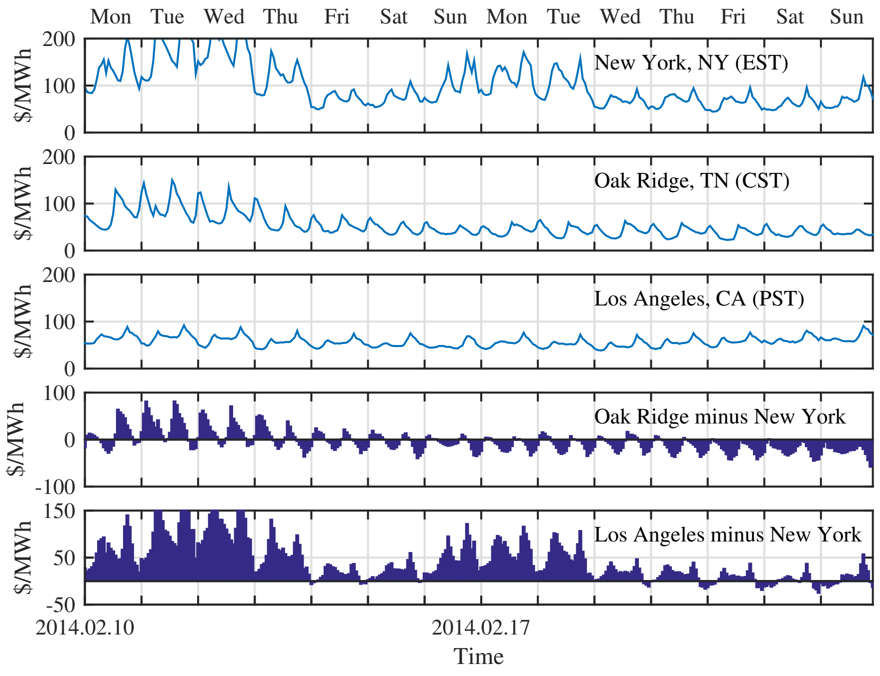


Figure 2.4: Electricity price fluctuations in day-ahead market for the period of 10th to 23rd Feb 2014 for three cities which lie in different time zones. For spatial comparison among cities, in the last two graphs, the time line is shown according to Eastern Standard Time (EST). Data sourced from [27, 117, 118].

Another attempt toward resolving this inefficiency, without necessitating a major infrastructure upgrade in metering, has been the introduction of, now ubiquitously prevalent, demand charge. Here the users are charged for the peak power consumption (kW), in addition to the total energy consumption (kWh). For example, see currently effective tariffs in three regions around California: [34, 134, 139]. The demand charge can be a substantial part of the overall electricity bill; cases where more than 50% of the overall cost is due to the peak power surcharge are not uncommon [6, 49, 59]. Demand charge is typically only levied on non-residential customers.

The other category is that of dynamic pricing tariffs. In these, the price of electricity is not fixed and can vary significantly depending upon the demand. For example, the price of electricity during hours of peak demand, such as after-

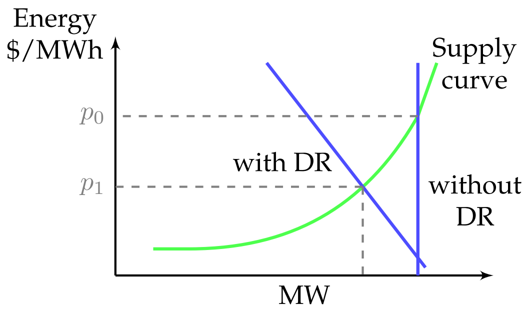


Figure 2.5: Elasticity in electrical demand can result in lower electricity prices.

noons in summer months, might be ten times more than the price during hours of low demand, such as early mornings. The peak timings also shift monthly and seasonally. Motivated by these price variations, large consumers of electricity (for example, industrial-scale consumers) try to schedule their workload intelligently according to the expected price of electricity. This might lead to substantial monetary gains. Under these tariffs, the market volatility is passed fully to the consumers of electricity. The end customers pay the same price as dictated by day-ahead or real-time electricity market [127] plus an offset for factors such as taxation, delivery etc. Mostly large data centers buy energy under contracts that fall into this category. However, in some markets such tariffs also being offered to the residential customers [36, 125].

2.2.4 Demand Response

Generally, in the electricity market, the total load on the grid at any instant, is considered to be inelastic, that means, it must be met at all times or else the stability of the grid might be jeopardized. Also, storing the electricity in any appreciable level at the scale of the grid is still not possible. This results in in-elasticity in demand. Such demand pattern is disruptive for the normal workings of an open market because increase in the price of commodity can not be compensated through decreased demand as shown in Figure 2.5. Consequently, the price of electricity rises very high at the time of peak usage. Producing even a small elasticity in demand can result in substantial cost savings. For instance, it has been reported that a small reduction of demand by 5% could have been resulted in a 50% price reduction during California electricity crisis in 2000-2001 [5].

In this context, demand response (DR) is a mechanism to introduce elasticity in the electrical demand. DR aims to alter the electricity usage by end-users from their normal consumption pattern in response to changes in the price of electricity. For example, the alteration of usage patterns can result from shifting some time-wise elastic activity either earlier or later in time, such as, delayed activation of an airconditioning unit [35]. This might result in temporary loss or degradation of service for the consumers. Sometimes this might not be tolerable for all consumers. Hence, most of the DR programs are voluntary for end-consumers.

Different mechanisms are employed to shape the electricity demand. These can be categorized according to the timescale they target [140, p.52]. For example, an arguably long term DR mechanism is to make electrical devices more energy efficient in response to rising electricity prices. The results for this action might start appearing on timescale of years. But commonly used DR mechanisms include time-of-use (TOU) pricing, demand charge and real-time pricing for electricity. The effectiveness of TOU pricing and demand charge are seen on the timescale of months, where as real-time pricing is a more short term solution. Another method to achieve DR is by directly controlling consumer's electrical devices. Although it is effective for peak control, the adoption remains low due to its intrusive nature and the infrastructure required.

DR has significant potential for cost savings through peak reduction. It has been estimated that a 5% reduction in U.S. peak electricity demand could produce approximately $35 billion in cost savings over a 20-year period [51]. This shows the economic potential for such a facility that can save the electricity in time of low demand and deliver it at the peak time. For this reason, electricity storage facilities, such as pumped hydro storage and battery based electrical storage system (BESS), are attracting attention. We tackle this problem in Chapter 4.

2.2.5 Environmental Penalty

Electricity generation remains one of the chief producers of environmental pollution and anthropogenic GHGs. According to a recent report from International Energy Agency (IEA), the energy sector is responsible for two- thirds of the total GHG emissions and 80% of CO_2, globally [76]. Although the renewable sources are gaining market share, particularly in Europe, their share in the United States in 2015 was 10% [152]. Similar figure has been reported on global level [76].

The reliance on fossil fuel for electricity production is the main cause of the environmental pollution. IEA also reported that global CO_2 emissions from electricity and heat almost doubled between 1990 and 2013, driven by the large increase of generation from coal mainly in developing economies [76]. Although, the percentage share of electricity generated through fossil fuel has decreased in these years, the absolute share has almost tripled owing to the increases in population and GDP in the developing economies specially in Asia and South America. It is believed the trend is likely to continue in the future as well.

Being one of the major users of electricity, ICT, in general, and data centers, in particular, have a substantial contribution in the environmental pollution and GHG emissions. Here we ignore the indirect contributions of ICT, for example in optimizing the electricity production processes, and consider only its direct contributions. According to recent studies, ICT products accounted for 4.6% of the total annual electricity consumption where data centers have been the fastest growing segment. Data centers use 85.6 TWhs annually in the United States alone. This represents approximately 2.2% of the total U.S. consumption [83, 155].

Moreover, the load profiles of the data centers are generally quite uneven as they are influenced by the local weather and the service request patterns of their customers. The change in local weather, such as ambient temperature, changes the air conditioning requirements of the data center. For instance, this can result in peaks in summer afternoons, perfectly coinciding with peak generated by the population's requirement of air conditioning. Similar scenario is repeated in numerous situations where DCs exacerbate the problem of peakiness in the electrical demand.

The peaks in the electrical demand from the grid cause an extra environmental penalty, as the peaks in demand are often served through fast-acting and inefficient diesel and gas generators that lack the sophisticated apparatus needed to clean the exhaust they produce. Also, it has been estimated that diesel generators used to serve the peaks, are responsible for a substantial fraction of the total nitrogen oxides released in the process of electricity production [162].

Peaks in the electrical demand also have another indirect effect on the environment. As the electricity prices rise during the peak times, this incurs a hefty financial penalty on the large-scale Internet services that need to purchase the expensive energy in this period for providing the service. Hence, they consume the budget that could have been utilized to purchase or produce the green energy.

Analogously, an even load profile results in a two-fold return. It reduces the environmental footprint, and, can be served more economically as compared to an uneven load profile as the supply curve is a convex function of required load. We analyze the problem of balancing the load profile in Chapter 4.

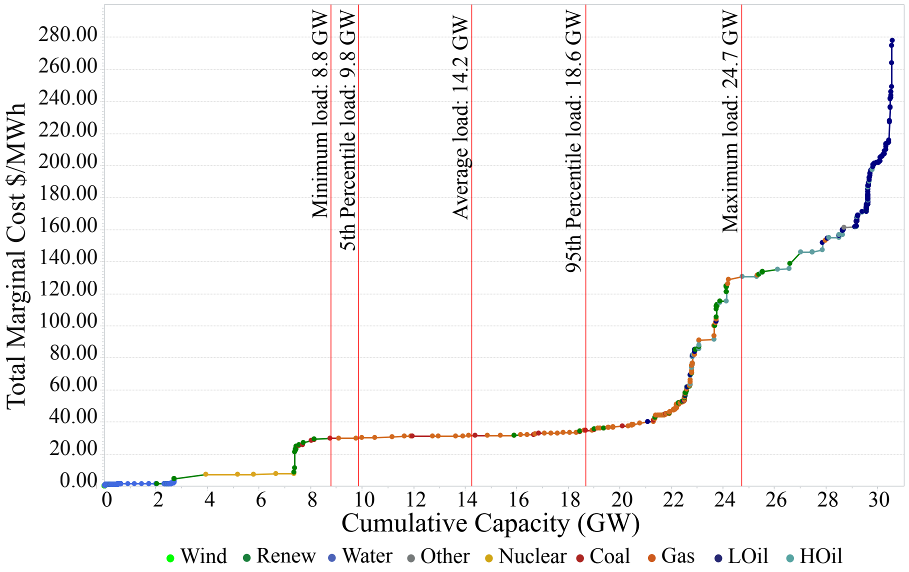


Figure 2.6: Market supply curve for NYISO as of 5th May 2010. [154].

2.2.6 Renewable Sources

Electricity can be produced using sources which naturally replenish themselves on a human timescale to negate environmental effects resulting from the burning of fossil fuels. For example, solar, wind, rain, geothermal, biomass and tidal sources can be exploited to generate energy. Among these, the ones that are seeing the quickest adoption have been solar and wind. World-wide investments into wind and solar sources of electricity have seen a substantial growth in the last couple of decades (see Figure 2.7). In 2014, out of USD 270 billion new investments in renewable energy, USD 250 Billion were in solar and wind [105].

As more investment and research money is poured into the wind and solar sources of renewable energy, the cost of energy production through these sources has been decreasing. However, owing to the capital costs of establishing these energy production facilities, the energy produced is still more expensive than that produced by the conventional sources that consume fossil fuel. According to a recent report by the U.S. Energy Information Administration for power plants entering in service in 2022, the levelized costs of energy produced by offshore windmills and onshore are estimated to be 146.7 USD/MWh and 56.9 USD/MWh, respectively. Similarly, the levelized costs for photovoltaic based solar energy production and thermal based solar energy production are esti-

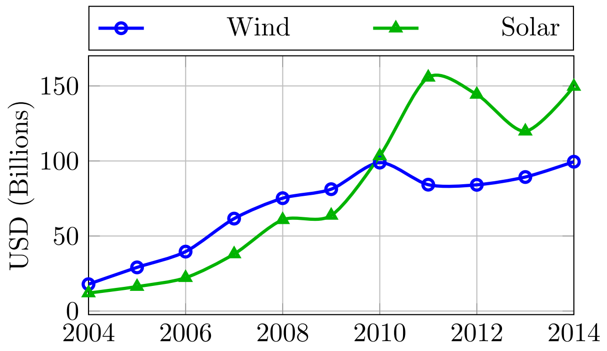


Figure 2.7: Worldwide investments in wind and solar energy capacities [105].

mated to be at 66.3 USD/MWh and 179.9 USD/MWh, respectively. In contrast, the fossil fuel based energy is expected to be produced at the cost of 57–58 USD/MWh [153]. These estimates include the tax rebates offered and penalties levied on different sources of electricity production due to their environmental impact, however, local renewable energy production facilities (e.g. rooftop PV panels) are not considered due to their small impact. Clearly, as people prefer the green energy over the fossil fuel based energy, with the growing user base, the capital costs will get amortized, increasing the feasibility of the green energy. In the future, we believe that the greener sources of energy will become ever more competitive in terms of their production cost, resulting in a rich mix of green and brown electricity options available to the end consumers to choose from.

To gain access to renewable energy, two main approaches are followed. Some businesses opt to build their own small-scale production facilities such as setting up rooftop photo-voltaic panels. The other option is to contract a specialized electricity producer who contributes the contracted amount of green electricity to the grid. In the case of first option, a connection to the grid is necessary due to the intermittent nature of wind and solar sources. In the second case, a mix of brown and green energy is bought from the grid depending upon a number of factors such as available budget, weather conditions, energy price etc. We tackle both these cases in context of maximizing the green energy usage in Chapter 3. We concentrate primarily on solar and wind energy in our analysis. However, the framework we propose is applicable to any source that has a limit on availability and has a non-zero cost.

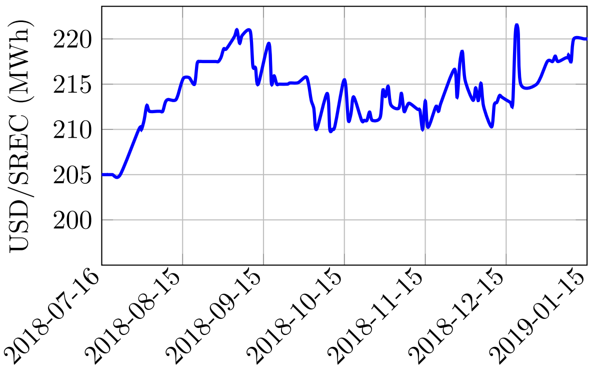


Figure 2.8: Solar REC price for New Jersey [53]

2.2.7 Carbon Market

In order to curtail environmental pollution a market based scheme was devised during UNFCCC meeting in 1997 at Kyoto, Japan, called Kyoto Protocol [148]. Under this scheme, each of the participating countries is allocated a specified quota of carbon credits. One carbon credit or emissions permit is considered equivalent to one metric ton of carbon dioxide (CO_2) emissions. Other names for carbon credits are Kyoto units and Certified Emission Reduction units (CER). Each participating government makes sure that net carbon emissions originating in their country do not exceed the allocated credits. Those countries that need to emit more CO_2 than the allocated credits, buy them from others who have excess and vice versa. These trades take place on open market basis among countries, which insures that emissions are reduced at that place where it costs the least to optimize.

Similar schemes exist at national levels which are often called **cap-and-trade** schemes. The government or a central regulatory authority allocates a limited number of credits for each administrative region, as per the cap for that region, to discharge specific quantities of pollutants per time period. Businesses with emissions above a certain level are mandated to hold the credits equivalent to their emissions. Since the total credits are capped for each region, the businesses that need to increase their emissions must buy credits from others willing to sell them. Also, the cap on credit insures that these have a value. In effect, the buyer is paying a charge for polluting, while the seller is rewarded for having reduced emissions. The largest of such schemes is European Union Emission Trading Scheme (EU-ETS).

U.S. being a nonparticipant in Kyoto Protocol, does not take part in the international emission trading. Lacking a nation-wide regulation, there are state-level regulations in place inside U.S. that focus on the main polluter i.e., electricity production and strive to enhance the use of renewable energy. Big consumers of electricity are required to buy renewable energy credits (RECs) in proportion to their electricity consumption. Renewable energy producers are allowed to sell one REC for every MWh of energy they contribute to the grid. These credits are traded on free market, hence their prices fluctuate on the basis of supply and demand, similar to electricity. Figure 2.8 shows the price variability for solar credits in New Jersey. Price of RECs can basically be seen as a surcharge that is to be paid on top of brown energy price to make it "green". The framework we develop to reduce the environmental footprint of data centers in Chapter 3 caters for this case implicitly.

2.3 Energy Storage System

Energy storage systems help capture the energy produced at one time to be used at a later time. There exist many systems for energy storage mainly differing from each other in the form in which they store the energy. For example, batteries store energy in the form of chemical energy, whereas hydro pumped storage systems store it in the form of potential energy by pumping the water to higher altitude. The choice of the most appropriate storage system depends on the application's charge and discharge ratings, the actual required energy storage and its daily operating cycle. Other considerations include economic and environmental issues.

For applications related to DR or peak shaving in electrical supply systems, the most important considerations are peak durations and conversion efficiency. Also, the terrain on which the facility is to be built plays an important role in the choice of a storage system. Presently, world-over the most commonly used storage systems are battery electrical storage systems (BESS) and pumped-storage.

In the following sections, we compare both these systems considering their feasibility for peak shaving applications.

2.3.1 Pumped Storage System

A pumped storage system basically consists of two nearby water reservoirs at different altitudes. At the off-peak time, the system consumes electricity to pump water up to the higher reservoir, while at the peak time the reverse happens where the flowing water moves an alternator to produce electricity.

Worldwide, the pumped-storage hydroelectricity (PSH) scheme is the most commonly used method for energy storage in the grid. The Electric Power Research Institute (EPRI) reports that PSH accounts for more than 99% of grid storage capacity globally, representing around 127,000 MW [48]. PSH reported energy efficiency varies in practice between 70% and 80%, with claims of up to 87% [131].

Although PSH dominates the energy storage systems, traditional PSH has limited capacity for expansion. Sites that fulfill the requirements needed for such systems, are neither easy to find, nor can they be expanded as per the growing demand. Creating such system where and when needed involves exorbitant financial and environmental costs, including the damages to the local flora and fauna.

2.3.2 BESS

The second most common form of energy storage system is a battery electrical storage system (BESS). In a BESS, the central component of the system is a rechargeable battery that is used to store electrical energy in the form of chemical energy. Over the years, the capabilities of batteries have increased and lead-acid technology is gradually being replaced by more environmental friendly sodium-sulphur (NaS), nickel-cadmium (NiCd) techniques. Also, the former provides better short-term power ratings and lower maintenance requirements. An example of such systems is the battery system in Golden Valley, Fairbanks, Alaska [57]. This system is capable of providing 27 MW for up to 15 minutes with an expected life time of 20 years. Installation cost of this project was $35 million. However, the worlds biggest battery storage system is going to be based on lithium-ion (Li-ion) technology, capable of 400 MWh, and, will replace a gas-based 'peaker' plant [3]. A comparison of different battery technologies along with their cycle efficiencies is shown in Table 2.3.

The high installation price of the battery electrical storage systems (BESS) has hampered its adaptivity. The biggest contribution in the price of BESS comes from the battery. However, keeping the upcoming trends about electric cars in mind, this can potentially change. On the average, a car remains parked for 23 hours daily. Normally, while in parked state, the electric vehicles are connected to the grid for charging. During this time, their batteries can be used to shave off the peaks occurring in the grid, thereby easing the load on grid and curtailing the net payable bill [65, 80]. This development makes BESS and associated techniques a promising aspect for the future.

Table 2.3: Cost and efficiency of various battery systems [166, 137]

Battery System	Cycle efficiency	Self discharge (per day)	Life cycles	Capital cost (USD/kWh)
Lead-acid	70–90%	0.1–0.3%	500–800	150–200
NiCd	70–90%	0.2–0.6%	2,000	350–1,000
Li-ion	>90%	0.1–0.3%	1,000–10,000	250–1,800
NaS	87%	~20%	2,500	300–550

2.3.3 Other Technologies

Quite a number of other technologies has also been used for energy storage. Examples include flywheel energy storage systems (FES), pressurized air storage system and heated water storage. Particularly, the FES has been used in a number of facilities around the world. It does not hold much market share currently but has high potential due to its long life, low maintenance requirement, fast charging time and immunity to temperature variance. However, the technology has still not matured [4].

2.3.4 Suitability

In order to assess the suitability of a technology for introducing the flexibility of load shifting in the electrical grid, a number of factors need to be considered. Important factors include the initial investment, environmental effects, conversion efficiency, maximum capacity and response time among others. Pumped hydro storage and pressurized air storage are two most commonly used methods for energy storage but they suffer from limitations as they require appropriate geological features which are not found commonly. Also the available geological features decide the maximum achievable capacity. Due to this specific nature of storage facility, these can not be replicated at any site as per requirements at hand [47, 73]. In comparison, BESS does not have any such specific requirement for terrain and the maximum capacity can be scaled as required. Due to these advantages it is more feasible for small to medium scale peak reduction applications such as for a single industrial unit or a data center. However, the solution we develop in this dissertation (Chapter 4) is agnostic to the technology used for energy storage.

With the increasing share of renewable energy sources in the grid, and, owing to the intermittent nature of the renewable sources, a more general application

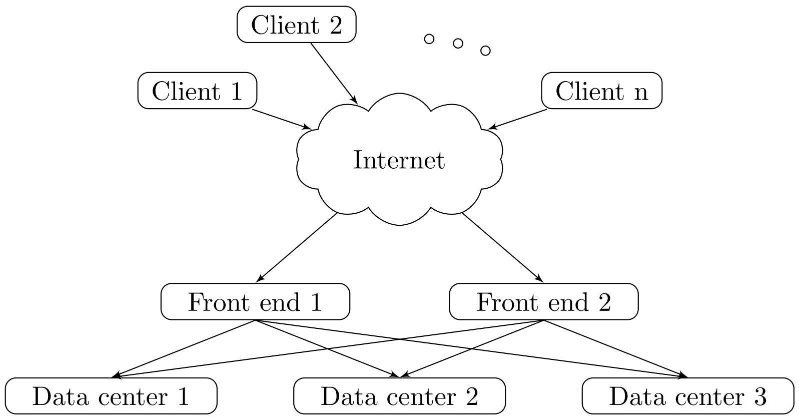


Figure 2.9: Typical Internet service architecture. Data centers 1, 2 and 3 are the back-ends. The front-end devices are typically spread across multiple data centers.

of grid-integrated storage is load shifting instead of peak shaving, that we restrict ourselves to, in the scope of this dissertation. Peak shaving is a special case of load shifting. However, the important factors mentioned above for suitability of a technology remain the same for both cases.

2.4 Internet Services

In this section, we take a look at the general architecture of the multi-DC based Internet services and the factors that effect their operating costs as well as their impact on the environment.

2.4.1 Architecture

Internet services are generally provided through a group of geographically distributed data centers that work in tandem to provide a multitude of services with different service level agreements (SLAs) and with high reliability to the globally distributed clientele consisting of millions of users, through sharing the workload amongst them. Typical architecture of an Internet service is shown in Figure 2.9. Front-ends handle incoming requests from the clients and redirect them to the appropriate back-end that can serve them. After this redirection, the client directly interacts with the back-end for all the subsequent requests of the session. Although, logically the front-end and the back-end are two different entities, physically both might be co-located in the same data center. The user content and the data important for running the service, are kept at the back-end and are mirrored at multiple locations. The cost for this redundancy is the extra state-coherence traffic that is generated to keep all copies of the data in sync with each other. To avoid this extra state-coherence traffic, typically the data is not mirrored at more than 2 or 3 locations.

Typically the term 'request' is used when the duration of computation required is in the range of a couple of seconds, e.g. a search request to Google. On the other hand, when longer computation is required, e.g., in the range of several minutes or hours, it is called a 'job', e.g. indexing jobs for a search engine. In Chapter 3, we use the work model of requests, whereas in Chapter 5 we follow the work model of jobs.

2.4.2 Important Aspects

Internet services are complex and huge systems with a lot of possibilities of optimizations in terms of energy usage and environmental footprint reduction. Considering their distributed nature and the spatial variability of electricity prices, there exists a potential to save electricity cost as well as reduce their environmental impact, if an appropriate load distribution strategy is employed. However, making a right load distribution decision is not trivial and the following important factors can potentially impact the outcome.

Varying prices of electricity

The prices of electricity, both green and brown, vary temporally and geographically. Although many electricity supply companies have shifted to time-varying prices for bigger consumers, some still follow the fixed-priced energy contracts with constant cost per energy unit. Moreover, actual processors consume substantially more energy during processing of requests than during idling. This implies that the variance in energy used for processing the requests, i.e. the active energy component, is a significant fraction of the total energy [127]. An appropriate workload distribution amongst the member data centers that caters for this variability effectively can reap substantial financial gains.

Multiple services with different SLAs

Data centers are either dedicated to only one client, e.g. Facebook's data centers or they service more than one client such as Amazon's EC2. In general, it can be said that data centers are expected to offer more than one service to more than one client, under different SLAs with varying degrees of QoS guarantees and with different pricing. Majority of the previous work has focused on a single data center providing a single service. The impact of multiple SLAs and multiple services being offered by a group of data centers has often not been considered.

Session-based services

The services offered by the data centers are either session-based or stateless. In the case of session-based services, not all requests can be arbitrarily routed to any data center. The requests belonging to one session must either be served by the same data center, or the context transfer has to be considered.

Communication latency due to geographical distance

The distance between the data centers and the front end causes additional delay in serving the routed requests. Previous studies [127] have found that the delay is correlated to geographical distance. This delay should be considered when distributing the requests otherwise the SLA might be violated.

Energy cost of sleep-wake transitions

Putting a server in a data center to sleep or bringing it back for executing is not free in terms of energy consumption. Sleep-wake transitions incur additional energy costs that need to be catered when deciding to route the incoming load. By selecting a server that is already in operation, extra overhead caused by the transition can be saved.

Energy consumption of infrastructure

Data centers do not only consist of servers. There are also other non-computing devices as well like networking switches, routers, cooling devices and lighting. The average energy consumed by these devices is almost the same as the energy consumption of processors (typical PUE=1.7 [74]). These devices contribute substantially toward the environmental footprint of a data center and their effect must be considered.

Energy sources and caps

There are three basic sources of energy in each data center: 1) green energy harvested through the local resources (like a local wind farm), 2) green energy bought in form of carbon credits, and 3) brown energy. Many data centers nowadays include some local facilities to produce green energy, e.g., [8, 149]. The energy produced by the local facilities is audited and converted to carbon credits [116] which can be used just as other credits bought at local markets. The price for these credits has to be paid in the form of initial expenditure on the renewable energy facility. Local wind or solar farms can produce limited supply of green energy and its maximum production cannot exceed its rated output. This can be considered as a limit on availability.

2.5 Parallelized Workloads and Dark Silicon

The amount of data that DCs need to process each day is increasing at a rapid rate, thereby straining the computing resources required for processing it. It has been estimated that the total digital information will increase 300 times between 2005 and 2020, i.e., from 130 exabytes to 40,000 exabytes [54]. In this era of data explosion, DCs employ ever evolving parallel processing mechanisms to compute and deliver results in a timely manner. In view of these requirements, the hardware setup employed in the DCs is relatively simple and comprises of a large number of common-off-the-shelf (COTS) PCs that are interconnected through a high-speed data link. However, the software part has been evolving rapidly with the aim of deriving maximum performance out of all the available servers by utilizing them in parallel. A number of software frameworks have appeared for this including MapReduce [42], Hive [145] and Spark [161], among others.

These software frameworks share the salient architectural features among them, in that, there is a central 'master' node, that divides and distributes the work among the 'worker' nodes and keeps track of their progress through the periodic 'heartbeat' messages. Scheduling and error handling is the job of the master node. Naturally, to gain performance through full parallelization, the compute-workloads must not have any data interdependency though. This is the most common scenario for batch processing of data in DCs ('jobs', as mentioned in Section 2.4.1).

Just as DCs have thousands of individual servers connected through a high-speed intranet, many-core processors have thousands of individual processing cores sharing an on-chip network [20]. The architectural similarities between these two fields are quite apparent. Hence, the facts that Intel named their prototype 48-core processor [72] Single-Chip Cloud Computer (SCC), and, DCs are seen as "a large multiprocessor" by researchers in the field of scheduling [95].

An important topic in the domain of many-core chips is that of power management. Due to the limitations imposed by power provisioning circuitry within a chip and heat removal apparatus from many-core chips, a significant fraction of these chips must always remain inactive [50]. This phenomenon has been termed as 'dark silicon'. The gap between the total available silicon on the chip and the amount that can be turned on simultaneously is growing, and it is widely believed that increase in the performance of modern processors will originate from parallelization instead of higher frequency of operation [144, 2]. As the power is becoming a scarce resource within a chip, the decision about where and how much power to use is becoming increasingly important [120, 128, 21] and this trend is likely to continue.

Similar to many-core chips, DCs also have a pressing need to optimize their power consumption because of binding legislation and environmental concerns. Owing to the architectural similarities between the DCs and many-core chips, both the DCs and the many-core chips can be modeled using very similar power consumption abstractions for computation of timing critical workloads. Exploiting these similarities, we focus only on the problem of power management within many-core chips but our results are generally valid for wider applicability in data centers.

2.6 Related Work

In this section, we discuss relevant works in the area of minimizing environmental footprint of data centers through maximizing the usage of renewable energy, i.e., the main mechanism we use to manage environmental impact and energy in data centers. Also discussed are the approaches that aim to conserve energy in DCs and cost management strategies by exploiting electricity price variance.

2.6.1 Greening the Data Centers

DCs and Internet services being major electricity consumers in the IT sector, have been focus of a lot of research to make them environmentally friendly. This can be divided into three main categories:

Energy conservation

These studies aim to decrease the energy consumption of a DC, where decreased environmental footprint is basically a by-product. Examples include [30], [67], [159]. Mostly, these aim to optimize a single DC. For example, Wang et al. [159] present a scheme to reduce power consumption while fulfilling the generalized SLAs within a single DC. The solution we present builds on top of these schemes as we aim for multiple DC optimization and single DC optimization is part of that.

Electricity cost management

These studies are nearer to our approach. The key difference between this category and the previous one is that, here, multiple and geographically distributed DCs are considered. Examples in this category include [127], [92], [103], [102]. Qureshi et al. [127] were the first to tackle the problem of cost minimization by exploiting the geographic variance of energy prices but they do not consider the carbon market dynamics. These are also not considered in [92] and [102].

Utilizing the green energy

This is a relatively new direction with only few initial studies e.g [163], [138], [130]. Our approach falls into this category. [163] present how to maximize the use of environmental friendly green energy to power the servers in DCs, while maintaining the average response time for incoming requests. However, since they use queuing theory to model the service provision, it can not handle generalized SLAs, for instance, in the form of percentile guarantees; e.g. an SLA of the form (L, P) where P percent requests must be satisfied within time L. The same argument also applies to the limitations of the research in [130], [86], [138]. Moreover, [130] and [138] do not consider time-varying workloads, multiple services, or market interactions. Stewart and Shen [142] also focus on minimizing the environmental penalty by reducing the use of brown energy. They use a model in which Internet service providers own the renewable energy farm. *In contrast, we consider the more general case where the renewable energy* *can be locally produced or bought in form of RECs by the commercial producers and contributed to the grid.* Le et al. [85] are more thorough in their approach towards the problem. They focus on cost reduction by exploiting the distributed nature of DCs for dynamic request dispatching while maintaining SLAs. They are the first ones to consider carbon interactions. Our approach has two main differences from [85]: Firstly, we aim to maximize the green energy usage within budgetary constraints as opposed to maximizing profits within brown energy cap. Secondly, in our solution, we divide the optimization problem to smaller parts: one to be solved by each data center and the other for the front end. This helps two folds: 1) we can include more factors to model energy consumption, including the infrastructure for networking, computation, cooling devices, etc., and 2) the optimization problem can be solved more frequently because of the reduced complexity at the front end. The latter also results in a shorter horizon for energy price and traffic predictions, making predictions more accurate.

2.6.2 Peak Shaving

The peak shaving problem in utility networks is a special case of demand response, where the aim is to reduce the highest peak in the power consumption. In the scope of this dissertation, we restrict ourselves to the peak shaving problem only. Theoretically, it is a special case of a more general load shifting problem. But practically, peak shaving problem has a bigger financial impact in the todays market-place than its general version due to still low penetration of renewable energy sources (10% in 2015 in the Unites States [152]). The reason for peak-shaving problem having a bigger financial impact is the ubiquitous presence of 'demand charge' in almost all electricity tariffs offered in the United States. Demand charge is a charge levied on non-residential customers according to the peak power they draw from grid, in addition to the bill they receive for total energy usage. There exist some results in the literature for peak minimization problem, such as [24, 88, 119, 11, 151, 12, 94, 1]. These works can be divided into two main categories on the basis of the pricing model they employ, as follows.

In the first model, the electricity provider controls the electrical load of the consumer and turns it off during the peak load times. The maximum duration for which the load can be turned off is agreed via contract. This approach is adopted in [88, 1].

In the second pricing model, the demand response is encouraged through controlling the pricing for the peak. This is still the most common approach to reduce the demand charge. There have been works in this direction such as [94, 11, 119].

In both of these cases, the peak demand can be decreased by introducing a battery in the system. To determine the most appropriate battery size, and to measure the effectiveness of a control algorithm, an important milestone is to be able to quantify the peak load that can occur in presence of that battery and that control algorithm.

In our approach, presented in Chapter 4, we devise a methodology for quantifying the peaks. Fundamental difference between our work and the previous works such as [1, 88] is that we consider the load as inelastic which must be fulfilled at the time of demand. As mentioned in the beginning of this section, due to still low penetration of general load shifting possibilities in the contemporary electricity market, this assumption is more in line with current market practices. However, this must change in the future to ease the transition toward renewable sources.

Among the works that follow the second pricing model, [11] offers online algorithms for shaving peaks and present worst-case competitive ratio analysis for these. This work aims to minimize the peak through specific algorithms whereas we aim to quantify them for any given monotonic algorithm. They also do not consider the practical aspects of inefficiencies in batteries. More recently, [94] tackle the same problem using an approach of optimal control in context of data centers. Their focus, however, has been on decreasing the peaks on the average, whereas, for demand charge, the electricity providers consider the highest peak only.

2.6.3 Internal Optimizations for Data Centers

Managing the power consumption within a DC is similar to the management of power consumption within a many-core chip. As discussed earlier (Section 2.5), we focus on the equivalent problem of power and energy management within many- and multi-core chips instead.

A primary concern in many-core chips of the dark silicon era is to decide how many and which cores to turn on to successfully compute a timing critical workload. An error in the proper selection of a subset of chip might result in the chip overheating and negatively affecting its availability later on. This might result in real-time tasks missing their deadlines. For this, we propose a peak power-management scheme for many-core chips in Chapter 5. In this section, we present the relevant works in this regard.

In past, much research has focused on power, energy and thermal management for multi-core systems [52, 158, 77]. In [52], the global thermal-aware scheduling of sporadic tasks is analyzed to minimize the peak temperature using DVFS as a knob. In [77] an optimal procrastination interval for each task with real-time constraints is derived to minimize the energy consumption. Likewise, [158] proposed a scheduling analysis to minimize the energy consumption under thermal constraints. In these works the scheduling decisions are intended to reduce the energy or temperature by controlling the average power consumption. Therefore, these cannot be effectively modified to control the *peak* power consumption to remain within the thermal design power (TDP) constraint, as is the case in our work. Moreover, we cater for heterogeneity of cores and tasks as well.

The work in [114, 129, 135] focuses on *maximizing performance under a power constraint, e.g., TDP*. In [114], a control-based framework is proposed to obtain the optimal trade-off between power and performance of asymmetric multi-core systems under a specific power budget (TDP). The work in [129] exploits the process variations between the cores in a homogeneous multi-core system to pick the more suitable cores for an application to improve performance. Their results show that the performance efficiency can be increased along with the increasing dark silicon area, due to the proportional increment of the process variations. However, in both of these works [114, 129], the performance is not guaranteed, making it unsuitable for real-time tasks. In our work, we tackle the dual problem, in which we focus on *minimizing the peak power consumption* while considering the schedubility of the *hard real-time tasks as constraints*, i.e., delivering a guaranteed performance. In [135], Sartori et al strive to boost the performance while guaranteeing that power consumption of the chip does not shoot beyond a threshold. However, the performance is again not guaranteed. In contrast, we provide a simple, polynomial time, online admissibility test for hard real-time task sets.

Another work in high correlation to ours is [91]. In it a new scheduling algorithm is developed that minimizes the peak power consumption for real-time tasks. However, the complexity of the method is so high that it can only be used for offline design. In comparison, we present polynomial time algorithms, that can be used for online scheduling.

3 Maximizing Green Energy Usage

In this chapter we focus on the problem of massive environmental footprint of data centers and propose mechanisms for its reduction. We propose and evaluate a framework for optimization-based request distribution that enables multi-data-center Internet service providers to manage their environmental foot print and energy costs, while respecting their SLAs. The proposed framework allows services to take full advantage of the geographical locations of distributed data centers. Specifically, it exploits data centers that pay different electricity prices (the pricing scheme can be either fixed, TOU or dynamic as explained in Chapter 2), data centers located in different time zones and data centers that are powered by renewable energy sources. At the same time, the framework considers the existing requirements for high throughput and availability. Based on the framework, we propose a methodology for online load distribution decisions so that the green energy utilization is maximized while remaining within the allocated budget. We propose an optimization-based policy which uses mathematical optimization algorithms, time series analysis for load prediction, and statistical performance data from data centers. We also propose a greedy heuristic designed with the same goals and constraints as the optimization-based policy.

We evaluate these policies using real electricity prices and actual traces from Wikipedia. Our results show that the optimization-based policies can account for substantial reduction in environmental footprint by a marginal increase in costs through intelligently leveraging time zones and hourly electricity prices.

3.1 Overview

In this chapter, we focus on the minimization of the environmental footprint of DCs under the budget constraint and the generalized SLAs, including percentile and average response time guarantees. We present a software optimization strategy to dynamically dispatch the incoming requests from the central hub of an Internet service provider (such as Google or iTunes) to the distributed DCs. This is a multifaceted optimization problem, and many important aspects need to be considered in such a setting, as detailed earlier in Section 2.4.2.

In our approach, we divide the problem into two subproblems, one to be solved individually by each DC and the other by the central dispatching hub. We present a practical solution encompassing all the energy-consuming components in a DC. That includes the energy consumption from the infrastructure for networking, computation, and cooling devices. Our solution is flexible enough to be applicable to DCs consisting of heterogeneous servers as well as able to accommodate different SLAs. We evaluate this with real-world workload traces from Wikipedia [150] and varying electricity prices from different regions in USA obtained from NYISO [117]. We show that our greedy algorithm is able to find a good solution to this optimization problem by relaxing the budget constraint and can be easily adopted in data centers.

3.2 System Model

In this section, we formalize the system model and discuss how we handle the challenges discussed previously.

We consider a network of N DCs as shown in Figure 3.1. A central dispatcher receives all the requests and dispatches them to the N DCs according to a *to-be-designed* dynamic load balancing strategy. The data centers share a common operational budget for a *budgeting period* (e.g. a month). The budgeting period is divided into smaller *control periods* (e.g. an hour). The network of data centers collaboratively provides the total required service Λ_b (the request rate) in a control period b.

Energy Sources. We consider that each DC has Z different energy sources to choose from. These can be different forms of green or brown energy sources. The cost to buy a unit (\$ per kWh) from the j^{th} energy source in DC i during control period b is $C_{b,i,j}$. We assume that $C_{b,i,j}$ is time varying. Importantly, fixed-cost energy contracts are just a special case of this more general setting. DCs with local green energy production facilities have to bear the initial investment and continuous management costs for such facilities. These costs, amortized over time, can be considered as the price of green energy.

When one unit of energy (kWh) is purchased from the j^{th} energy source in DC i, the associated penalty is defined as $\phi_{i,j}$. In general, green energy sources have none, while brown energy source has a positive penalty.

The availability of renewable energy and carbon credits in the market depends on the weather conditions and the cap set by the legislation authorities. Availability affects the price of energy and the cap enforces an upper limit. We assume the j^{th} energy source in all DCs to be limited to a maximum usage of L_j in the current budgeting period.

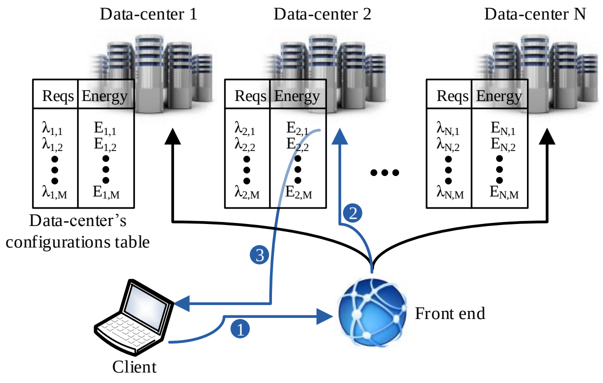

Data-center 1
Data-center 2
Data-center N
Reqs
Energy
$\lambda_{1,1}$ $E_{1,1}$
$\lambda_{1,2}$ $E_{1,2}$
$\lambda_{1,M}$ $E_{1,M}$
$\lambda_{2,1}$ $E_{2,1}$
$\lambda_{2,2}$ $E_{2,2}$
$\lambda_{2,M}$ $E_{2,M}$
$\lambda_{N,1}$ $E_{N,1}$
$\lambda_{N,2}$ $E_{N,2}$
$\lambda_{N,M}$ $E_{N,M}$
Data-center's configurations table
1
2
3
Front end
Client

Figure 3.1: Architectural overview of a network of N data centers with a typical route for a request and its reply

Service Level Agreements. DCs offer multiple services to multiple clients under different SLAs. This factor can be incorporated by dividing each DC into smaller cells to cover all the services that should be provided through the DCs. Each cell is considered as an individual DC. However, it is not mandatory that each DC covers all the services, due to the following reasons: 1) The overhead for maintaining the coherence of the states is larger than the performance gains [87]. 2) Not all clients are geographically suitably located to be served by some of the DCs because the communication latency is correlated to the geographical distance [127]. Therefore the clients can be statically assigned to a subset of DCs. Please note that SLAs that we consider are only within the premises of service providers. Our SLA can be combined with Internet QoS approaches to extend the guarantees all the way to the users' sites [164]. For the rest of this chapter, we only present how to deal with one SLA for the simplicity of presentation.

Session-based services. Incoming requests from the clients are distributed by a central dispatcher. We assume that once a request has been routed, the reply comes directly from the corresponding DC. If it is a session-based service, all the further correspondence is directly with the DC which the first request in session was assigned to. We assume that the front end is not part of the routing process after the initial decision, hence does not cause any additional latency.

DC Configuration Table. Every DC requires some energy as an input to provide some service as output. The required energy consumption depends upon the service requirements as well as the hardware and infrastructure configurations of the DC. This behavior can be captured in a table for energy requirement versus the maximum service (in terms of the request rate) in a DC under the specified SLA.

We consider DCs with discretized service levels, and each level has its required energy consumption in a control period. Every DC has up to M different energy usage levels (configurations) to choose from. Each energy consumption level corresponds to a particular maximum satisfiable service requirement. A DC i, in its k^{th} configuration uses $E_{i,k}$ kWh of energy to satisfy $\lambda_{i,k}$ service requirement, under the given SLA. In the second step, using 1) these tables for all participating data centers, 2) the workload distribution policy, and, 3) total service required at any instant, the energy required at each data center can simply be looked up in the respective table.

Naturally, the question arises how to construct such an energy-vs-service table for each DC. To this end, there exist approaches for considering the energy consumption of the servers under an SLA for a DC, e.g. the methodologies in [64] or [32]. We assume, the energy consumed by infrastructure is also part of the total energy consumption $E_{i,k}$. *The DC configuration table forms the basis of a very general solution* as it can include the energy spent on cooling, the energy consumption of network equipment, the hardware heterogeneity and various settings of SLAs. It has potential to capture most of the relevant aspects of a DC with a selectable granularity.

Another important aspect is the energy cost for the off→on transitions of the servers in the DCs. We assume that the entries in a DC configuration table already include the worst case energy requirement for such transitions. Hence, we do not explicitly include them in the model. Since the transition only occurs once (∼1 min [87]) per control period (1 hour in our model), i.e. turning the required servers on at the beginning of every control period, adding such worst case energy requirements does not increase the actual energy consumption significantly.

For notational brevity, if the available energy configuration of data center i is m and $m < M$, we define $\lambda_{i,j} = \lambda_{i,m}$ and $E_{i,j} = E_{i,m}$ for $m < j \leq M$. Without loss of generality, with respect to k, we also assume that $\lambda_{i,k}$ is non-decreasing and $E_{i,k}$ is non-decreasing as well. We assume that the first entry $\lambda_{i,1}$ in the data center configuration table for DC i is 0. The corresponding energy consumption $E_{i,1}$ may be 0 when the infrastructure and the hardware do not consume any energy when the DC is not used in the control period. However, practically, $E_{i,1} > 0$ and represents the energy cost of network infrastructure and other equipment, e.g. lighting, etc. In essence, it is an offset that can be added to all the entries of the configuration table.

Discretization. Commonly available processors on the market which are employed in the servers in the DCs are designed to operate on multiple discrete frequency (and corresponding voltage) settings [37]. The performance of the processor increases when operating at a higher frequency and vice versa, with the cost of higher performance being higher power consumption. Selection of the appropriate frequency setting for each processor is controlled by the OS and is normally decided on the basis of the total load on the system. DC-wide all the processors can be selected to run at any of the allowed frequency settings. Since the processors are only capable of operating at discrete frequency (and hence discrete power) settings, their combination, i.e. a DC, also has discrete power consumption points. To reflect this situation accurately we model the DC through a discretized table of energy vs service. This forms the basis of an integer optimization problem that we present in the following section.

3.3 Problem Definition and Future Prediction

3.3.1 Problem Statement

The objective is to *minimize the total environmental penalty* in the current budgeting period while satisfying the *service requirement* with the quality of service (QoS) as contracted in the SLA, without exceeding the *total budget S* with the time varying energy prices. Each DC can choose a fraction of the total required energy in the period from any of the available sources. The optimization goal is to select an index k_i with $1 \leq k_i \leq M$ for DC i such that the total environmental penalty is minimized under the service requirement constraint $\forall b \; \sum_{i=1}^{N} \lambda_{i,k_i} \geq \Lambda_b$ and the budget constraint.

Summarizing this,

i, j, k, b	=	Indices used for DCs, energy sources, configurations and control periods
N	=	Total number of DCs
M	=	Maximum number of configurations per DC
Z	=	Maximum number of energy sources
B	=	Maximum number of control periods in a budgeting period
L_j	=	Maximum energy availability from j^{th} source for all DCs combined (kWh)
S	=	Total allowed cost budget for all DCs ($)
$E_{b,i,k}$	=	Energy required at DC i for k^{th} configuration during b^{th} control period (kWh)
$\phi_{i,j}$	=	Penalty associated with j^{th} energy source in i^{th} DC (kg of CO_2. Here, CO_2 is equivalently replaceable by any other GHG or the sum of all GHGs, without loss of generality)

$C_{b,i,j}$	=	Cost of j^{th} energy source in i^{th} DC during the b^{th} control period (\$ per kWh)
Λ_b	=	Total service required during the b^{th} control period
$\lambda_{i,k}$	=	Service provided at DC i's kth configuration
$x_{b,i,j}$	=	In i^{th} DC, portion of j^{th} energy source to fulfill the energy requirement during the b^{th} control period
$y_{b,i,k}$	$\in$	$\{0,1\}$ for all b, i, k. Binary decision variable. 1 if k^{th} configuration in b^{th} control period in i^{th} DC is selected to be used, else 0.

With these symbols, the optimization problem can be formulated as follows:

$$\text{Minimize: } \sum_{b=1}^{B}\sum_{i=1}^{N}\sum_{j=1}^{Z}\sum_{k=1}^{M} y_{b,i,k}\cdot E_{b,i,k}\cdot x_{b,i,j}\cdot \phi_{i,j} \tag{3.1a}$$

$$\text{such that: } 0 \le x_{b,i,j} \le 1, \quad \text{for all } b, i, j \tag{3.1b}$$

$$\sum_{j=1}^{Z} x_{b,i,j} = 1, \quad \text{for all } b, i \tag{3.1c}$$

$$\sum_{i=1}^{N}\sum_{k=1}^{M} y_{b,i,k}\cdot \lambda_{i,k} \ge \Lambda_b, \quad \text{for all } b \tag{3.1d}$$

$$\sum_{b=1}^{B}\sum_{i=1}^{N}\sum_{k=1}^{M} y_{b,i,k}\cdot x_{b,i,j}\cdot E_{b,i,k} \le L_j, \quad \text{for all } j \tag{3.1e}$$

$$\sum_{b=1}^{B}\sum_{i=1}^{N}\sum_{j=1}^{Z}\sum_{k=1}^{M} y_{b,i,k}\cdot E_{b,i,k}\cdot x_{b,i,j}\cdot C_{b,i,j} \le S. \tag{3.1f}$$

These can be restated as:

(3.1a): Minimize the sum of environmental penalty in all DCs in a budgeting period, such that

(3.1b): usage of any energy source in a DC in any control period cannot be more than total energy requirement for that data center in that control period,

(3.1c): sum of all the proportions from all the energy sources should satisfy the energy requirements of the DC,

(3.1d): provided service should satisfy the required service for all control periods, (3.1e): usage of any energy source cannot exceed its availability in the market, and

(3.1f): the sum of the costs occurring at the DCs should remain within the overall budget.

3.3.2 In-feasibility due to Unknown Future

A solution to the problem detailed in Equations (3.1a)–(3.1f) will result in the optimal reduction in environmental penalty. However, to solve this, we need Λ_b and $C_{b,i,j}$ for all future control periods. This is, however, not possible. Electricity prices change on hourly basis and the horizon for "certain" knowledge spans only an hour in the future. Similarly, as service requests follow long term (monthly) and short term (hourly) trends (see Figure 3.3), good enough predictions are possible only for an hour in advance. Due to these factors we transform the problem to maximize the usage of green energy within a *single control period*. The problem can be modified as follows for a control period b, where $1 \leq b \leq B$: (with modified set of old symbols which belong only to a single control period)

$$\text{Minimize: } \sum_{i=1}^{N}\sum_{j=1}^{Z}\sum_{k=1}^{M} y_{i,k} \cdot E_{i,k} \cdot x_{i,j} \cdot \phi_{i,j}, \tag{3.2a}$$

$$\begin{aligned}
\text{such that: } \quad 0 \leq x_{i,j} &\leq 1, \quad \text{for all } i,j && (3.2b)\\
\sum_{j=1}^{Z} x_{i,j} &= 1, \quad \text{for all } i && (3.2c)\\
\sum_{i=1}^{N}\sum_{k=1}^{M} y_{i,k} \cdot \lambda_{i,k} &\geq \Lambda, && (3.2d)\\
\sum_{i=1}^{N}\sum_{k=1}^{M} y_{i,k} \cdot x_{i,j} \cdot E_{i,k} &\leq L_j - L_j^{b-1}, \quad \text{for all } j && (3.2e)\\
\sum_{i=1}^{N}\sum_{j=1}^{Z}\sum_{k=1}^{M} y_{i,k} \cdot E_{i,k} \cdot x_{i,j} C_{i,j} &\leq \psi(S - S^{b-1}). && (3.2f)
\end{aligned}$$

here,

- ψ = function for budget distribution. The input for this function is the total remaining budget at the start of b^{th} control period, i.e. $S - S^{b-1}$. We

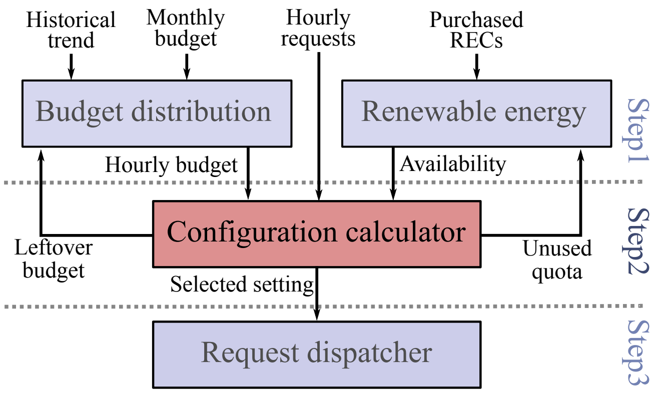


Figure 3.2: Outline for modified methodology

denote this by Δ in the rest of this paragraph for the ease of readability. $\psi(\Delta)$ can be as simple as distributing the remaining budget equally on remaining control periods (i.e., $\frac{\Delta}{B-b+1}$), or, it can be complex enough to include the predictions of traffic and pricing. However, it must always satisfy $\psi(\Delta) \leq \Delta$ to insure that the budget constraint is satisfied.

- L_j^δ = Used-up quota of energy availability for j^{th} type of energy up to δ control period, where $L_j^0 = 0$.
- S^δ = Budget consumed in the past for control periods up to δ with $S^0 = 0$.

Henceforth, we tackle the problem of greening the DCs as per Equations (3.2a)-(3.2f) i.e., according to the methodology shown in Figure 3.2. For every control period, we first calculate the budget on basis of traffic forecast. Predictions based on historical information or other prediction models, e.g. [25] or [156], can be adopted and an error buffer can be planned depending upon the prediction accuracy to cater for uncertainties. In the second step a load balancing strategy has to be designed for the data centers under the calculated budget constraint and the Λ constraint with the specified SLA. The requests are dispatched to different DCs as a result of the second step. *The main focus of our methodology in this chapter is the second step, i.e. load balancing for minimizing the environmental penalty under the budget and the service requirement.* We assume that dispatching overhead is negligible.

Hardness. The problem formulated in Equations (3.2a)-(3.2f) is $\mathcal{NP}$-hard even for deriving a feasible solution. This can be proved by reducing from the decision version of the knapsack problem.

Proof. We reduce from the decision version of the knapsack problem. For an input instance of the knapsack problem, we are given N items and two constants W and V, in which each item i has a weight w_i and a value v_i. The objective of the knapsack problem is to select a subset of the N items such that the total weight of the selected items is less than or equal to W and their value is larger than or equal to V. The knapsack problem is $\mathcal{NP}$-complete [56].

The reduction works as follows: We construct N DCs such that each DC has only two configurations for the performance and energy consumption. That is, for DC i, $\lambda_{i,1} = 0, E_{i,1} = 0, \lambda_{i,2} = v_i, E_{i,2} = w_i$. The performance requirement in current budgeting period b, $\lambda_{F,b}$ is set to V, while the budget is set to W. The cost to buy one unit from the brown energy source is set to 1 as well.

Therefore, there exists a feasible solution for the knapsack problem if and only if the reduced instance for the studied problem has a feasible solution. Hence, we conclude that deriving a feasible solution under budget and performance constraints for the studied problem is $\mathcal{NP}$-hard. $\square$

3.4 Our Solution

The drawback of solving the optimization problem separately for each control period (Equations (3.2a)–(3.2f)) is that the global optimality is not guaranteed. I.e., the possibility to trade off expensive green energy in one control period against cheaper green energy in another control period might remain unutilized. We show this by solving this problem optimally within each control period through dynamic programming. After that we present a simple greedy algorithm that, by optimizing the budget distribution, produces better results in our simulations. Finally, we combine the positives of both approaches to form our final solution.

3.4.1 Dynamic Programming (DP)

Penalty Table for a DC

We first consider how to optimize for any DC i in a control period when the local budget S_i and the local service requirement Γ_i are given. A point to note here: Γ_i is not the same as the previously mentioned Λ. In a given control period, Λ is the total service required at the whole ISP level, whereas Γ_i is the portion of that service that has to be provided by DC i. According to the definition, we know that we should choose the least power-intensive configuration, say k^*, of the data center that fulfills the service requirement, i.e., $\lambda_{i,k^*} \geq \Gamma_i$.

Suppose that $x_{i,j}$ with $0 \leq x_{i,j} \leq \min\{1, \frac{L_j}{E_{i,k^*}}\}$ is the fraction of the total energy purchased from the j^{th} energy source in DC i. It is now clear that the objective for this case is to minimize $E_{i,k^*} \sum\limits_{j=1}^{Z} x_{i,j} \cdot \phi_{i,j}$ such that $\sum\limits_{j=1}^{Z} x_{i,j} \cdot C_{i,j} \cdot E_{i,k^*} \leq S_i$ and $\sum\limits_{j=1}^{Z} x_{i,j} = 1$. This can be solved by using the linear programming solver in general. Since, the green energy sources have zero environmental penalty, the above linear programming can be solved by a simple algebra calculation in $\mathcal{O}(Z)$ time complexity given that energy sources are presorted for preference. We omit the details of algebra here.

By iterating all possible values of S_i and Γ_i, we can build the corresponding penalty table $p(i, \Gamma_i, S_i)$ to show the minimum penalty for DC i under the above configurations. If it is not feasible to support Γ_i under budget S_i, then, $p(i, \Gamma_i, S_i)$ will be set to ∞.

We remove the infeasible and dominated entries in the penalty p-table for DC i created above, to decrease the size of this table. An entry $p(i, \lambda, s)$ is dominated by another entry $p(i, \lambda', s')$ if $s \geq s'$, $\lambda \leq \lambda'$, and $p(i, \lambda, s) > p(i, \lambda', s')$.

Suppose that the p-table has Q_i entries for DC i after the above procedure. The p-table has to be generated in each control period because the penalty incurred depends on the time-varying energy prices which are not know a priori. For the k^{th} entry in the p-table for DC i with $k \leq Q_i$, we denote

- $\ell_{i,k}$ as the service provided (request rates),
- $s_{i,k}$ as the allocated budget, and
- $\pi_{i,k}$ as the penalty stored in $p(i, \ell_{i,k}, s_{i,k})$.

Building the Dynamic Programming Table

On the basis of the penalty tables (p-table) obtained for each data center in the previous step we can now build a dynamic programming table to select the appropriate configuration of every DC to provide the total required service.

Suppose that $P(i, \lambda, s)$ is the minimum penalty for the *first* i DCs under the budget s to provide the service requirement (total request rate) λ. For brevity, when $\lambda < 0$ or $s < 0$, we define $P(i, \lambda, s)$ as ∞. Clearly, for $\lambda \geq 0$ and $s \geq 0$, we know that

$$P(1, \lambda, s) = p(1, \lambda, s). \tag{3.3}$$

Where p-table is from the previous section.

For $i = 2, 3, \dots, N$, the following recursive formula can be adopted to minimize the total penalty P under budget $s \geq 0$ and service requirement $\lambda \geq 0$:

$$\begin{aligned} P(i, \lambda, s) &= \min_{k=1,2,\dots,Q_i} \{P(i-1, \lambda - \ell_{i,k}, s - s_{i,k}) \\ &\quad + \pi_{i,k}\}. \end{aligned} \tag{3.4}$$

Clearly, $P(N, \Lambda, S)$ is the minimum penalty for distributing the requests and the budgets. The standard dynamic programming technique can be adopted and the solution can be obtained via backtracking from $P(N, \Lambda, S)$. The time complexity for calculating a single entry $P(i, \lambda, s)$ based on Equation (3.4) is $\mathcal{O}(Q_i)$. To build the table correctly, we have to calculate $P(i, \lambda, s)$ from $i = 1, 2, \dots, N$ and from $\lambda = 0$ to Λ and from $s = 0$ to $s = S$ sequentially. This gives the overall time complexity $\mathcal{O}(NS\Lambda Q_{\max})$, where $Q_{\max}$ is $\max_i Q_i$.

Optimality and Complexity

The above presented DP approach derives the optimal solution to minimize the environmental penalty for a control period. However, in the problem scale, some level of discretization in both budget and service is mandatory. Appropriate discretization results in a smaller global penalty table (P) and this reduces the computation complexity. The construction of the table P depends on how we discretize the values of λ from 0 to Λ and the values of s from 0 to S. The complexity can be reduced by rounding down $s_{i,k}$ and s to the nearest integer multiple of a given number, let's say, I_s. That is, $s'_{i,k}$ is $\left\lfloor \frac{si,k}{I_s} \right\rfloor I_s$. Similarly, we can also round down $\ell_{i,k}$ and λ to the nearest integer multiple of a given number, let's say, I_λ. That is, $\ell'_{i,k}$ is $\left\lfloor \frac{\ell i,k}{I_\lambda} \right\rfloor I_\lambda$. Then I_s and I_λ can serve as the discretization factors of budget S and Λ. This makes the time complexity to $\mathcal{O}(N \frac{S}{I_s} \frac{\Lambda}{I_\lambda} Q_{\max})$.

3.4.2 Greedy Algorithm

We now present a heuristic algorithm based on a greedy strategy without building the penalty p-table constructed in Section 3.4.1. The two important factors to be considered are the penalty and the budget. These two factors are inversely related, i.e., to reduce penalty more budget has to be paid and vice versa. We devise a heuristic strategy which strives to minimize the weighted sum of both.

Suppose that in the DC i, it has been decided to use the k_i^{th} configuration. That is, it will provide λ_{i,k_i} service with E_{i,k_i} energy consumption. Suppose that $x_{i,j}$

with $0 \leq x_{i,j} \leq \min\{1, \frac{L_j}{E_{i,k_i}}\}$ is the fraction of the total energy purchased from the j^{th} energy source in DC i. If k_i is given for every DC i, the objective for this case is to

$$\text{minimize} \quad \sum_{i=1}^{N} E_{i,k_i} \sum_{j=1}^{Z} x_{i,j} \cdot \phi_{i,j} \tag{3.5a}$$

$$\text{such that} \quad \sum_{i=1}^{N} \sum_{j=1}^{Z} x_{i,j} \cdot E_{i,k_i} C_{i,j} \leq S, \tag{3.5b}$$

$$\sum_{j=1}^{Z} x_{i,j} = 1, \quad \text{for all } i \tag{3.5c}$$

$$\sum_{i=1}^{N} E_{i,k_i} \cdot x_{i,j} \leq L_j. \quad \text{for all } j \tag{3.5d}$$

As $\forall i \forall j x_{i,j}$ is a real number, with its value between 0 and 1, the above linear program can be solved optimally by using a linear programming solver or via linear algebraic calculation with low time complexity.

The algorithm works as follows: all the DCs are set to their lowest service setting, i.e. $k_i = 1$ and we check for feasibility of this setting in terms of budget and service by verifying the feasibility and solving the optimal solution for Equation (3.5a). If $\sum_{i=1}^{N} \lambda_{i,k_i}$ is no less than Λ, the algorithm terminates; otherwise it increases one DC i^* among the DCs to the next configuration $k_{i^*} + 1$. The selection of i^* is as follows:

Suppose that the current solution has set k_i. By advancing only DC i to the configuration $k_i + 1$, we can find the optimal setting in Equation (3.5a) for minimizing the penalty under this setting. Please note that the penalty is set to ∞ if there is no feasible solution for Equation (3.5a). By advancing the configuration of DC i, suppose that $\Delta_i^{service}$ is additional service, $\Delta_i^{penalty}$ is the additional penalty, and Δ_i^{budget} is the additional budget (this is none-zero when the budget has not yet been exhausted in the current solution).

For a DC i, we define two terms: *brownness*, i.e. penalty caused per unit of provided service ($\frac{\Delta_i^{penalty}}{\Delta_i^{service}}$) and *economy*, i.e. budget spent per unit of provided service ($\frac{\Delta_i^{budget}}{\Delta_i^{service}}$). The heuristic that we use is $brownness \cdot w_b + economy \cdot w_e$. Where w_b and w_e are the weights that can be assigned to prefer brownness over economy or vice versa.

Algorithm 1 presents the pseudo-code of the above greedy algorithm. The worst-case number of combinations that we have to check for different k_i in this algorithm is $\mathcal{O}(N^2 M)$, as in each while loop in Algorithm 1 we consider

Algorithm 1: The greedy algorithm

```
Input: Data center configuration table for all DCs, Service requirement: $\Lambda$,
       Budget: $S$, weights: $w_b$, $w_e$
Output: Configuration for all DCs: $k_i$
$k_i \longleftarrow 1$ for each DC $i$;
while true do
    if $\sum_{i=1}^{N} \lambda_{i,k_i} \geq \Lambda$ then
        if Equation (3.5a) has a feasible solution then
            return the solution $k_i$ for each DC $i$ with the purchase plan by
              solving Equation (3.5a) optimally;
        else
            return the solution $k_i$ for each DC $i$ but with "over budgeting" by
              buying all energy from the cheapest brown source;
    for each DC $i$ with $k_i < M$ do
        $\Delta_i^{service} \longleftarrow \lambda_{i,k_i+1} - \lambda_{i,k_i}$;
        calculate $\Delta_i^{budget}$, $\Delta_i^{penalty}$ based on Equation (3.5a);
    let $i^*$ be the minimum $(\frac{\Delta_{i^*}^{penalty}}{\Delta_{i^*}^{service}} \cdot w_b + \frac{\Delta_{i^*}^{budget}}{\Delta_{i^*}^{service}} \cdot w_e)$;
    $k_{i^*} \longleftarrow k_{i^*} + 1$;
```

up to N DCs and the number of iterations in the while loop is at most NM. For each combination, we have to solve Equation (3.5a). This can be sped up by starting based on the current solution. However, solving Equation (3.5a) by using linear programming solvers is already quite efficient. As we are not able to guarantee the budget satisfaction, over budgeting may be needed by borrowing from future invocations, as presented in pseudo-code.

3.4.3 Greedy + DP (G+D)

The greedy algorithm, when allowed over-budgeting, guarantees to find a feasible solution, if there exists one. It keeps increasing the offered service progressively in search of a feasible solution. In the worst case, it configures all the DCs to run at maximum service setting. However, in the average case, it finds a feasible setting much earlier. Moreover, the heuristic used for the greedy algorithm does not buy overly expensive green energy, resulting in an efficient budget usage. In comparison, the DP method finds the optimal solution in terms of environmental penalty, even if the cost to reduce the environmental penalty is overly prohibitive.

We devise a method to combine both approaches to accumulate the benefits of both: for a given control period we execute the greedy algorithm to find a feasible solution. We analyze the budget requirement of this solution and set this as the maximum budget constraint for the DP method. Since the greedy algorithm optimizes for the budget as well, its solutions are more miserly in terms of budget usage. Setting this budget as upper limit for DP results in a reduced search space for dynamic programming approach. In this way we achieve a solution which incorporates the budget optimization of the greedy algorithm with the optimal search for minimal environmental penalty from DP approach.

As G+D uses greedy and DP sequentially, its worst case time complexity is $\mathcal{O}(N^3 M \frac{S}{I_s} \frac{\Lambda}{I_\lambda} Q_{max})$, using the previously introduced symbols.

In the following sections we present our simulation setup and evaluation results.

3.5 Simulation Setup

We adopt the settings from [163] to evaluate the proposed solution by simulating the Google's setup for the location of DCs in the US. For these locations, we obtain the electricity pricing information from [117]. For our simulations, the following factors are important.

Non-varying factors include the hardware capabilities of the DCs. These include server capabilities and cooling infrastructure. We consider four DCs, in which each data center is equipped with homogeneous servers, as detailed in Table 3.2. We use the method in [159] to build the DC configuration table, presented in Section 3.2, by considering 50 servers in each data center. The resulting table has at most 87 entries in each data center. Other methodologies like [64] and [32] can also be adopted for calculating the DC configuration tables. Please note that the complexity of the presented solutions does not directly depend on the number of servers in DCs, but the number of entries in the DCs' configuration tables. Even when the servers in a DC increase, we can reduce the number of entries in the DC configuration tables by changing the management granularity.

The penalty for a green energy source is set to 0. The penalty for a brown energy source is set to 1. This multiplied by CO_2 kg generated per kWh gives actual environmental penalty.

Time-varying factors include energy prices. The availability for both forms of energy does not vary. The fluctuation in the production of green energy due

Table 3.2: Data center settings used for simulation (adopted from [92]): Speed ratio is the ratio of the frequency by adopting dynamic voltage frequency scaling (DVFS) to the maximum frequency in the system.

Specifications			Power settings		
			Speed ratio	Service (req/sec)	Power (W)
DC # 1	Location:	San Luis Valley Colorado	1.00	750	174.09
	Processor:	AMD Athlon	0.90	675	141.28
	Max Freq:	3.0 GHz	0.66	500	88.88
			0.50	375	68.13
			0.26	200	55.29
DC # 2	Location:	Los Angeles California	1.00	750	93.99
	Processor:	Pentium 4, 630	0.80	600	62.76
	Max Freq:	3.0 GHz	0.50	375	37.99
			0.40	300	34.10
			0.30	250	32.37
DC # 3	Location:	Oak Ridge Tennesse	1.00	850	194.00
	Processor:	Pentium D950	0.85	725	146.19
	Max Freq:	3.4 GHz	0.64	550	102.13
			0.44	375	78.82
			0.29	250	71.20
DC # 4	Location:	San Luis Valley Colorado	1.00	750	174.09
	Processor:	AMD Athlon	0.90	675	141.28
	Max Freq:	3.0 GHz	0.66	500	88.88
			0.50	375	68.13
			0.26	200	55.29

to environmental factors causes a shift in its price but the overall availability contracted by the suppliers in the form of RECs is fulfilled. Green energy has a higher price than brown energy as explained in the introduction (Section simulation we assume a surcharge of 1.5 cents and 18.0 cents per kWh for wind and solar energy [39] respectively in addition to the brown energy price. For price trace of electricity, we use the data from NYISO [117]. Specifically, we use Day-Ahead price data for Nov'07 for four regions previously mentioned. This period was chosen as it corresponds to the period of available Wikipedia traces.

The other time varying factor is the total service requirement, Λ_b. It is a random variable but overall it follows a weekly recurring pattern (see Figure 3.3). We use the actual workload trace from Wikipedia [150], i.e., Oct'07 for forecasting

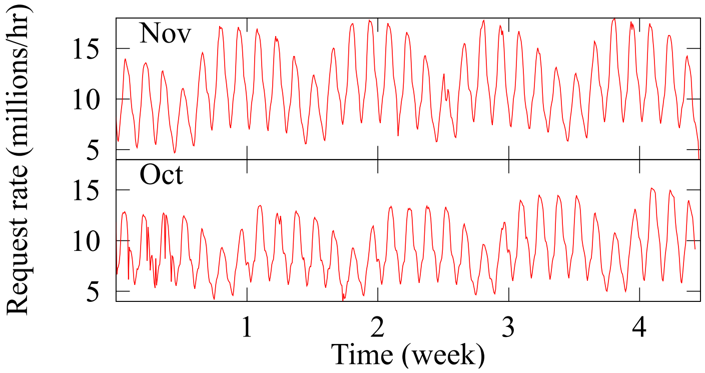


Figure 3.3: Wikipedia workload trace in Oct. and Nov. 2007 [150]

and the Nov'07 for the actual workload.

3.6 Evaluation

In this section, we present the results of our evaluations. We take a month as a budgeting period and an hour as a control period. For the greedy algorithm proposed in Section 3.4.2, we configure the heuristic weights as $w_b = 10$ and $w_e = 1$ in Algorithm 1. The presented algorithm (G+D) is evaluated for three main criteria, i.e. budget allocation and usage, environmental penalty minimization and computation time. We compare it with base line schemes of "All Green" and "All Brown" as well as DP approach (Section 3.4.1) and simple greedy (Section 3.4.2).

3.6.1 Budget allocation

The hourly budget is allocated as a weighted average of current monthly budget where the weights are calculated based on predictions. We adopt a simple prediction scheme that predicts the number of requests in the current control period based on the history. Any other prediction scheme, e.g., [160, 38] can be used for better predictions.

The budget usage comparison is presented in Figure 3.4. The maximum allowed budget was set to USD 80k. As expected "All Brown" uses the mini-

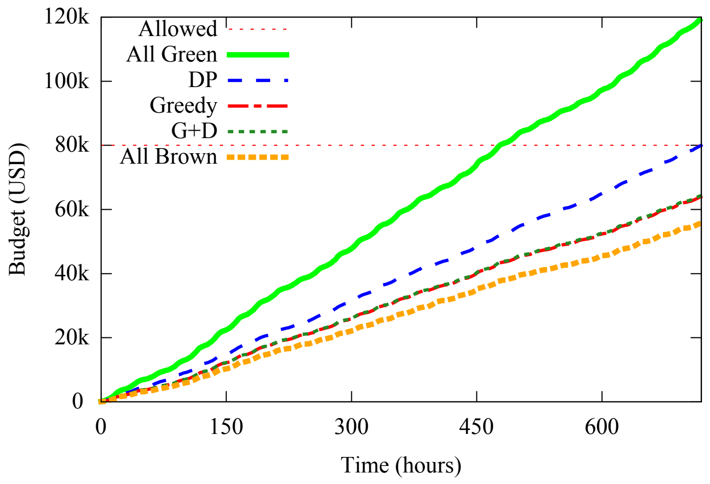


Figure 3.4: Budget usage

mum amount of budget at the cost of huge environmental penalty, whereas, “All Green” approach violates the maximum budget constraint.

The proposed solution, G+D, follows the same budget allocation as the greedy algorithm. In comparison with the optimal budget allocating scheme, “All Brown”, it uses only one eighth more budget but produces a 15 fold reduction in environmental penalty as shown in Figs. 3.4 and 3.5. In comparison with “All Green”, G+D uses only half the budget.

Relaxing the budget constraint results in decreased environmental penalty for the presented solution. The result is shown in Figure 3.6. The effect is, however, non-linear. This is because increasing the monthly budget beyond a certain point makes the availability of green energy the limiting factor.

For minimizing the environmental penalty, among the presented schemes, G+D outperforms all others that follow the budget constraints as shown in Figure 3.5.

The fundamental difference between G+D and the DP approach is the allocation of budget. Unlike DP, G+D tries to minimize the budget usage. This provides G+D a relaxed budget constraint progressively at subsequent control periods, as compared to the DP approach. DP produces the optimal results in terms of environmental penalty within a single control period. To this end, it sometime uses excessive budget for gaining a marginal reduction in penalty. This makes the budget constraint tighter in subsequent control periods, resulting in higher overall penalty for dynamic programming.

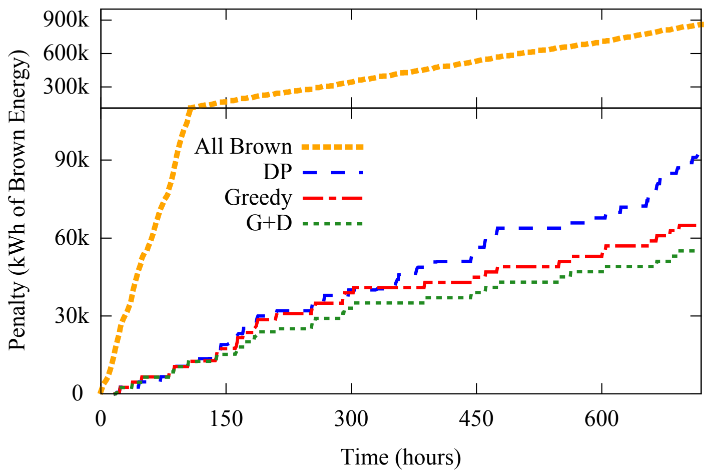


Figure 3.5: Method comparison

3.6.2 Computation Time

For the results to be useful, the maximum computation time must remain a negligible fraction of the length of the control period. This condition can be fulfilled by lengthening the control period. But, this, in turn, makes the prediction horizon longer for Λ_b and $C_{b,i,j}$. Irrespective of the prediction scheme, this results in deteriorated prediction quality hence affecting the solution quality.

One way to decrease computation time can be to compute the necessary tables offline. But, due to the time-varying factors like price and availability of energies, the offline tables will be huge and unpractical.

Online computation of solution at the beginning of every control period is the only viable option. Figure 3.7 presents the calculation time as a function of problem size for a single control period on a normal desktop machine (Intel i3, 6GB RAM, Linux). It is clear that the greedy algorithm is the fastest with majority of the computation times remaining within a second. However, G+D may take up to a minute in a few cases. This remains suitable for a control period of around an hour as necessitated by the electricity price horizon. changing the granularity of discretization in the DP. Furthermore, G+D is often faster than DP alone, although it uses DP as a subroutine. The is because of the reduced search space for DP that greedy algorithm prunes out. With growing problem size, the computation time increases linearly for the greedy algorithm and exponentially for dynamic programming based solutions. This was expected as DP is pseudo-polynomial time algorithm. However, G+D, still fairs better than

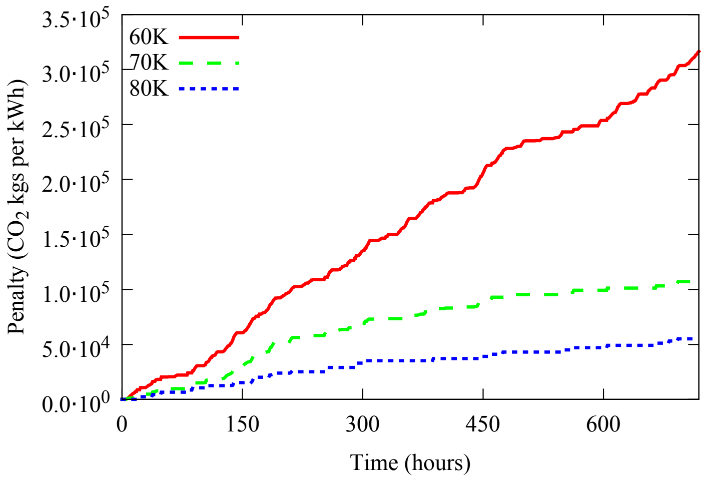


Figure 3.6: Budget vs. penalty (G+D)

DP alone. The is because of the reduced search space due to the pruning by greedy algorithm.

3.7 Summary

The environmental footprint of DCs is becoming significant. In this chapter, we formalized the problem of minimizing the environmental footprint of a large scale Internet service (or maximizing the green energy usage) while fulfilling the budgetary and service constraints. We showed that this problem is an $\mathcal{NP}$-hard problem and presented a viable greedy heuristic for optimization. The solution that we presented 1) is up to date, in that, it is based on current legislative and economic trends. 2) It is practical. By dividing the problem into two subproblems and solving them separately, it gives us the flexibility to add different kinds of SLAs and is also valid for heterogeneous servers in a single DC. 3) It is holistic in nature as it is cognizant of the energy usage for computation hardware, the networking hardware and also the cooling infrastructure of the DC.

The novelty of our approach lies in dividing the problem into two independent steps, that is, per DC optimization and a central optimization scheme. This forms the basis of a general solution that can include factors like power consumption due to cooling infrastructure, power consumption of networking

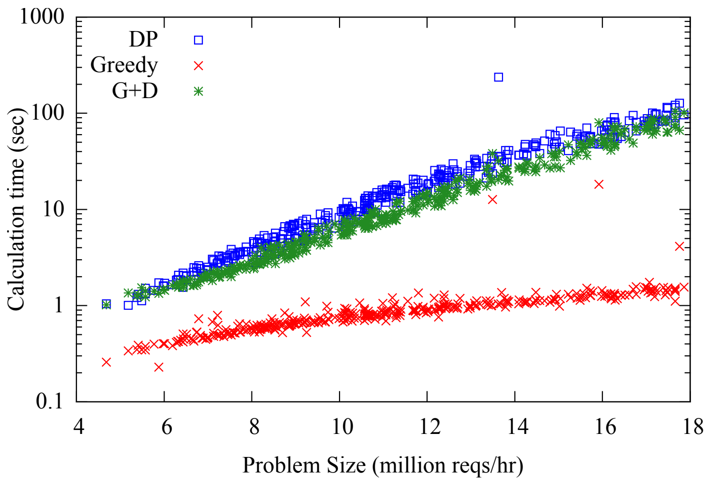


Figure 3.7: Problem size vs. calculation time

infrastructure, on-site renewable energy generation systems and multiple services with multiple SLAs.

We evaluated the presented solutions with traces of electricity prices and typical Internet workloads. Extensive evaluations based on real data for price, traffic and locations demonstrate the effectiveness of our approach.

4 Battery Inclusion for Peak Shifting

Electrical demand is generally random in nature. This randomness is a result of factors such as human life patterns (e.g., office-home cycle, weekdays and weekends, seasonal holidays etc.), weather phenomena, periodicity of seasons, tax laws and industrial production cycles, etc. Some of these factors show a periodic trend whereas others are sporadic one-time events.

Expected electricity production on the other hand, which used to be quite predictable and easier to plan for, is also getting random. This is a consequence of the ever-increasing share of electricity produced by renewable sources in the overall power-mix. Renewable sources, owing to their dependence on weather, are inherently more unpredictable than the traditional fossil fuel based sources.

Considering the above two factors, it becomes non-trivial to economically guarantee the grid reliability which consumers of electricity have grown accustomed to, while still optimally utilizing the renewable energy sources. The traditional approach in this case is to guarantee reliability by over-provisioning the fossil-fuel based production of electricity. This over-provisioning has to be proportional to the expected worst-case peak that can occur in the demand. The additional costs that are incurred in order to insure the reliability of the network are often passed on to the end consumers in the form of demand charges.

An often used mechanism for passing on these charges to the end consumers consists in the introduction of peak penalty tariffs, such as those offered to corporate consumers like industrial production facilities, data centers, etc. In this case, the actual bill not only depends on how much power was consumed but also how quickly it was consumed. I.e. two prices, say, c_1 and c_2, are fixed in the electricity purchase contract, then the monthly bill for the electricity demands of, say $(d_1, d_2, ..., d_n)$, is of the form $c_1 \cdot \sum_i d_i + c_2 \cdot \max_i\{d_i\}$. Here, the second term of the summation is the peak penalty. Peaks are often measured in the time scale of 15 mins to 1 hour. The peak penalty, also referred to as demand charge, forms the main chunk of the bill and often exceeds all other charges combined together [6].

A point to note: These legacy tariffs are still the most widely offered tariff structures for bigger consumers. However, with increasing dominance of renewable sources, it is expected that these tariff structures are going to evolve into a more general form, where the users will be encouraged to “shape” their electrical demand according to an expected electricity production curve. Considering their current market dominance, the focus of this chapter is on the former special form of these tariffs, i.e., where the electricity producers encourage a flatter demand by the consumers.

Three main approaches exist to economize the bill charged by electricity providers, as discussed previously in Chapter 2. Namely, 1) rescheduling/shedding the loads [26], 2) spatial redistribution of the loads [126], and 3) employing a store of energy to minimize the peaks [12, 119, 151]. All three of these are mutually orthogonal and can be employed independently. Our methodology is based on the last approach, i.e., employing a storage system to shift the demands.

As a side note, the demand shifting problem can be generalized as a minimax problem applicable to any item that can be stored in time of lower demand and then dispensed during the periods of higher demand. Economic lot sizing [15] is one example. This is a commonly found general problem. The approach that we present here is also more generally applicable to such problems. However, for the convenience of presentation, we will restrict ourselves to the terminology of batteries and electrical grid.

The contents of this chapter are based on our work presented in [110]. Here we extend our previous work in three main aspects. Firstly, we simplify the model we use for a battery storage system without losing the generality to include the losses that occur in such systems. Secondly, we simplify the mathematical proofs that we presented earlier. Lastly we use the electricity load trace from Karlsruhe Institute of Technology in Karlsruhe, Germany (KIT) and also provide an extended evaluation section.

4.1 Overview

The basic architecture of a demand shifting system is shown in Figure 4.1. It consists of three main components. The first component is the power grid. It is the service provider for electricity. In our model, we assume that this is the only source of energy input into the system. The second component is the local reservoir of energy where the surplus energy can be stored and later retrieved as needed. There are two main aspects of an energy storage system that determine its suitability for a demand shifting system, i.e., the cycle efficiency of the storage system and the maximum amount of energy that can be stored in it.

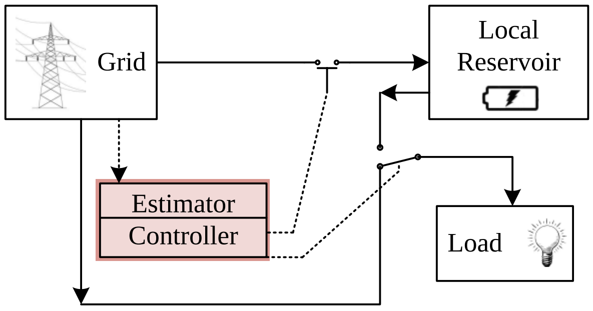


Figure 4.1: Generalized architecture of a demand shifting system. Solid arrows indicate the direction of energy flow and dotted arrows indicate the flow of information and control.

Many technologies exist for energy storage as discussed previously in Chapter 2. However, for the industrial consumers of electricity – our main target group in this study – battery electrical storage system (BESS) is the most promising option due to the unique benefits that it offers. E.g., it is not dependent on any special geological features that other systems like compressed air and pumped hydro storage need. It offers good cycle efficiency. Moreover, with the battery technology maturing, BESS will become ever more economical with time. Hence, we present our results mostly in the terminology pertaining to BESS, however, the presented results are equally valid for other storage systems as well.

In addition to the two main aspects discussed above, the third important component in a battery storage system is the controller that decides about when and how much energy to divert to the storage, and when to retrieve it back. The controller is the most crucial component in the system as the effectiveness of the whole system can potentially be compromised by a false control policy. It is also a focus of our research presented in this chapter.

We present a concrete mathematical basis for measuring the performance of a demand shifting system. The novelties of our approach presented in this chapter as compared to existing results in the literature are twofold. Firstly, given an arrival curve as an upper bound for the possible electrical loads, we devise a strategy to estimate the peak power demand when using a battery as an energy storage device. Secondly, we show the efficacy of such a scheme by employing it to decide the most appropriate battery size for large-scale consumers, though the presented scheme is equally applicable to consumers of any size.

Our approach can be summarized as follows: We first characterize the loads by adopting the concept of arrival curves in Network Calculus [89]. That is, the power loads for any given time interval are upper bounded by the given arrival curve. After this, we present a strategy to quantify the impact of any general controller constrained by the following two properties. 1) For a given sequence of electrical loads applied to a demand shifting system, if we start with a higher state of charge (SOC) of a battery, the controller must ensure that the final state of the battery also remains higher, or at least the same, as the final state achieved when starting with the lower SOC of the battery; 2) to increase the power demand at any instant in time, t, the load that occurs nearer to t in the past has a bigger, or at least the same impact on the total demand at t than the load occurring much earlier. We name this class of controllers monotonic controllers.

4.2 Organization

We presented a general overview about peak shifting systems in the beginning of this chapter. In the next section, we introduce the concept of applying the arrival curves for quantifying electrical loads and the corresponding mathematical properties. This is followed by the formal model of the system on which we base our analysis. In the following section, we provide a simple controller that charges/discharges the storage system based on a given threshold and prove that such a controller is a monotonic controller. We present a methodology in Section 4.6 to analyze the (worst-case) peak power demand to the power grid based on the arrival curve for monotonic controllers. The presented analysis technique is applied to the traces of real loads from the Karlsruhe Institute of Technology. The evaluations are presented in Section 4.7 followed by a section about the possible applications of the presented techniques.

4.3 Application of Arrival Curves to Electrical Loads

It is customary to provision the electrical supply for a building or an installation according to its rated load, where 'rated load' is the sum of individual load contributions from all the appliances installed in the building. The scenario where a building or an installation actually operates at its rated load is very unlikely to occur in practice, though. Consider for instance a residential building which is equipped with both an electrical heating system and an air cooling system.

Clearly, the heating and the cooling system – being weather dependent – are operated in a mutually exclusive way.

For a demand shifting system, basing the estimates of the peak demand of an installation on its rated load tends to be pessimistic and can lead to an over provisioned system requiring a significantly larger battery than what is actually required. As the battery is the main contributor towards the total cost of the demand shifting system, an over provisioned system implies economic inefficiency in form of avoidable costs. In the scope of our work, we attempt to rectify this by providing tighter upper bound for the electrical load posed by an installation. In order to achieve this goal, we adapt the concept of arrival curves as introduced in the framework of Network Calculus. The arrival curves were originally conceived for modeling the traffic arriving at different nodes in a packet-switching computer network, such as a router in Internet [89]. In such networks, each packet that arrives at any node in the network is buffered before being further processed or forwarded. For a reliable delivery of packets it is important to correctly dimension the buffering space for each node, where the size of buffer depends upon the processing capability of the node as well as the rate of arrival of the packets.

We extend the concept of the arrival curve to provide an upper bound for the electrical load. Similar to a stream of packets arriving at a node in a computer network, the electrical loads can be seen as a stream of electrical demands arriving at the electrical grid. These demands can either be fulfilled through a battery or directly through the grid or both. Here, the size of the appropriate battery required for a demand shifting system depends on the magnitude of electrical loads. We provide a mechanism to quantify the worst-case electrical demand that considers their magnitude, i.e., through an arrival curve as shown in Figure 4.2.

A point to note is that an arrival curve is not the same as a cumulative load trace. A cumulative load trace represents *one* concrete instance of loads, whereas an arrival curve provides a model which represents all possible load traces. We term any sequence of loads that is upper bounded by an arrival curve α as α-compliant.

4.3.1 A Method to Generate an Arrival Curve

Arrival curves are normally based on the historic trend of the loads. In this section, we present a simple method to generate an arrival curve. We set the value of Δ to be the smallest possible discrete window used by the supplier of electricity as basis for a demand charge, typically 15 mins. The highest load that ever occurred in the whole historic trace in any interval of length Δ gives

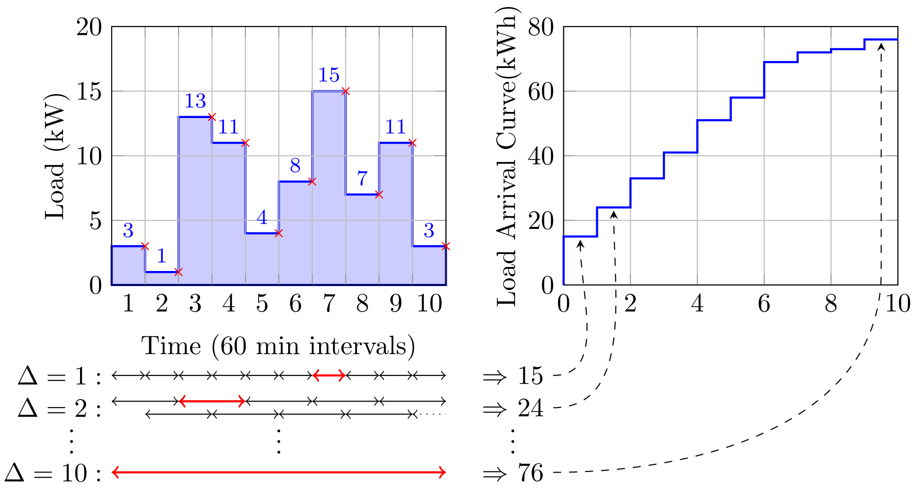


Figure 4.2: A method to generate an arrival curve from a historic load trace.

us the value of the arrival curve for that Δ. For the next iteration, we increase the size of Δ by a unit amount and repeat the same process, which gives us the second point for the arrival curve. The process is repeated until Δ is equal to the interval of our choice.

For example, we might use a load trace with the length of one year to generate an arrival curve with the length of one month. For this, we start with the Δ equal to one hour and increment it in each step by an hour till it equals one month. This gives us an arrival curve for one month. The 'pessimism' in the arrival curve can be increased by considering a longer load trace from history, for example ten years instead of one, and so on. This method is graphically shown in Figure 4.2. The left side of the figure shows the loads that need to be served. In order to discretize, we assume for every time slot that the load is accumulated at a single point in the end of the slot, as shown in the Figure 4.2. The right side of the figure shows the resultant arrival curve on basis of the loads. The intervals in which the maximum load is seen are marked in red at the bottom of the figure.

Note that we have moved from the absolute time to the relative time domain, thus some knowledge from the demand stream is lost, such as the exact time at which a specific situation occurred as well as any knowledge of the average load.

We argue that the arrival curve of the electrical loads of an installation is a tighter approximation of the actual loading pattern as compared to the contemporary practice of a single valued 'rated load'. This information can be exploited for choosing a suitable battery size for that installation, as we show in the following sections of this chapter.

The method presented here to generate an arrival curve is a simple and effective method with low computational complexity. However, it is not the only method. Other more effective methods can also be researched. But this is not the focus of our work. We assume that an upper load arrival curve is given.

4.3.2 Properties of Arrival Curves

In this section, we discuss the significant properties of arrival curves that are relevant to the results presented in this chapter. Please note that a counterpart to an upper arrival curve, i.e. a bound for the maximum load, is a lower arrival curve, i.e. a bound for the minimum expected load. However, in accordance with our goals, we limit our focus only to upper arrival curves. When not explicitly mentioned otherwise, we refer to upper arrival curve when using the term 'arrival curve'.

Definition

Given a function α, defined for $t \geq 0$, we say a series of chronologically ordered, consecutive load requests, say $\ell_1, \ell_2, ...$, are upper bounded by α, if and only if for any time interval $(s, t]$, where $0 \leq s \leq t$, the following equation holds.

$$\forall t \geq s \geq 0 \qquad \sum_{s<i\leq t} \ell_i \leq \alpha(t-s) \tag{4.1}$$

and $\alpha(0) = 0$.

We denote the function α as the upper arrival curve. For consistency in discretization and without any loss of generality, we assume the arrival curves to be left-continuous. Also, as we assume a unidirectional flow of energy, all the load requests are non-negative. Hence, an upper arrival curve cannot have a negative value. We use the short form of "α-compliant" for any sequence of loads that is upper bounded by an arrival curve α.

Sub-additivity

Any given load sequence can be compliant to more than one possible upper arrival curve as shown in Figure 4.3, e.g., α_1 and α_2. Both of these provide

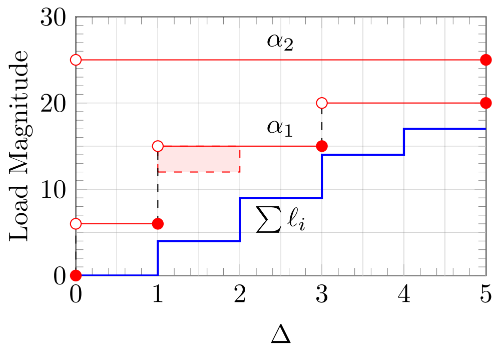


Figure 4.3: Different upper arrival curves for the same cumulative load sequence ($\sum \ell_i$) are shown. α_1 is not tight. α_2 is too pessimistic. α_1 can be safely made tighter by excluding the shaded area from it.

an upper bound for the shown cumulative load sequence. α_2 shows a very pessimistic bound for the load sequence. α_1, on the other hand, is not 'tight'. Consider for example $\alpha_1(1) = 6$, i.e., the maximum load in any single time slot cannot be more than 6, whereas $\alpha_1(2) = 15$, i.e., the maximum cumulative load in two consecutive slots can at most be 15. The latter is a more relaxed constraint than the former one, hence it is superfluous (shown as the shaded area in Figure 4.3). If the maximum load in one time slot is limited to 6, then maximum load in two consecutive slots can not exceed 12. Hence, using $\alpha_1(1)$, we can generate a more strict constraint for $\alpha_1(2)$, i.e., 12. Generalizing this, a 'tight' arrival curve is the one that follows $\alpha(s+t) \leq \alpha(s) + \alpha(t)$. This property is called *sub-additivity*. We assume that any arrival curve that is given is sub-additive.

Truncating an Arrival Curve

A given arrival curve can be safely truncated to a shorter length, if required. One motive to reduce the length of an arrival curve can be to reduce the computational effort for any subsequent analysis steps. While truncating an arrival curve to a required length, say t, it has to be insured that it remains defined for all points s, such that $0 \leq s \leq t$, i.e. the right most part of the curve is removed to shorten it to a desired length.

The loss of information resulting from the truncation of an arrival curve might remove some of the constraints from the sequence of loads that is bounded by this arrival curve. Due to reduced constraints on the acceptable load sequence, the amount of 'pessimism' in the system increases.

Combining Arrival Curves

More than one arrival curve can be combined together to make the load approximations better. The basic idea can be explained with the help of an example. Suppose, through an expert analysis, we know that the worst loads are generated each year in the month of December. We analyze the load traces for the month of December from last n years and generate corresponding arrival curves. These n disjointed arrival curves, say $\alpha_1, \alpha_2, ..., \alpha_n$ can be combined together to result in a single arrival curve, α_R by selecting a point wise maximum among all, i.e. $\alpha_R(i) = \max(\alpha_1(i), \alpha_2(i), ..., \alpha_n(i))$, where i ranges from 0 to the maximum number of time units in December.

A point to note here that the mathematical operation to combine more than one arrival curve into a single arrival curve remains the same even if the curves do not belong to similar intervals. For example, lets say through expert analysis it is found out that loads from last year's June are expected to occur in this year's December because of an aperiodic festival. In this case, we might need to combine arrival curve from last year's June with a historical arrival curve from December to get a prediction of the possible loads.

4.3.3 Alternative Uses of Arrival Curve

An approach similar to load quantification through arrival curves can be used to estimate the amount of energy produced by sporadic sources of energy, such as wind and solar. In contrast to the previous case where we estimated the electricity consumption through the maxima of the sums of loads in each interval Δ, we here choose the minimum energy produced in each interval. This data can be named 'production curve', and can be used to predict the least amount of energy that is guaranteed to be produced in any future period of a given length. This information can be useful for predicting the amount of fossil fuel based energy that should be provisioned in the future.

Note: For the rest of the discussion in this chapter we assume the arrival curve for the electrical load of the concerned installation is given a priori.

4.4 System Model and Problem Definition

In this section, we formalize the system model and present the problem definition. Figure 4.4 shows the abstract architecture for the peak shaving system

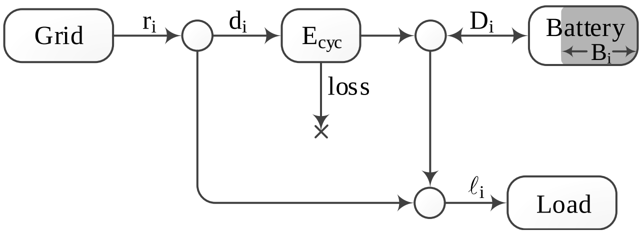


Figure 4.4: Abstract model for a demand shifting system. The arrows show the direction of energy flow.

considered in our scheme. Although such a demand shifting system operates continuously, for ease of modeling, we discretize the system. We assume that all the operations, such as charging, discharging and requests to the grid, are point-operations and take place at the end of the corresponding time slot.

4.4.1 Load Model

As mentioned before, we discretize the time into time slots and index them through i. For time slot i, the plant requests ℓ_i amount of energy as a non-elastic load, i.e. it is *non-postponable* and *non-reducible*. It is assumed that the connection to the grid is capable of fulfilling the worst-case peak demand at any instant. We assume that the load ℓ_i is unknown until the time slot i and is lumped together as a point load at the end of the time slot. The loads could come irregularly, periodically, sporadically, or with jitters. The only constraint is that they are upper bounded by an arrival curve α. Throughout the analysis we assume that the arrival curve α is given.

4.4.2 Battery Model

The system is equipped with a battery whose maximum capacity is given as B_{max}. At time slot i, the state of charge (SOC), c_i, is the available percentage of B_{max}, where $0 \leq c_i \leq 1$. In other words, the amount of energy available in the battery at slot i, B_i, is given as $c_i B_{max}$. Note, B_i is the amount of energy in battery before putting a slot's request, r_i, to the grid or satisfying the load, ℓ_i, because we assume that the load is lumped at the end of the slot. In the later sections, we use SOC or B_i interchangeably to refer to the residual energy in the battery. Clearly, $0 \leq B_i \leq B_{max}$. The battery may either be charged or discharged, but not both, in a single time slot. The amount of energy extracted from the battery in a time slot i is denoted by D_i. If, in i, the battery is charged, instead of discharged, then D_i is negative.

In practice, an actual, non-ideal battery suffers from two types of inefficiencies. Firstly, it suffers from the cyclic inefficiency, i.e. the charge that is stored in the battery cannot be completely retrieved later. We model this loss as follows. Out of the total energy routed towards the battery, d_i, we assume only a fraction of it, D_i, actually gets stored in the battery, with $D_i = E_{cyc} d_i$ and $0 \leq E_{cyc} \leq 1$. The typical cyclic efficiency of commonly used batteries is normally 70–90%, as presented earlier in Table 2.3. Secondly, the battery loses a fraction of its charge continuously with time which is called self-discharge rate. The typical value of self-discharge rate in each time unit of interest (~15 min for demand charge) is so small that we ignore it to keep the model simple.

It is worth noting here that for the sake of simplification, we lump the cyclic loss, $1 - E_{cyc}$, of the battery at a single point at which the battery is charged, instead of spreading it over the whole charge-discharge cycle. Henceforth, we only consider D_i, i.e., the amount of energy that can be retrieved from the battery after charging, with the losses implicitly accounted for as $D_i = E_{cyc} d_i$.

4.4.3 Monotonic Controllers

In a system featuring a battery and a charging controller, the peak demand from the electrical grid depends not only on the electrical load, but also on the decisions of the controller, because the timing of charging and discharging the battery influences the peak demand posed to the grid.

For every time slot, the controller has to decide either to charge or discharge the battery and the amount of energy to request from the grid. That is, the controller decides following two parameters for a time slot, i.

1. r_i: the amount of energy requested from the grid, and,
2. D_i: the amount of energy extracted from the battery. If the battery is being charged instead, then D_i is negative. Clearly, $(B_i - B_{max}) \leq D_i \leq B_i$.

We assume that the controller is causal and does not have knowledge of the future. We also assume that at any time slot the controller can satisfy the load partially from the battery and partially by requesting the grid, i.e., $\ell_i = r_i + D_i$. Fractional mixing is allowed. This is a technically feasible and fairly common assumption [109]. A controller is said to be a feasible controller, if it ensures that the demand is always satisfied, i.e., $\forall i \; \ell_i = r_i + D_i$. As the grid is assumed to be capable of fulfilling any demand at any instant, there are no restrictions on r_i.

Clearly, for a system with negligible losses, the total energy input to the system should be equal to the total output for a set of contiguous time slots in an interval, i.e.

$$\forall s \leq t \qquad B_s + \sum_{s \leq i < t} r_i = B_t + \sum_{s \leq i < t} \ell_i. \tag{4.2}$$

For the approach presented in this chapter, we only consider *monotonic* controllers. Formally, a monotonic controller is defined as one that satisfies the following two properties in addition to being feasible.

M1 *First property of monotonicity.* For a given loading sequence, if a controller starts with a higher SOC, it ends with higher SOC. This is formalized as follows. Suppose, we are given a contiguous sequence of loads $\ell_s, \ell_{s+1}, \ldots, \ell_{t-1}$ for time slots in an interval $[s, t)$, where $s < t$. We are also given two initial SOC of a battery, B_s and B'_s. Using the controller for the given load sequence when starting from B_s at time slot s, the battery charge level at time slot t is B_t. Starting from B'_s at time slot s for the same load sequence, the battery charge level at time slot t is B'_t. A controller is said to be *monotonic* if it guarantees $B_t \leq B'_t$ if $B_s \leq B'_s$.

M2 *Second property of monotonicity.* The nearer the load occurs before the point of measurement, the higher is its impact in terms of energy requested from the grid at this point of measurement. Formally, suppose that we are given a chronological contiguous sequence $\mathcal{L}$ of loads $[\ell_1, \ell_2, \ldots, \ell_i, \ell_{i+1}, \ldots \ell_{t-1}]$. Consider another chronological contiguous sequence $\mathcal{L}'$ of loads $[\ell_1, \ell_2, \ldots, \ell'_i, \ell'_{i+1}, \ldots \ell_{t-1}]$. $\mathcal{L}$ and $\mathcal{L}'$ are identical except in slot i and $i+1$, where the difference is given as,

$$\ell'_i = \ell_i - \sigma \tag{4.3}$$

$$\ell'_{i+1} = \ell_{i+1} + \sigma \tag{4.4}$$

Here σ is strictly positive, i.e., $\sigma \in \mathcal{R}_+$. A helpful visualization of $\mathcal{L}$ and $\mathcal{L}'$ is shown in Figure 4.5. The controller, $\mathcal{C}$, is said to be *monotonic* if the request from the grid for slot t by considering $\mathcal{L}'$ is no less than the request from the grid for the same time slot by considering $\mathcal{L}$.

The analysis techniques presented in this chapter are only applicable to monotonic controllers. In Section 4.5, we show a practical example of such controllers.

4.4.4 Problem Definition

The peak measurement problem is defined as follows. *Given a monotonic controller $\mathcal{C}$, which serves as an interconnect between the electrical grid, a battery of size*

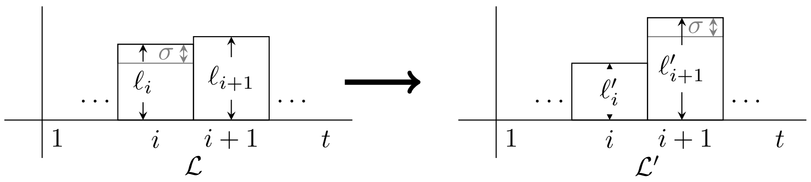


Figure 4.5: Constant request controller - Second property of monotonicity (M2). Shifting the load (σ) nearer to the point measurement, t, shall not decrease the magnitude of request at t.

B_{max} and inelastic electrical loads whose demands are bounded by (upper) arrival curve α, the objective is to find the maximum peak request among the unknown demands.

4.5 Constant Request Controller

In this section we present a simple example of a monotonic controller as defined earlier. This controller is referred to as constant request controller (see Algorithm 2). It always requests a constant amount R_{th} from the grid as far as the battery allows. If the total request is more than the load, the surplus is charged to the battery and if it is less, the deficiency is fulfilled from the battery. If requested energy, R_{th}, is more than free capacity in battery and the load combined together, then situation is termed as an overflow and the request from the grid is decreased to the sum of demand (ℓ_i) and free capacity of battery ($B_{max} - B_i$). Analogously, if the battery and R_{th} combined together are still not enough to satisfy the load, then an underflow is said to have occurred and the unsatisfied load is supported directly from the grid.

The pseudo-code for the constant request controller is presented in Algorithm 2. Here the main outputs of the algorithm are battery status for the next time slot and the amount of energy to request from the grid for the present time slot. For these quantities, Algorithm 2 can be simplified as follows.

$$B_{i+1} = \max(0, \min(B_{max}, R_{th} - \ell_i + B_i)) \tag{4.5}$$

$$r_i = B_{i+1} - B_i + \ell_i \tag{4.6}$$

Algorithm 2: Constant request algorithm

```
Input: Load demand: $\ell_i$,
       Algorithm parameter: $R_{th}$,
       Battery parameters: $B_{max}, B_i$.
Output: Grid request: $r_i$,
        Battery state: $B_{i+1}$.
$r_i = R_{th}$
$B_{i+1} = r_i + B_i - \ell_i$
/* Applying battery constraints */
if $B_{i+1} > B_{max}$ then
    // Overflow
    $r_i = r_i - (B_{i+1} - B_{max})$
    $B_{i+1} = B_{max}$
else if $B_{i+1} < 0$ then
    // Underflow
    $r_i = r_i - B_{i+1}$
    $B_{i+1} = 0$
return $r_i, B_{i+1}$
```

4.5.1 Offline-optimal Controller

The constant request controller can also be used to emulate other simple controllers, such as an offline-optimal controller. As the name implies, an offline-optimal controller optimally chooses the time-slots for charging and discharging the battery with complete prior knowledge of the load trace. To emulate an offline-optimal controller through the constant request controller, R_{th} is set to the average of the loads in the budgeting period, then, provided the battery is of sufficient size, i.e. underflow and overflow never occurs, this controller gives the optimal reduction in peak request.

The basic purpose of the offline-optimal controller is to provide a baseline to which the performance of other controllers can be compared.

It can be shown that the constant request controller is a monotonic controller as follows.

Lemma 1. *The constant request algorithm satisfies the property M1 for monotonic controllers.*

Proof. We prove the statement for M1 defined in Section 4.4.3, for constant request algorithm. M1 states a controller is said to be monotonic if it guarantees

$B_t \leq B'_t$ if $B_s \leq B'_s$.
I.e.,

$$(B_s \leq B'_s) \implies (B_t \leq B'_t) \tag{4.7}$$

This can be re written as follows.

$$\begin{aligned} & \neg(B_t \leq B'_t) \implies \neg(B_s \leq B'_s) \\ \Rightarrow \quad & (B_t > B'_t) \implies (B_s > B'_s) \end{aligned} \tag{4.8}$$

We start with the left hand side of equation 4.8 and show that the right hand side holds after applying the constant request controller.

Starting in time slot t, $B_t > B'_t$. Then, applying the constant request algorithm i.e. Equation (4.5) we can write as follows.

$$\begin{aligned} & B_t > B'_t \\ \Rightarrow \quad & \max(0, \min(B_{max}, K + B_{t-1})) > \max(0, \min(B_{max}, K + B'_{t-1})). \end{aligned} \tag{4.9}$$

where, $K = R_{th} - \ell_{t-1}$, which is the same for both cases as we have the same charging rate and the same load sequence.

For any x and y, if $\max(0, x) > \max(0, y)$, then $x > y$. Also, for any x, y and C, if $\min(C, x) > \min(C, y)$, then $x > y$.

Using these inferences, Equation (4.9) can be simplified to,

$$B_{t-1} > B'_{t-1}.$$

Applying the same argument backwards progressively to all slots until slot s would imply $B_s > B'_s$. This shows that equation 4.8 holds, hence constant request algorithm satisfies M1. □

Lemma 2. *The constant request algorithm satisfies the property M2 for monotonic controllers.*

Proof. We prove that the statement for M2, as defined in Section 4.4.3, holds for constant request algorithm. Using the same loading sequences, $\mathcal{L}$ and $\mathcal{L}'$, as defined in Section 4.4.3, we observe that they consists of three contiguous, non-overlapping intervals: $[1, i-1]$, $[i, i+1]$ and $[i+2, t)$. In the first and the last interval the loading sequences are identical and only differ from each other in the middle interval where the differences are shown in Equations (4.4) and (4.3). We assume the initial battery status for both loading sequences is the same, i.e. B_0, and battery status for any subsequent slot, say q, is given as B_q for $\mathcal{L}$ and B'_q for $\mathcal{L}'$.

We start with the first interval $[1, i-1]$. Since the control algorithm, the loading sequence and the initial battery status are the same for both loading sequences, we conclude that the final battery statuses at the end of the first interval will also be the same for both controllers, i.e.

$$B_i = B'_i \tag{4.10}$$

Considering the third interval, $[i+2, t)$, the loads in $\mathcal{L}$ and $\mathcal{L}'$ are the same from $i+2$ to $t-1$. Therefore, if the inequality $B_{i+2} \geq B'_{i+2}$ always holds then by Lemma 1, we can say $B_t \geq B'_t$.

Therefore, to prove the second property of monotonicity for the constant request controller, it suffices to prove the following inequality.

$$B_{i+2} \geq B'_{i+2} \tag{4.11}$$

Here, on basis of Equation 4.5, B_{i+2} and B'_{i+2} are given as

$$\begin{aligned} B_{i+2} &= \max(0, \min(B_{max}, R_{th} - \ell_{i+1} + B_{i+1}) \\ B'_{i+2} &= \max(0, \min(B_{max}, R_{th} - \ell'_{i+1} + B'_{i+1}) \end{aligned}$$

Say, $K_1 = R_{th} - \ell_{i+1}$ and using Equation (4.4), the above two equations can be reformulated as:

$$\begin{aligned} B_{i+2} &= \max(0, \min(B_{max}, K_1 + B_{i+1}) \\ B'_{i+2} &= \max(0, \min(B_{max}, K_1 - \sigma + B'_{i+1}) \end{aligned}$$

For any x and y, if $x \geq y$ then $\max(0, x) \geq \max(0, y)$. Therefore, to prove Equation (4.11), we must show that

$$\min(B_{max}, K_1 + B_{i+1}) \geq \min(B_{max}, K_1 - \sigma + B'_{i+1}) \tag{4.12}$$

For any x, y and C, if $x \geq y$ then $\min(C, x) \geq \min(C, y)$. Therefore, to prove Equation (4.12), we must show that

$$\begin{aligned} & K_1 + B_{i+1} \geq K_1 - \sigma + B'_{i+1} \\ \Rightarrow \qquad & B_{i+1} \geq -\sigma + B'_{i+1} \end{aligned}$$

Substituting the values of B_{i+1} and B'_{i+1} as per Equation (4.5) in this, we get

$$\max(0, \min(B_{max}, R_{th} - \ell_i + B_i) \geq -\sigma + \max(0, \min(B_{max}, R_{th} - \ell'_i + B'_i)$$

Using Equations (4.10), (4.3) and $K_2 = R_{th} - \ell_i + B_i$, we can rewrite this as

$$\max(0, \min(B_{max}, K_2) \geq -\sigma + \max(0, \min(B_{max}, K_2 + \sigma) \tag{4.13}$$

Therefore, in order to prove Equation (4.11), we need to prove Equation (4.13). There are four sets of conditions that can occur here. We show that for each one of these, Equation (4.13) holds.

1. $\underline{B_{max} < K_2 \text{ and } B_{max} < K_2 + \sigma}$.
   Under these conditions, Equation (4.13) can be simplified as follows.

$$\begin{aligned} & \max(0, B_{max}) \geq -\sigma + \max(0, B_{max}) \\ \Rightarrow \quad & B_{max} \geq -\sigma + B_{max} \end{aligned}$$

   Since, B_{max} and σ, both are strictly positive, Equation (4.13) holds with $>$.

2. $\underline{B_{max} < K_2 \text{ and } B_{max} > K_2 + \sigma}$.
   For every $B_{max} < K_2$, it must hold that $B_{max} < K_2 + \sigma$ because $\sigma \in \mathcal{R}_+$. Hence, this case is not possible.

3. $\underline{B_{max} > K_2 \text{ and } B_{max} > K_2 + \sigma}$.
   Under these conditions, Equation (4.13) can be simplified as follows.

$$\max(0, K_2) \geq -\sigma + \max(0, K_2 + \sigma) \tag{4.14}$$

   Considering this equation, there are only two possible scenarios here as follows.

   a) $\underline{K_2 > 0}$.
      Under this condition Equation (4.15) can be simplified as follows.

$$K_2 \geq -\sigma + K_2 + \sigma$$

      Clearly this equation holds. Hence, for this case, Equation (4.13) holds with equality.

   b) $\underline{K_2 \leq 0}$.
      Under this condition Equation (4.15) can be simplified as follows.

$$0 \geq -\sigma + \max(0, K_2 + \sigma)$$

      For this equation, the right hand side can have the possible values from the set $\{-\sigma, K_2\}$. Both outcomes are less or equal to zero. Hence, for this case also, Equation (4.13) holds.

4. $\underline{B_{max} > K_2 \text{ and } B_{max} < K_2 + \sigma}$.
   Under these conditions, Equation (4.13) can be simplified as follows.

$$\begin{aligned} & \max(0, K_2) \geq -\sigma + \max(0, B_{max}) \\ \Rightarrow \quad & \max(0, K_2) \geq -\sigma + B_{max} \end{aligned} \tag{4.15}$$

   Considering this equation, there are only two possible scenarios here as follows.

a) $\underline{K_2 > 0}$.
Under this condition Equation (4.15) can be simplified as follows.

$$\begin{aligned} & K_2 \geq -\sigma + B_{max} \\ \Rightarrow \quad & K_2 + \sigma \geq B_{max} \end{aligned}$$

This equation is true as we know that the second pre-condition for this case was $B_{max} < K_2 + \sigma$. Hence, for this case, Equation (4.13) holds with $>$.

b) $\underline{K_2 \leq 0}$.
Under this condition Equation (4.15) can be simplified as follows.

$$\begin{aligned} & 0 \geq -\sigma + B_{max} \\ \Rightarrow \quad & \sigma \geq B_{max} \end{aligned}$$

This equation is true because, using the second pre-condition for this case, i.e., $B_{max} < K_2 + \sigma$, and $K_2 \leq 0$, it must hold that $\sigma > B_{max}$. Hence, for this case also, Equation (4.13) holds with $>$.

Hence, we conclude that Equation (4.13) and therefore, Equation (4.11) holds for all cases, therefore constant request controller satisfies M2. □

Theorem 4.1. *A controller $\mathcal{C}$, that follows the algorithm presented in Algorithm 2 is a monotonic controller.*

Proof. This is simply based on Lemmas 1 and 2. □

4.6 Analysis Technique

In this section, we present the main contribution of this chapter, i.e. , the technique to analyze the peak electrical request from the grid given 1) a load arrival curve and 2) a monotonic control scheme to manage the battery.

The core idea of the analysis technique is based on the following three observations.

1. Firstly, the more the amount of charge in the battery, the more it will be able to help in the reduction of request from the grid
2. a load occurring nearer prior to the point of measurement results in a not less drained battery at the point of measurement than a load occurring further back in time, and

3. trivially by the definition of the arrival curve, the worst-case load in any time interval $[t, t + \Delta)$ is upper bounded by the initial Δ portion of the arrival curve.

Based on these observations, we can synthetically construct a load trace for a given time slot, that maximizes the load at that slot. When the load is maximized, this leads to the peak in request from the grid at that point. Repeating this procedure for every time slot that belongs to an interval of interest, gives us the maximum peak possible for each time slot of that interval. The highest peak for the interval then is the highest peak among all the calculated peaks for the interval.

The above presented observations form the foundation for our analysis technique. The first two of these are, indeed, the two properties of monotonicity as discussed in the context of monotonic controllers, that is why our scheme is limited only to monotonic controllers.

The third observation is a direct corollary of the definition of the arrival curve, i.e., Equation (4.1), which can be used to generate worst case loading sequences. Worst case loading sequences are discussed in detail in the following section.

4.6.1 Worst Case Load

As the name suggests, a worst case load sequence is such a loading sequence that results in the highest possible peak in the request from the grid at the point of interest. Due to its property of causing a peak in the request, worst case load sequence is used extensively in the analysis technique we present. In this section, we discuss the concept of a worst case loading sequence and present a procedure for generating such a sequence when an arrival curve is given.

According to the definition as shown in Equation (4.1) of an arrival curve, say α, the worst case load of any single slot is upper bounded by $\alpha(1)$. Similarly, the worst case load of any two consecutive slots is upper bounded by $\alpha(2)$, and so on. This property of arrival curves, combined with second property of monotonicity, can be exploited to generate the worst case load sequence for any arbitrary slot t. For example, to generate a worst case load sequence of two slots, the following set of equations must be satisfied.

$$\begin{aligned} \ell_2 &= \alpha(1) \\ \ell_1 + \ell_2 &= \alpha(2) \end{aligned}$$

Similarly, to generate a worst case load sequence of length three, the following set of equations must be satisfied.

$$\begin{aligned} \ell_3 &= \alpha(1) \\ \ell_2 + \ell_3 &= \alpha(2) \\ \ell_1 + \ell_2 + \ell_3 &= \alpha(3) \end{aligned}$$

These sequences can be simplified to give a loading sequence of size t as

$$\forall k \in [1, t-1] \qquad \ell_{t-k} = \alpha(k+1) - \alpha(k)$$

and $\ell_t = \alpha(1)$.

We generalize this idea and prove its correctness in Theorem 4.2, where Equation (4.16) formally defines the worst case loading sequence for any slot t.

Important to note, considering the two examples of worst case load sequences discussed above, the loading sequence for slot 2 is actually the same as that of slot 1 except shifted by one place and one term added in the beginning. In other words, if the worst case loading sequence for slot 1 is $\mathcal{L}$, then the same for slot 2 can be stated as $\mathcal{L}' = (\ell'_1, \mathcal{L})$. This implies that if we need to calculate the worst case loading sequences for all slots belonging to a contiguous single interval, only one extra term needs to be computed for every slot which is then prefixed to the loading sequence computed for the earlier slot. This property of worst case loading sequence makes it computationally easier to calculate.

Theorem 4.2. *Under a given arrival curve, α, a monotonic controller and an initial battery charge B_0, the peak request the controller puts to the grid at a time slot t is that which results due to a load sequence $\mathcal{L}$ of loads $\ell_1, \ell_2, \ldots, \ell_t$ satisfying the following:*

$$\forall i \in [1, t] \qquad \sum_{k=i}^{t} \ell_k = \alpha(t - i + 1) \tag{4.16}$$

Proof. Suppose there exists a loading sequence $\mathcal{L}'$ that results in a higher request at time instant t than $\mathcal{L}$ when starting with the same B_0. Then, due to the monotonicity of the controller, $\mathcal{L}'$ must have an interval $[i, t]$ with higher cumulative load than $\mathcal{L}$, i.e., $\sum_i^t \ell'_k > \sum_i^t \ell_k$, where $i \leq t$. This implies that $\mathcal{L}'$ lies outside the bounds of the arrival curve α and hence is not a valid loading sequence. □

4.6.2 Methodology

To determine the maximum possible peak request that might be put to the grid by a controller in an interval, say $[0, t')$, we follow a three step procedure.

Firstly, we determine the worst case loading sequence for a slot, say t, belonging to the interval $[0, t')$ using the given arrival curve. This loading sequence is unique for every slot and is synthetically generated as described in Theorem 4.2.

In the second step, using the previously generated loading sequence for the time slot t in combination with a given monotonic control algorithm and battery parameters, we simulate the normal working of the system. Then, the request that the controller puts to the grid at the slot t is recorded.

Both of these steps are repeated for every t belonging to the interval $[0, t')$, i.e. a budgeting period. Once we have all the worst case requests for every time slot, we simply choose the maximum among them in the last step,. This gives us the maximum possible peak that might occur in the interval of interest.

An important point to note here is that if the system starts with an empty battery, then the highest peak in request is trivially the one that occurs in the first slot for a single-slot worst case loading sequence which is equal to $\alpha(1)$. In other words, if we start with an empty battery, it is as if we did not have a battery in the system as far as peak analysis is concerned. However, if the starting battery status is non-zero then the highest peak in the request is less than highest peak in the worst case load. For this reason, the control algorithm must insure that at the end of each budgeting period the battery is not completely drained so that the next budgeting period does not start with an empty battery.

4.6.3 Battery Partitioning

There are several ways to make sure that the SOC at the start of each budgeting period is non-zero. For instance, one way is to partition the battery into two parts, $\mathcal{R}$ and $\mathcal{N}$, let's say with the capacities of 25% and 75% of the original battery, respectively. For the normal working of the system only $\mathcal{N}$ is used while $\mathcal{R}$ is constantly charged in preparation for the next cycle. The rate of charge per time slot for $\mathcal{R}$ is set to $\frac{\text{free capacity in } \mathcal{R}}{\text{no. of time slots left in the budgeting cycle}}$, i.e. linear charging to ensure that $\mathcal{R}$ is full by the end of the budgeting cycle. If during the normal operation $\mathcal{N}$ is fully charged, then the normal requests (R_{th}) can be used to supplement the charging of $\mathcal{R}$. The requests from the grid to charge $\mathcal{R}$ are in addition to the normal requests that the controller generates for the normal operation. This guarantees that at the beginning of every new budgeting period at least 25% SOC is available in the battery. However, a price is paid: only 75% of the total capacity of the battery is available for the normal operation of the system.

Another possibility is to partition the battery after a specific fraction of the budgeting period has elapsed (e.g., in the middle), instead of doing this in the start. That is, the whole battery is employed for a normal peak shaving operation up to the chosen instant and afterwards a part of the battery is reserved for linear charging while the rest continues to be used for the normal operation. This is a generalization of the scheme discussed earlier. The primary beneficiary in this version are those systems that are equipped with a battery comparatively smaller than their mean load. In such systems, owing to the small capacity of the battery, the conditions of overflow and underflow are more likely to occur. Hence, by delaying the partitioning as late as possible, the system can employ the full battery for a longer period. Moreover, as the battery is not too big, the linear charging in the last portion of the budgeting period is not likely to cause a high peak.

Finding the appropriate partitioning point is an optimization problem in its own right. For instance considering sizing, the bigger the portion of battery reserved for future preparation, the lesser the system's ability is to dampen the peak in load that occurs after partitioning. Conversely, the smaller the portion of the battery reserved for the preparation of the next budgeting cycle, the higher the amount of pessimism is in the analysis for the next cycle because the guarantee for initial SOC is proportionally scaled down. Similarly, considering timing, if the battery is partitioned too early, it becomes more probable that peaks in the load will occur during the period when the system's ability to respond is decreased. On the other hand, if the battery is partitioned too late, the overhead in the request due to the additional load for charging might be too high, causing peaks of its own. Fortunately, these decisions can be fine tuned by considering the historical trends and running simulations on the old traces. However, we do not explore this aspect in the scope of this report.

4.6.4 Computational Complexity

The computational complexity of the presented methodology is derived from the above mentioned three steps. Let's say there are n time slots in a budgeting period. In the first step, the worst case loading sequence is generated for each of these. At maximum, this requires as many computational steps as are the number of slots in a budgeting interval. For a worst case loading sequence of n slots, it takes n steps.

In the second step, the control algorithm is simulated for worst case loading sequence for each slot. The complexity of this step depends upon the complexity of the control algorithm. In case of the constant request algorithm (Algorithm 2), it takes linear time which can be upper bounded by n steps. Complexity of both these steps together for a single slot can then be upper bounded by $(n+n)$.

For a total of n slots, the combined complexity for these two steps is given as $n(n+n)$, hence $2n^2$.

In the last step, the highest peak is selected among the peak requests for every slot. This is equivalent to choosing the highest number in an unsorted array of n numbers, i.e. n-steps. Considering this, the computation steps can be upper bounded by $2n^2 + n$, i.e. $\mathcal{O}(n^2)$.

4.7 Evaluations

To evaluate our technique for Algorithm 2 presented in Section 4.5, we use the electrical load for the month of April 2012 from Karlsruhe Institute of Technology (KIT), Germany. As April includes Easter, it provides a good example of monthly recurring and non-recurring usage patterns. Figure 4.6 shows the trace that we use. The figure comprises of two charts with synchronized x-axes, each of these is elaborated as under.

The chart at the top shows KIT's actual load trace for April 2012 along with a request trace that a constant request controller would generate when a battery of 20 MWh is employed in the system where the constant charging rate (R_{th}) is 6.428 MW. R_{th} is selected to be 10% more than the mean of the load trace, whereas the battery size is selected to be approximately four times that value.

The chart at the bottom shows the corresponding SOC for the battery over time (SOC chart). Naturally, when the battery is used to support the load, the SOC decreases and vice versa. The starting SOC is 25%. As can be seen in the graph, for the last week of April, starting at midnight 24^{th} April, the battery is logically partitioned into two portions, one for the normal operation and one for the preparation of the next budgeting cycle, namely $\mathcal{R}$ and $\mathcal{N}$ respectively. $\mathcal{N}$ is equivalent to 75% of the total capacity of the battery, whereas $\mathcal{R}$ occupies the rest. From the point of partitioning onwards, $\mathcal{R}$ is charged at a constant rate so as to have it fully charged by the end of the budgeting period. Additionally, when $\mathcal{N}$ is full and the load is less than R_{th}, then the underutilized portion of R_{th} is also diverted to $\mathcal{R}$'s charging, as discussed in Section 4.6.2. This is seen in the graph, e.g. the progression of SOC for $\mathcal{R}$ during the day of 26^{th} April is higher than usual. This mechanism guarantees that at the start of the next budgeting period, the battery will have at least 25% SOC.

In the SOC chart, looking at the progression of charge accumulation in $\mathcal{R}$, it can be seen that partitioning the battery a week in advance is not an optimal decision in this particular case as $\mathcal{R}$ saturates in less than 3 days. Hence, the partitioning point could have been delayed further. In the rest of the evaluations, we set the partitioning of the battery to occur at an instance when 90%

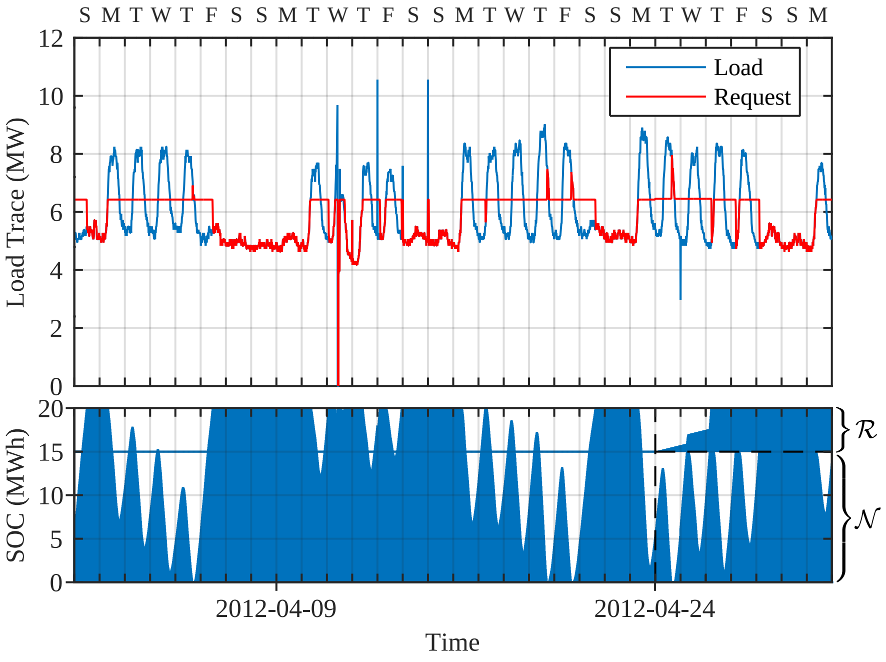


Figure 4.6: For April 2012, KIT's load profile vs request profile by constant request controller for a battery size of 20 MWh and charging rate (R_{th}) of 6.428 MW. Bottom diagram shows the SOC of battery. $\mathcal{R}$ is the portion of the battery reserved for the preparation of the next cycle, whereas $\mathcal{N}$ is the leftover portion employed for normal operation.

budgeting period has elapsed, and we divide the battery into portions of 25% and 75% as $\mathcal{R}$ and $\mathcal{N}$, respectively.

Apart from this, two key observations can be made in Figure 4.6. Firstly, the resultant request curve is significantly flatter than the original load curve. And, secondly, the peaks in the request occur whenever the battery is empty and hence unable to support the load. Both of these aspects generally correspond to the objectives of a peak shaving system, i.e. to reduce the maximum peak in request and to have flatter request curve.

We examine these aspects further in Figures 4.7 and 4.8. As previously noted,

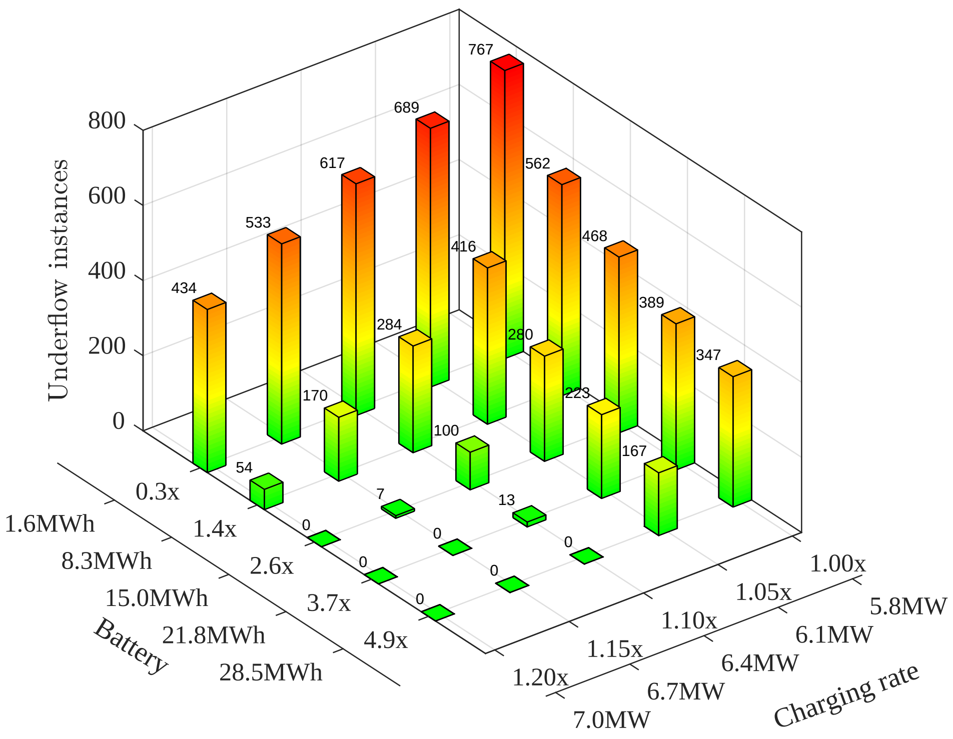


Figure 4.7: Number of underflow instances for different R_{th} and different battery capacities for constant request algorithm.

the peaks in the request normally occur when the battery is completely drained (underflow). Hence, the total number of such instances can be used as a metric to judge the performance of the control algorithm. In Figure 4.7, we present the results of the evaluation of constant request algorithm (Algorithm 2) in this regard. We plot the total number of underflow instances that occur during the simulation of actual load trace for different values of the constant charging rate (R_{th}) and the battery capacity.

Clearly, the magnitude of requests and the optimum size of battery for an installation is intrinsically linked to the shape and magnitude of its load curve. To give an insight into the interdependence of these variables, two scales are presented on both of the independent axes in Figures 4.7 and 4.8: the outer scale in absolute units and the inner scale as a function of average value of the load trace.

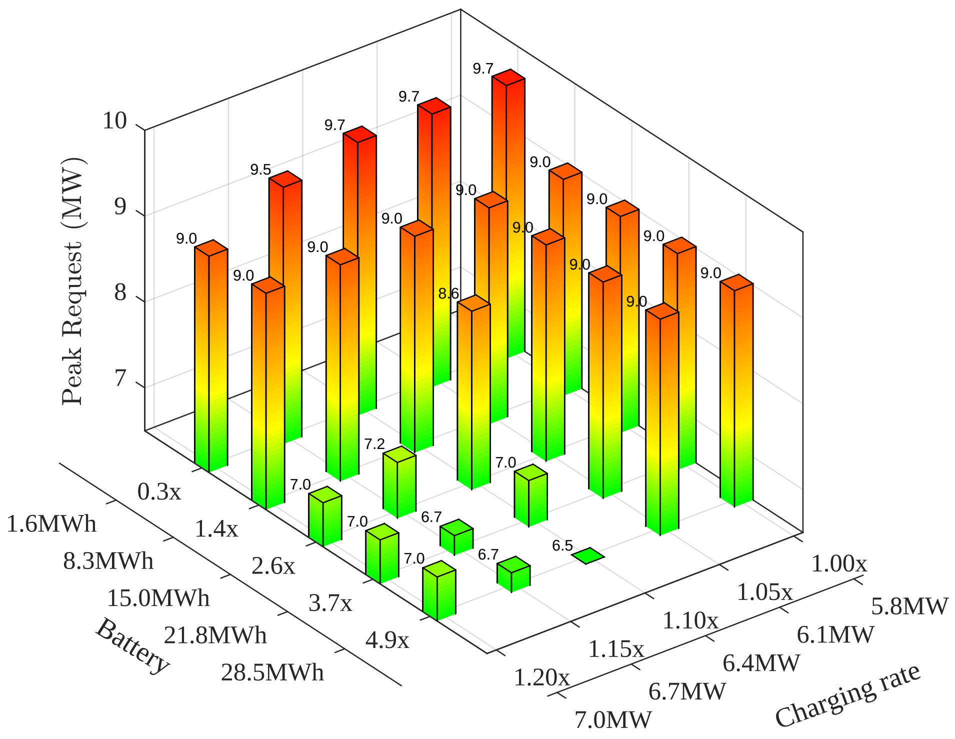


Figure 4.8: Peak request for constant request algorithm for different R_{th} and different battery capacities.

A point to note here is that although Figures 4.7 and 4.8 have a similar appearance, they express two different aspects of the peak shaving system. Figure 4.7 exhibits the average performance of the system whereas Figure 4.8 shows the worst case performance. We elaborate this point with the aid of an example. Consider the data points belonging to the series where battery capacity is 8.3 MWh. In Figure 4.7, we see that with the increase in charging rate the number of underflow instances decrease, i.e. the performance of the system improves. Considering the same series in Figure 4.8, we see that the peak request remains at 9 MW although the R_{th} is increased by 20%. This shows that the battery size of 8.3 MWh is not enough to substantially affect the highest peak in the load trace. Based on this, we conclude that in this case increasing R_{th} improves the average case performance of the system but not the worst case.

In Figure 4.8, we can observe that for the worst case performance of the system both R_{th} and battery capacity can be the limiting factor and the peak in request can not be reduced substantially through any one of them alone. Consider for instance data points belonging to the R_{th} of 6.1 MW in Figure 4.8. Here when the battery size is increased from 1.6 MWh to 8.3 MWh the resulting highest peak decreases. But after this, further increase in battery capacity does not result in any decrease in the highest peak. The reason for this is that in this case R_{th} has become the limiting factor. Observing the corresponding data points in Figure 4.7, i.e. where R_{th} is 6.1 MW, reveal that increasing the battery size helps reduce the underflow instances, i.e. resulting in the flatter request curve. But the highest peak in the load trace remains beyond the reach of our system due to too small R_{th}. Similarly, observing the data points belonging to the battery capacity of 1.6 MWh in Figures 4.8 and 4.7, one can observe that increasing R_{th} helps in the average case but not in the worst case. This can be attributed to a similar reason as discussed earlier, i.e., the battery capacity has become the limiting factor here.

Another difference between Figures 4.8 and 4.7 lies in the fact that average performance of the system as measured by the number of underflow instances improves monotonically with an increasing resource input to the system, i.e. R_{th} and battery capacity. This however is not the case with the worst case scenario as measured by peak power request in Figure 4.8. Here we see, for instance, for the battery capacity of 21.8 MWh, an increase in R_{th} results in the decreased peak request in the beginning, but since R_{th} itself is also included in the total request, increasing it further becomes counterproductive.

It is also worth noting that the performance of Algorithm 2 is more sensitive to changes in R_{th} as compared to changes in the battery capacity. For example in Figure 4.7, for the battery size of 1.6 MWh, which is a quarter of the mean load (5.92 MW), increasing R_{th} by 20%, that is from 5.8 MW to 7.0 MW, results in a reduction of over 42% in underflow instances. For the same 20% increase in R_{th} but with a bigger battery in system, i.e. 8.3 MWh instead of 1.6 MWh, results in over 90% reduction in underflow instances. To achieve the same reduction through a bigger battery alone, the size has to be more than doubled. Consider for example the instances which correspond to battery sizes of 8.3 MWh and 15 MWh with an R_{th} of 6.7 MW. Here the battery size is almost doubled to effect a 90% reduction in underflow instances. This shows that R_{th} can be used to control the peak more effectively, as long as the battery is not the limiting factor.

Table 4.1: Summary of Electricity Tariff HPN0 for Industrial Consumers - Duke Energy, Indiana [6]

Component	Charge
Total Demand Charge	23.3084 \$/kW
Total Energy Charge	0.0295 \$/kWh

4.8 Applications

A demand shifting system consists of a number of components whose interplay determines the net utility of such a system. The ability to model the components and estimate the outcome can save one from expensive experiments. The methodology presented in this chapter enables us to model the interplay of different components of such a system and determine the worst case results for different objectives.

The premier application of the presented technique is in determining the most appropriate battery size as well as the configuration parameters for the control algorithm. We present an example here to illustrate our point.

Consider, for example, an office complex such as KIT being served by a typical electricity tariff for an industrial consumer, such as the one discussed in the case study presented in [6]. We reproduce the salient features in Table 4.1. From trace analysis for KIT (Figure 4.6), we know that magnitude of highest peak for April was 10.560 MW. According to the mentioned tariff, this results into a demand charge of USD 246 thousand for the month.

Keeping these facts in mind, the important question here is: how big a battery should we employ in order to maximize the financial savings?

Normally, for peak shaving systems the battery of choice is Sodium Sulphur (NAS) flow battery as it is considered the most appropriate for such applications [124]. Its costs around 300 \$/kWh and has a service life of up to 20 years. For our analysis we amortize the cost over the span of the whole life with a nominal interest rate of 5%.

Building on the evaluations done earlier, we use the same data points and present the results in Figure 4.9. Here we use two factors to calculate the net monthly costs: 1) the amortized cost of the battery, and 2) the cost of the peak in request that occurs at the point after employing the battery. We find out that using the battery of 15 MWh with an R_{th} of 7 MW is the most economical configuration where the net monthly operation costs USD 193 thousand, i.e.

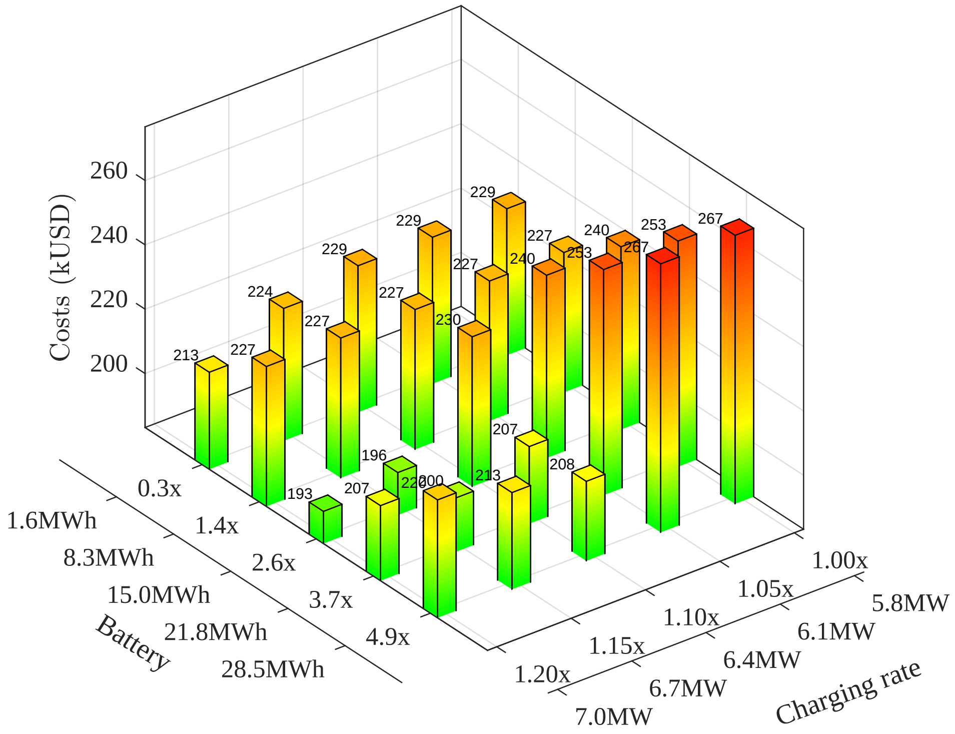


Figure 4.9: Cost estimates for NAS battery with 20 year life cycle for constant request algorithm for different R_{th} and different battery capacities.

realizing a savings of USD 53 thousand per month or more than half a million per year.

4.9 Summary

In this chapter our aim was to analyze and provide solutions for the demand shifting problem so that the associated environmental and financial benefits can be realized. We tackled a special version of the problem, i.e. the peak shaving problem using the battery as an energy storage device in the system.

We started with a scheme to quantify the electrical loads using arrival curves and argued that it is more realistic than the existing 'rated load' scheme. On this foundation, we devised a methodology to calculate the peak demand for a given battery under monotonic controllers. We presented a simple and effective example of a monotonic control algorithm and employed it to illustrate the conditions that must be met for a controller to be monotonic. After this we presented our methodology for monotonic controllers with which we can determine the worst case peak requests that a peak shaving system can produce. In the last section we evaluated the constant request controller using actual traces from Karlsruhe Institute of Technology and showed the potential associated financial benefits.

With the increasing share of renewable energy sources in grid, the demand shifting problem is gaining in importance. In this chapter we attempted to contribute by solving a special version of this problem. However, in general, many avenues of research in this field remain open, such as tackling flexible or postponable loads, providing absolute or probabilistic guarantees about grid availability, managing distributed storage etc.

5 Intra DC Optimizations

In the previous chapter, we discussed a technique which can be applied in a DC to manage peak power consumption by employing an energy storage device. In this chapter, we present another approach to that end, applicable specifically to DCs, by using real-time scheduling techniques. We explore the possibility of scheduling the jobs on servers within a DC in a centrally coordinated manner, so that peak power consumption is minimized while meeting the timing constraints set for each job.

To handle jobs in DCs, the MapReduce model [41, 42] is widely used. It follows a simple two phase approach, i.e., `Map` and `Reduce`. Many complex programs can be trivially broken down into this computing paradigm. We present the details of this process in Section 5.2. A cluster of servers executing a MapReduce program in the presence of an SLA is fundamentally similar to a many-core processor executing tasks that must be finished within their deadlines, i.e., real-time constraints.

In this chapter, we provide an overview of the MapReduce model and establish its congruence to many-core processors. Next, we present the techniques for power management for such systems under the timing constraints imposed by real-time workloads. For this, we present polynomial time heuristic algorithms for different cases that originate due to heterogeneity of processing cores and tasks and demonstrate their effectiveness in decreasing the power consumption using a simulation based on a 48-core prototype processor.

The techniques presented in this chapter are based on our previous work [112]. We reproduce the same mathematical results here with an extended discussion about them. Also, we discuss the applicability of our results to scheduling in MapReduce.

5.1 Organization

This chapter is organized as follows. In the next section, we present an overview of the MapReduce architecture (Section 5.2) along with its resemblance to a large many-core processor. Next, we discuss the problem of power management on many-core processors and the related challenges arising due to the phenomenon of dark silicon. In Section 5.3, we present a brief introduction and motivation for the problem of dark silicon. This is followed, in Section 5.4, by a formal model of a many-core processor executing a real-time workload. In Section 5.5, we present the core idea of our scheme for power management on many-core processors. In Sections 5.6, 5.7 and 5.8, we deal with the different cases arising from the heterogeneity of tasks and processing cores, while focusing only on frame-based real-time tasks. In Section 5.9, we generalize our solution to include implicit deadline periodic tasks. Section 5.10 includes the results of simulations, followed by the chapter summary in Section 5.11.

5.2 MapReduce

Invented by Google Inc., MapReduce [41] is a popular programming model for processing and generating large data sets, by exploiting the parallel processing capabilities of a cluster or a grid of commodity off-the-shelf computers. In this section, we provide an overview of the MapReduce model and its salient architectural details, along with a short example of how it works and its similarities to many-core processors. It is not, however, intended to provide extensive details about MapReduce here. For these, the reader is referred to the many studies produced on this topic, e.g., [84, 42, 113], among others.

5.2.1 Overview

Among the many distributed computing paradigms available on the market, MapReduce has established itself as one of the more popular models. The benefits of the MapReduce programming model have been demonstrated on a wide spectrum of domains, ranging from search and ads analysis [9, 60, 70] to bioinformatics [99, 104, 133], artificial intelligence, machine learning and data mining [23, 55, 165]. Over the years, a number of different implementations of MapReduce have emerged, the famous ones being Hadoop MapReduce by Apache Corporation [84] and Disco by Nokia Corporation [113]. Both are published and maintained as open source software.

The central idea of the MapReduce model is to facilitate the use of large number of computers in a DC for parallel execution by abstracting out complex details, such as concurrency, failure recovery and time management. This allows the users to focus on data processing strategies instead of infrastructure issues. The user is required to provide only two functions `Map` and `Reduce`. These are applied sequentially to the input data and the intermediate outputs are stored on a distributed file system. The system takes care of the error management, scheduling, robustness, and locality issues related to the data and code.

5.2.2 An Example of the MapReduce Workflow

The MapReduce model is well suited for workloads consisting of batch requests, commonly called *jobs*. A job normally refers to a longer running execution entity, with processing time typically in the range of minutes to hours (in contrast to a *request*, with processing time in milliseconds to seconds). The processing of jobs in the MapReduce model is coordinated centrally by a "master" node which assigns the work packages to "worker" nodes. The worker nodes execute the work assigned to them and update their status by sending periodic "heart-beat" messages to the master. A classic example of a job is a program that is used to determine the popularity of different websites by counting the number of accesses of each URL that appears in the logs of different web servers. The size of these logs can reach up to multiple terabytes.

We present a sample MapReduce workflow to calculate the frequency of words in a set of documents, as shown in Figure 5.1. This example is similar to the previously mentioned, common use case of MapReduce: counting the accesses for each unique URL in the web server logs. The execution of such workflows is initiated by separating the source data into independent splits. The processing of the input-splits is divided into two phases: the map phase and the reduce phase. During the map phase, the master node initiates a map task for each split of the input data and assigns these tasks to the idle worker nodes. The worker nodes executing the map task apply the user-provided `Map` function onto each split, generating intermediate key-value pairs. In the presented example, a tuple in the form of (<word>,1) is emitted for every word encountered in the input data. This intermediate output is stored on a distributed file system that is accessible for all other workers. When all of the `Map` tasks have ended, the master node instantiates `Reduce` tasks on idle nodes in the network, one per unique intermediate key; i.e., in the presented example, one per unique encountered word. The job of the reduce tasks, in this case, is to aggregate the list of intermediate data produced by `Map` tasks to a single value representing word frequency by summing the values from the previous step.

When executing a MapReduce job, the number of map tasks does not depend on the number of available nodes in the network, but the number of blocks the input is split into. Each block is assigned to a single map task. Similarly, the number of reduce tasks depends upon the number of keys generated during mapping.

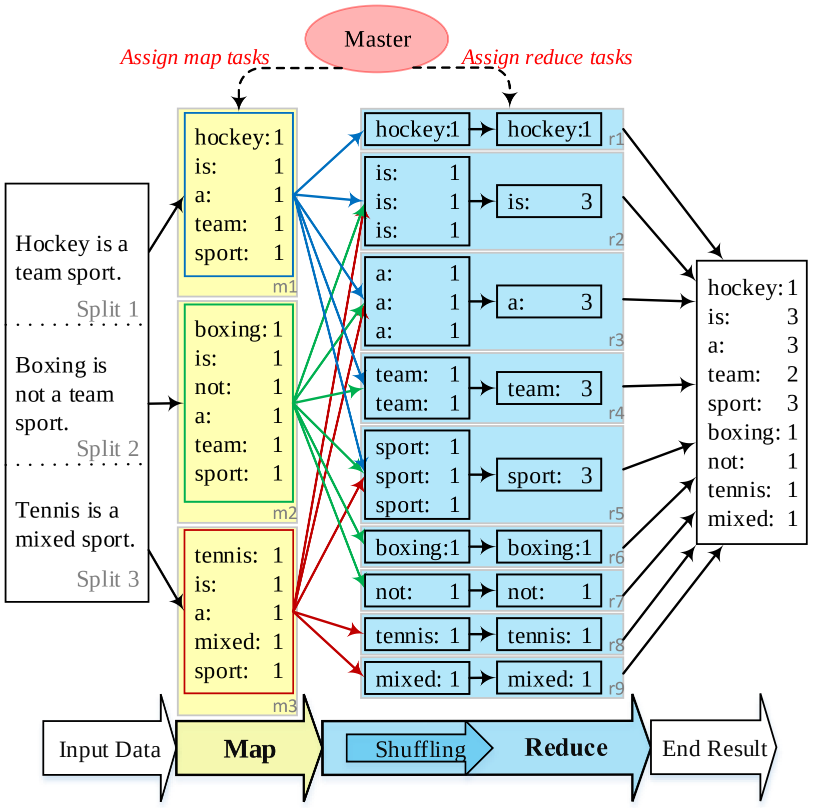


Figure 5.1: A sample MapReduce workflow: Execution phases and architecture.

5.2.3 Scheduling in MapReduce

In a single MapReduce job, there are multiple map and reduce tasks, each of which is a single unit of work that can be executed in parallel during the job's respective map and reduce phases. There can be multiple MapReduce jobs in the system, each with its own set of map and reduce tasks. Since the reduce phase consumes the output of the map phase, reduce tasks are executed only after all of the map tasks (i.e., the map phase) of the corresponding job have been completed. However, different MapReduce jobs might be undergoing different phases at the same time. The master node is responsible for scheduling these tasks, and distributes them to a number of slave nodes as per the number of free slots they have, where a slot is the smallest unit of time for which a processor is allocated to a task.

Generally, the MapReduce jobs are seen as best-effort, batch-mode-only applications, although, the trend is changing. MapReduce applications are increasingly latency-sensitive and operate under demanding workloads that require fast response times for data-intensive computations. Consider examples in which the timeliness is highly desirable, such as online log processing [18], traffic simulation [157] and personalized recommendations [9]. Moreover, due to input and output dependencies that often arise when carrying out complex algorithms, MapReduce jobs usually form Directed Acyclic Graphs (DAG), called MapReduce workflows. Formulating scheduling policies for jobs represented by DAGs, under real-time constraints, is currently an active research topic [93, 95, 97, 98, 123].

MapReduce deployments typically consist of thousands of machines sharing a high-speed and high-bandwidth intranet. Assuming bounded network delays, which can be subtracted from the respective deadlines, the MapReduce platform acts as a very large many-core processor [95]. This makes the results about scheduling for the domain of multiprocessor environment easily portable to MapReduce, and vice versa. In the rest of this chapter, we focus on scheduling and power management issues in the domain of many-core processors and related issues such as dark silicon.

5.3 Power Budget in Dark Silicon

The amount of silicon available on the processing chips is increasing, while Moore's Law continues to hold. Increased integration is resulting in many-core architectures. Analogously to Moore's Law for increase in silicon for every generation of processing chips, another prediction was made that the power consumption per transistor will keep decreasing at roughly the same rate as increasing silicon, i.e., the so-called Dennard scaling [45]. Historically, the transistor power reduction afforded by Dennard scaling allowed manufacturers to raise clock frequencies drastically from one generation to the next to achieve performance gains, without significantly increasing overall circuit power consumption. However, Dennard scaling holds no more. Now, with more silicon available on chips, designers' focus has shifted toward core heterogeneity and accelerator-rich designs to achieve higher performance. This increase of digital logic on the chips and the failure of Dennard scaling [45] is resulting in increased power densities on next-generation chips. Consequently, introducing the so-called Dark Silicon problem, where a significant percentage of the total available cores in a many-core system cannot be powered-on simultaneously due to thermal constraints [50, 144, 66]. Commonly, the chip manufacturers provide a Thermal Design Power (TDP) value which is considered to be the highest *sustainable* power that a chip can consume without triggering any performance throttling mechanisms [75], such as Dynamic Thermal Management (DTM). Heat-sink and the chip cooling solution are designed according to the TDP value.

Based on technological data from ITRS and Intel, various estimates [50, 66] show exponential growth in the amount of dark silicon with each process generation. For instance, according to [50], in a coming decade more than 50% of the chip area be dark for 8 nm chips. Clearly, with every successive generation, the operation below TDP will leave increasingly more parts of a chip dark.

The operational range below TDP is preferred for real-time tasks because it is guaranteed to be DTM-free and activation of DTM can result in hardware-based performance throttling to keep the chip within safe operating conditions. Modeling the resultant performance loss precisely to guarantee real-time performance constraints is non-trivial, because it introduces new variables in the problem, e.g., ambient temperature and the DTM policy of the chip. One way of guaranteeing sustainable performance is to have a DTM-free operation. If operation below TDP guarantees that DTM is not activated, then, *managing the cumulative peak power consumption of the cores to be within the TDP limit guarantees DTM-free operation*.

Treating TDP as a hard limit negates the possibility of using the cores at high power for short spans of time, i.e., so called *thermal sprinting*. On the other hand, the major benefit it provides is that it becomes a basis of a simple, online and pessimistic, but *sufficient*, test for guaranteeing the feasibility of real-time tasks scheduled on many-core chips whose TDP values are known. This *peak power based sufficient schedulability test* is similar to the widely used utilization based schedulability test for the design of schedulers in real-time systems. Numerous reports of the latter appear in the literature. Moreover, for scheduling decisions, this test abstracts the need to consider the details, such as initial temperature of the core, distance of the core from the periphery, distance from other active cores, etc.

In order to provide a DTM-free operation, another important aspect is the heterogeneity of cores and tasks. Different tasks have different peak power consumptions on different cores, as shown in Figure 5.2. Since heterogeneous multi-core scheduling is an $\mathcal{NP}$-complete problem [14], the approximation involved in task-partitioning algorithms often leaves cores less than 100% utilized. This leftover utilization can be used to put the core into a low power sleeping state. Appropriately scheduling this sleep period for each core can result in a decreased peak power consumption for the chip, thereby helping to remain within the TDP constraint and offering guaranteed performance due to the DTM-free operation.

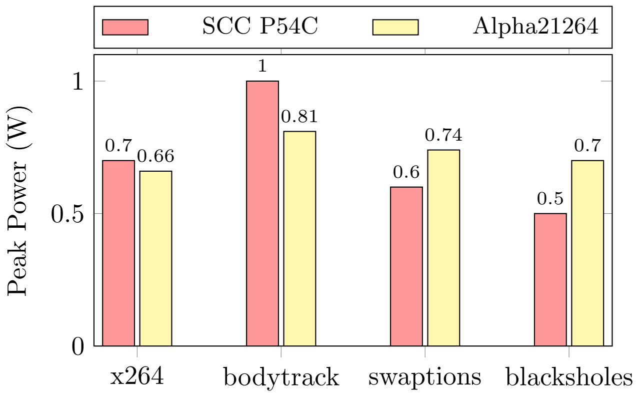
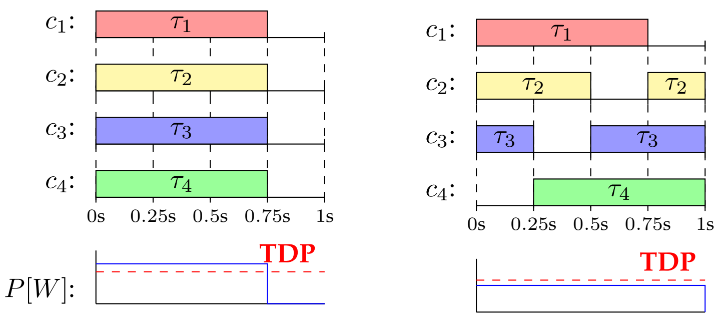


Figure 5.2: Peak power consumption values for a subset of applications from PARSEC benchmark [16] on two different cores.

It is important to note that peak power minimization is not the same as energy minimization. Much of the existing research has focused on *energy* minimization for real-time tasks on multi-core platforms [31, 33]. Energy minimization can be equivalent to *average* power minimization, whereas our focus is on *peak* power minimization. Also, for cores whose voltage and frequency are individually dynamically scalable (per-core DVFS), the problem of minimizing the peak power can be equivalent to the problem of minimizing the energy. That is, because the cores can be individually slowed down appropriately to finish the workload just in time, the core's workload gets distributed over the whole frame. This optimally suppresses any peaks in the power consumption, and also results in optimal reduction of the energy consumption because the cores are operated at the minimum feasible frequency throughout the frame. Results such as these are reported in literature, notably [31, 33]. However, due to monetary and chip-area costs, per-core DVFS is not feasible for many-core processors. Consider a system with thousands of cores, each one of them having its dedicated analog circuitry to control the voltage and frequency. To realize such a design would require not only a huge area on die, it might be financially unfeasible as well. Because of this, tiled architectures are becoming popular [72]. Each tile consists of a group of cores and the operating frequency is selectable at the granularity of a tile [72]. For such architectures, after the frequency of operation for a tile has been selected, the cores on that tile can be individually turned *on* or *off*; i.e., Dynamic Power Management (DPM) can be used to control the power consumption. For this, an appropriate scheduling of sleep time for each processing core on a tile can result in reduced peaks in the power consumption for that tile, thereby reducing the peak power consumption for the



(a) Peak Power of 8 Watts (b) Peak Power of 6 Watts

Figure 5.3: Motivational example of two different schedules.

whole chip. We address this problem later in this chapter.

In a nutshell, for a specific processing chip, the dark silicon problem consists of deciding the task partitioning and time schedules for the given cores, taking advantage of the hardware capabilities such that the physical power and temperature constraints are met, while satisfying the system's minimum required performance. We show this in the following simple example.

5.3.1 Motivational Example

Consider a many-core processor with four cores, i.e., c_1, c_2, c_3, and c_4, where each core executes tasks at 1 GHz. Assume that the power consumption for execution is 2 Watts per core, and that each core consumes 0 Watts when sleeping. Moreover, for simplicity in presentation, assume negligible overhead for sleeping. Under such hardware settings, consider that there are 4 real-time tasks arriving at time 0, i.e., τ_1, τ_2, τ_3, and τ_4. Each task needs to execute $7.5 \cdot 10^8$ computer cycles, and all tasks share a common deadline of 1 second, i.e., frame-based tasks. Although this results in a total utilization of 3, if task migration is not desired, then any task partitioning scheme will assign one task in each core.

Figure 5.3 shows two possible schedules where all tasks meet their deadlines. For the schedule in Figure 5.3a, all tasks start execution simultaneously at time 0, and all cores go to sleep after 0.75 seconds. This results in a peak power consumption of 8 Watts. On the other hand, by using DPM to control the sleep cycles of the cores, Figure 5.3b shows another possible schedule. For this second case, the peak power consumption is 6 Watts, from only activating 3 cores at any given time. Assume that TDP for this chip is 7 Watts. In this case, the first schedule will result in activation of DTM whereas the second one can avoid it. In the case of the first schedule, if the DTM is activated, it may trigger hardware-based performance throttling to keep the chip within thermally-safe operating conditions. This can result in the tasks missing their deadlines. Whereas, in the second schedule, DTM activation can be avoided. Here, once a schedule is selected for the real-time task set, a sufficient feasibility test can be designed which checks if the cumulative peak power of all cores will exceed the TDP for the chip. This example illustrates 1) the potential benefit of an appropriate coordinated schedule of sleep cycles of the cores in order to reduce the total peak power consumption, and, 2) the need to add an additional criterion of peak power consumption for checking the feasibility of a task set with real-time requirements.

Objective

Our goal here is to find a sleeping schedule for active cores on a fixed hardware platform, considering the heterogeneity of cores and tasks, such that the peak power is minimized to accommodate the TDP constraint, while guaranteeing that all real-time tasks meet their deadlines.

Our Contributions

For hard real-time tasks:

- We present a **p**eak **p**ower **m**anagement scheme (PPM). For the sake of completeness, we first deal with the task partitioning onto the available cores and determine the individual schedules for all tasks. Next, we schedule the sleep cycle, which is equivalent to the unconsumed utilization of each individual core. This is done in a way that peak power consumption is minimized, without violating the hard real-time requirements of the tasks.
- In addition to the existing utilization-based schedulability tests, we introduce the concept of a sufficient test for schedulability considering the peak power consumption of a task set with real-time requirements.
- An analysis of our scheme details how our solution determines the sleep cycle for individual cores, starting from *"homogeneous tasks on homogeneous cores"* to the most general case of *"heterogeneous tasks on heterogeneous cores"*.

- We simulate our PPM scheme using power traces collected from the 48-core SCC platform [72] and gem5 architecture simulator in combination with McPAT [96].

5.4 System Model and Problem Definition

We employ the commonly used system model for a heterogeneous multi-core system [14] as follows: Given a set of software processes (tasks), a set of processing entities (cores) upon which these tasks can execute, and the rate at which the processing cores execute the tasks and their corresponding power consumption values, our goal is to determine a mapping of the tasks onto the cores and determine a time schedule for the cores in such a way that the peak power consumption is minimized.

5.4.1 Task and Hardware Model

For the task model, we assume we have a set of n periodic tasks sharing the same arrival time and deadline, i.e. *frame-based tasks*, denoted as follows: $\mathbf{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$. All the tasks have same the period T (this task model is later extended to include more general *implicit deadline periodic tasks* in Section 5.9, where tasks do not necessarily share the same period). In each period, all the tasks have the same arrival time 0. We consider *partitioned scheduling*, in which each task is assigned onto a core, i.e., task migration among cores is not allowed. We also assume that all jobs are independent: they do not share resources, they do not have data dependencies, and that there is no interprocess communication. This model is not as restrictive as it appears, since there are ways to transform a set of dependent tasks to independent ones [81].

We focus on a multi-core system with a set of m heterogeneous cores, that is, $\mathbf{C} = \{c_1, c_2, \ldots, c_m\}$. Having a system with heterogeneous cores implies that tasks will have different execution times and power consumptions, depending on the core to which each task is mapped. Power consumption for execution consists of a dynamic and static component and cores can be put into sleep mode by gating the clock. We assume that changing the execution frequency of cores or setting them to sleep mode, and vice versa, takes a negligible amount of time, as it is accomplished through clock gating. During sleep mode, cores consume only static power, which might be different for each core. However, since static power is continuously being consumed, it only adds a constant offset to the peak power consumption of tasks (considering the tiled-architecture, only cores belonging to those tiles consume static power where at least one core is active, other tiles can be put to sleep). Hence, without loss of generality, we focus on dynamic power consumption and consider leakage a constant offset. Formally, we assume that the peak dynamic power consumption of task τ_i executing on core c_j is denoted as $p_{i,j}$, resulting in a peak power matrix $\mathbf{P} = [p_{i,j}]_{n \times m}$. Normally, during different execution phases, a task might have different power values, and considering only the peak value makes our approach pessimistic but safe.

Similarly, we define $u_{i,j}$ as the utilization of task τ_i executing in core c_j. That is, assume that task τ_i requires $x_{i,j}$ amount of time in the worst case when executing on core c_j to meet its deadline. For all i, the period of τ_i is T (frame-based tasks). Then, $u_{i,j} = \frac{x_{i,j}}{T}$. This results in a utilization matrix $\mathbf{U} = [u_{i,j}]_{n \times m}$. It holds that $0 \leq u_{i,j} \leq 1$, when $x_{i,j} \leq T$. If $x_{i,j} > T$, then we set $u_{i,j}$ to ∞ to guarantee that core c_j is not considered for placement of task τ_i.

Also, we assume the cores in the system have DPM capabilities, hence each core can be put into low-power mode individually, e.g., *idle*, *sleep*, *off*. The overhead for entering/leaving a low-power mode is considered negligible. Practically, this can be realized using commonly available mechanism of clock gating.

5.4.2 Problem Definition

For n frame-based tasks and a heterogeneous many-core platform with m cores, our objective here is to present a scheduling and mapping algorithm for executing all n tasks without violating their deadlines, while minimizing the total peak power consumption.

5.5 Solution Overview

To fulfill the aforementioned objective, we propose a novel peak power management scheme (PPM). Figure 5.4 shows the overview of our scheme and the taxonomy used. The solution presented in this chapter consists of two independent steps as discussed in detail in the subsequent sections. The first step deals with task partitioning into the cores and the individual schedule of each core. In the second step, we calculate a sleep cycle for individual cores in such a way that the peak power is minimized without affecting the individual schedules of cores. The second step is the key challenge targeted in this chapter. An in-depth discussion of these steps follows.

5.5.1 First Step: Task Partitioning

In the first step, as we consider partitioned scheduling, the mapping of the tasks onto cores must be decided. Performing partitioned scheduling to ensure the timing constraints has been researched extensively, in which the survey by Davis and Burns [40] provides a comprehensive study. However, as this part is related more to the feasibility of task partitioning, we present only the key concepts.

Deciding whether there exists a feasible task assignment for a task set with real-time constraints into multiple cores is an $\mathcal{NP}$-hard problem in the strong sense [14]. However, for example, the approximation algorithms by Graham [61] and Baruah [14] can be adopted to provide efficient and effective task partitioning for homogeneous and heterogeneous many-core processors, respectively. There exist other works in this direction as well [29]. Since task partitioning considering only the utilization values is already $\mathcal{NP}$-hard, we do not optimize for peak power consumption in this step.

The output of the task partitioning algorithms is a partition matrix $\mathbf{K} = [k_{i,j}]_{n\times m}$. For every element in the matrix, $k_{i,j}$ is set to 1 if task τ_i is to be executed on core c_j, otherwise 0.

The output of the next step (Section 5.5.2) is the matrix $\mathbf{A} = [a_{j,t}]_{m\times q}$. Every element in matrix $\mathbf{A}$, $a_{j,t}$, is set to i, if core j is set to execute task τ_i for time slot t and 0 otherwise.

Given:

$$\begin{array}{c|c} & \text{cores} \\ \text{tasks} & \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,m} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n,1} & u_{n,2} & \cdots & u_{n,m} \end{bmatrix} \end{array}$$

Utilization: $\mathbf{U}$

$$\begin{array}{c|c} & \text{cores} \\ \text{tasks} & \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,m} \end{bmatrix} \end{array}$$

Power: $\mathbf{P}$

Output Step 1:

$$\begin{array}{c|c} & \text{cores} \\ \text{tasks} & \begin{bmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,m} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n,1} & k_{n,2} & \cdots & k_{n,m} \end{bmatrix} \end{array}$$

Partition: $\mathbf{K}$

Output Step 2:

$$\begin{array}{c|c} & \text{time slots} \\ \text{cores} & \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,q} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,q} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,q} \end{bmatrix} \end{array}$$

Schedule: $\mathbf{A}$

Figure 5.4: Overview of steps for PPM and the taxonomy used.

It is not necessary to use all the given m cores for assigning the tasks; that is, the task partitioning may use fewer than m cores for assigning the tasks. If the time complexity is tolerable, the whole process (*Step 1* and *Step 2*) can be iteratively called to decide the number of cores for assigning the tasks. For the simplicity of presentation, we focus only on one iteration in which the number of cores for allocating tasks is fixed, and any known algorithm, e.g., [61, 14], for task partitioning is adopted. Moreover, the derived task partitioning also guarantees that the utilization of the tasks assigned on one core is less than or equal to 100%, in which adopting any workload-conserving scheduling policy (e.g., Earliest-Deadline-First (EDF)) in one core individually ensures that the tasks can meet the deadlines, due to the assumption of frame-based tasks. If the derived task partitioning has a core with utilization larger than 100%, either another more powerful task partitioning algorithm should be adopted or we should consider a bigger number of cores to partition the tasks.

5.5.2 Second Step: Sleep Schedule Decision

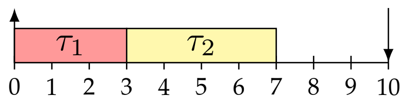
After the tasks are mapped onto the cores and every core's internal schedule is decided in *Step 1*, this phase decides the sleep schedule for each core, such that the internal schedule of the core remains unaffected and peak power is minimized. Please note that we are dealing with a frame-based task set hence all the tasks share the same arrival time and the same deadline. This gives us the flexibility to plan the sleep cycle for the core anywhere within the task period, as shown in the Figure 5.5.

Theorem 5.1. *If a frame-based, synchronous task set with period T and cumulative utilization U, can be feasibly scheduled on a single core, it can also be feasibly scheduled if the core is halted for at most $\lfloor T(1-U) \rfloor$ time in every T interval.*

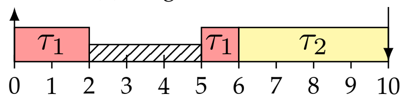
Proof. The system is either idle or executing jobs in time interval $[0, T)$. Since we focus on frame-based real-time tasks here, any schedule is feasible if the core is halted only by at most $T(1-U)$ amount of time, as the remaining time is used for executing jobs. □

Explanatory Example

Suppose that after partitioning, one of the processing cores has two tasks (τ_1, τ_2) with the period of 10 ms and worst-case execution times of 3 ms and 4 ms, respectively. The cumulative utilization is $\frac{3}{10} + \frac{4}{10} = 0.7$. As per Theorem 5.1, the processing core can be put to sleep for 30% of time and the task set can still be feasibly scheduled. A sample schedule is shown in Figure 5.5.



(a) Original schedule



(b) Schedule with sleep cycle

Figure 5.5: An example of modified schedule for frame-based tasks τ_1 and τ_2. Both tasks arrive at time 0 and have a deadline at time 10.

Based on the chip's architecture and task structure, we consider two observations.

1. *Task set heterogeneity*: The power consumption of a core caused by executing a task can either be task independent, or dependent. We call the former case *homogeneous* task set and the latter *heterogeneous* task set. For homogeneous task set, it holds that for all $i = 1, 2, \ldots, n-1$, $p_{i,j} = p_{i+1,j}$. In other words, all the rows in matrix $\mathbf{P}$ are identical.

2. *Core heterogeneity*: Like tasks, we classify the cores into two disjoint sets: *homogeneous* and *heterogeneous*. Homogeneous cores are those in which the power consumption of task τ_i is independent of the core where it is executed; i.e., for all $j = 1, 2, \ldots, m-1$, $p_{i,j} = p_{i,j+1}$ (all the columns in matrix $\mathbf{P}$ are same). Furthermore, this implies that all cores are equal in their capabilities, hence task τ_i has the same utilization on any core, i.e., for all $j = 1, 2, \ldots, m-1$, $u_{i,j} = u_{i,j+1}$ (all the columns of matrix $\mathbf{U}$ are same). Those cores that do not follow the above condition are heterogeneous cores.

These two conditions result in four possible cases. We discuss these cases one by one in the following sections and provide efficient solutions for each. It is to be noted here that, if claimed, optimality refers to the minimization of peak power after the tasks set has been partitioned, i.e., the notion of optimality is for peak power minimization considering only *Step 2*. As *Step 1* is already proven to be $\mathcal{NP}$-hard, a polynomial time solution for this step is unlikely to be found.

5.6 Homogeneous Tasks on Homogeneous Cores

In this section, we consider all the tasks are identical in their power consumption and differ from each other only in the utilization requirements. Also, all the cores have identical behavior for power consumption and computational performance output. This is the simplest case and will form the foundation for solving the complex cases that follow.

As we consider only homogeneous tasks being executed on homogeneous cores, it holds that, for all $p_{i,j} \in \mathbf{P}$, $p_{i,j} = p_{\text{const}}$. For this case, a time schedule with minimum peak power can be obtained by extending McNaughton's wrap-around rule [107] using core utilization values, as explained in more detail below.

Suppose that the core utilizations, for all m cores in the system, are represented through the vector $\mathbf{W} = [w_j]_m$. Using the partition matrix $\mathbf{K}$ and the utilization matrix $\mathbf{U}$, we can fill $\mathbf{W}$, such that for all $j = 1, 2, \ldots, m$, $w_j = \sum_{i=1}^{n} k_{i,j} \cdot u_{i,j}$. Clearly, for a feasible schedule, it should hold that for all $j = 1, 2, \ldots, m$, $w_j \leq 1$.

Using $\mathbf{W}$, we apply the wrap-around rule as follows. Assume that there are m bins of time $(b_1, b_2, \ldots, b_m)$, each of size T. The starting time for each is 0. Iteratively, we assign $T \cdot w_j$ time from bin b_k to core j, starting from b_1 and w_1. When core j is assigned time from bin b_k, the value of b_k is updated to $b_k - T \cdot w_j$. If, for core j, the time requirement cannot be fully satisfied from bin b_k, then we assign as much as possible from b_k so that b_k becomes zero, and the rest is assigned from b_{k+1}. As for all $w_j \in \mathbf{W}$, $w_j \leq 1$, a core gets time slices from at most two bins. The time slices assigned by this algorithm form the schedule of the core.

An example of assigning 3 cores having utilizations 0.5, 0.9 and 0.5 into 3 bins is presented in Figure 5.6. Here, c_1, having a utilization of 0.5, is completely assigned to the first half of b_1. In terms of the per-core DPM schedule, this allocation means that c_1 is turned on only in the beginning half of the frame, let's say T, then, from 0 to $0.5T$. c_2, with a utilization of 0.9, cannot be fully assigned to b_1, so it is partially assigned to the last half of b_1 and the rest is "wrapped-around" to the initial 40% of b_2. Correspondingly, for its DPM schedule, c_2 can be turned on in the initial 40% of the frame $(0 - 0.4T)$ and then in the last half of the frame $(0.5T - T)$. Similarly, c_3 is assigned time slot from $0.4T$ to $0.9T$, fulfilling its requirement of 0.5 utilization.

This is a polynomial time algorithm, with a time complexity of $\mathcal{O}(m)$ for m cores. Moreover, due to the wrap-around policy, it is also clear that at any time

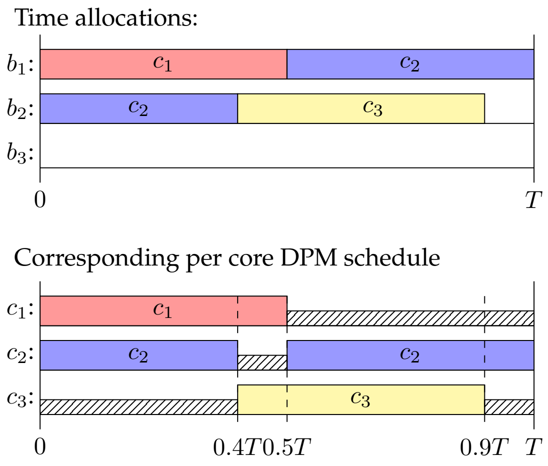


Figure 5.6: Example of wrap-around for 3 cores.

instant, the algorithm will activate at most c^* cores at the same time, where $c^* = \left\lceil \sum_{j=1}^{m} w_j \right\rceil$.

Clearly, the peak power consumption, π, of the tasks scheduled using the above presented rule is given as $\pi = p_{\text{const}} \cdot c^*$. Consequently, for testing the schedulability of the task set, if it holds that TDP $\geq \pi$, then this task set is feasible.

Theorem 5.2. *Given a set of homogeneous cores and a set of real-time frame-based tasks, at least $\lceil U \rceil$ processing cores must be simultaneously powered on to feasibly schedule a set of tasks with a cumulative utilization of U.*

Proof. Suppose that for a task set with cumulative utilization U, we have a feasible schedule in which the number of simultaneously activated cores is less than $\lceil U \rceil$. Then, it follows from the pigeon-hole principle that at least one of the cores must have a utilization of more than 1, therefore the task set is infeasible. This is a contradiction.

□

A trivial indication from Theorem 5.2 is that the wrap-around rule minimizes the number of simultaneously activated processing cores. Hence, it gives an optimal solution for the peak power minimization for homogeneous tasks scheduled on homogeneous cores, regardless of the task partitioning scheme used in *Step 1*.

Furthermore, this algorithm can also be used to approximate the cases in which the power consumption of the task set is non-homogeneous or the case in which the capabilities of the cores are heterogeneous.

5.7 Homogeneous Tasks on Heterogeneous Cores

In this case we assume that cores differ in their power consumption. This can be the result of manufacturing variations or complexity of the core due to added hardware accelerators. However, the power consumption is independent of the tasks. That is, for all $i \in \{2, \dots, m\}$ $p_{i,j} = p_{1,j}$ for every core j.

This case is particularly relevant to heterogeneous architectures where the cores differ so much from each other due to their capabilities that the power consumption profile of the cores is practically dependent only on the core being used; e.g., the popular ARM big.LITTLE architecture. [63].

To solve this case we present a greedy approach called Least Density First (LDF). This approach works as follows. First, we calculate the individual core utilizations, $\mathbf{W}$, as in Section 5.6. That is, using the partition matrix $\mathbf{K}$ and the utilization matrix $\mathbf{U}$, we can fill matrix $\mathbf{W}$, such that for all $j = 1, 2, \dots, m$, $w_j = \sum_{i=1}^{n} k_{i,j} \cdot u_{i,j}$. We start with the core that has the highest power consumption and assign it $\lceil w_j \cdot q \rceil$ slots from q total slots. After this, we update a density state vector, which is null initialized, by summing the total power consumption for all slots. For the next core, we again assign $\lceil w_j \cdot q \rceil$ slots with the lowest density so far, and update the density state vector, again. This is done iteratively for each core in $\mathbf{W}$. The pseudo code for this scheme is presented in Algorithm 3.

Formally, we set $\mathbf{P}'$ and $\mathbf{W}$ as follows for input to Algorithm 3:

- $\mathbf{P}'$ = per core power consumption, i.e., $\mathbf{P}' = [p'_j]$ where, for all $j \in \{1, \dots, m\}$, $p'_j = \sum_{i=1}^{n} k_{i,j} \cdot p_{i,j}$. The length of $\mathbf{P}'$ is m.
- $\mathbf{W}$ = per core utilization, i.e., $\mathbf{W} = [w_j]$ where, for all $j \in \{1, \dots, m\}$, $w_j = \sum_{i=1}^{n} k_{i,j} \cdot u_{i,j}$.

The output matrix, $\mathbf{A}$, from Algorithm 3 gives the schedule for the cores with the added sleep cycle. Due to the same power consumption for each task, for peak power consumption it does not matter in which sequence the tasks are executed on a core. Thus, we set the elements of $\mathbf{A}$ to either 1 or 0, where 0 means sleep time for the core and 1 active execution of tasks.

This schedule is feasible if the peak power consumption, π, is within the TDP. Here, π can be used as the basis for an offline, sufficient feasibility test for the

```
Algorithm 3: Least Density First (LDF)
Input:
  Per Core peak power consumption: $\mathbf{P}' = [p'_j]_{1\times m}$
  Per Core Utilizations: $\mathbf{W} = [w_j]_{1\times m}$
  Total time slots in a period: $q$
Output:
  Net peak: $\pi$
  Schedule assignment: $\mathbf{A} = [a_{j,t}]_{m\times q}$
Algorithm:
Density state: $\mathbf{D} = [d_t]_{1\times q} \leftarrow [0]_{1\times q}$ ;
$\mathbf{A} = [a_{j,t}]_{m\times q} \leftarrow [0]_{m\times q}$ ;
// Storing power values into an array and sorting
$\forall j \in (1,..,m)$ append $(p'_j, j)$ to $\mathbf{Q}$ ;
Sort $\mathbf{Q}$ according to $p'_j$ descending;
while Q not empty do
    Pop head of $\mathbf{Q} \rightarrow (p'_j, j)$;
    // Selecting the required lowest density slots in $S$
    $S = \{t : d_t$ is among $\lceil w_j \times q \rceil$ lowest density slot$\}$;
    for all $t \in S$ update $a_{j,t} = 1$;
    for all $t \in S$ update $d_t = d_t + p'_j$;
$\pi = \max(\mathbf{D})$;
```

task set. The problem of generating a schedule that minimizes the peak power is $\mathcal{NP}$-hard. This is formally proven in Theorem 5.3.

The working of this algorithm is further explained in the example shown in Figure 5.7. Here we have three cores, i.e., c_1, c_2 and c_3 with utilizations of $0.6, 0.7, 0.5$ and power consumption values of $3, 4, 2$ Watts. Assuming a frame-based taskset with the period of 10, the given utilization values translate to active times of $6, 7$ and 5 time slots for c_1, c_2 and c_3, respectively. To begin with, we choose the core with the highest power consumption, c_2, and assign it the required slots and update the density vector $\mathbf{D}$. Here, the height of the bars represents the density. After this, we choose the next-highest power consuming core and assign it the required number of lowest density slots and so on. This results in the per core schedule shown in the third iteration in the Figure 5.7, and the peak in power consumption is 7 Watts.

LDF is a polynomial time algorithm. We analyze its worst-case time complexity as follows. For execution of this algorithm, we start with two one-time operations having complexity q and mq for initializing $\mathbf{D}$ and $\mathbf{A}$, respectively. Next, we store the power consumption values per core in an array and sort it. The

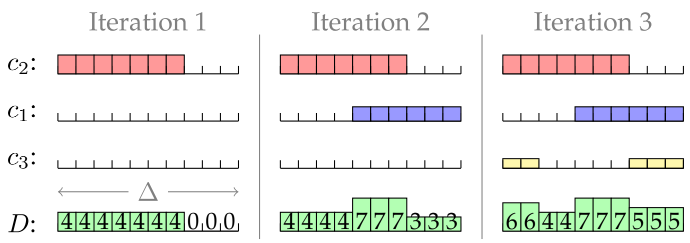


Figure 5.7: Example for least density first algorithm

time complexity of this operation is upper bounded by $m \log m$. Next, we enter the loop that executes m times for the following operations. Firstly, we remove the head of **Q**, i.e., constant complexity. In the second operation, we need to sort an array of maximum size q to select the required number of lowest valued elements. Hence its time complexity is upper bounded by $q \log q$. This is followed by two more operations, each having a worst-case complexity of q. Therefore, the total complexity can be given as $q+mq+m+m\log m+m(q\log q+q+q)+q$. This can be simplified to $\mathcal{O}(m \log m + mq \log q)$.

Theorem 5.3. *For a given task partitioning of homogeneous tasks onto heterogeneous cores, deriving a schedule for cores to optimally minimize the peak power is an $\mathcal{NP}$-hard problem.*

Proof. We reduce from the optimization version of the partitioning problem. Given a set of m numbers, the optimization version of the partitioning problem is to divide them into two disjoint subsets, A and A', such that the difference of the sums of the numbers in each subset is minimized. Such a problem is $\mathcal{NP}$-hard [56].

The reduction works as follows. Consider the special case of our problem in which we have m tasks with utilization 50%, every task has a different peak power consumption, and a deadline for the frame-based tasks of 1. The objective is to decide whether a core is executed in the window of $(0, 0.5]$ or $(0.5, 1]$, such that the final peak power consumption is minimized, which is the same as minimizing the difference between the peak power of the windows.

This problem is equivalent to the optimization version of the partitioning problem, where the peak power consumption of the tasks represents the numbers to partition. Deciding to execute a core in the window of $(0, 0.5]$ is equivalent to putting the number of its power consumption to set A. Similarly, deciding to execute a core in the window of $[0.5, 1)$ is equivalent to putting the number of its power consumption to set A'.

Therefore, an optimal solution to the optimization version of the partitioning problem is equivalent to an optimal solution to this special case of our studied problem and vice versa. Hence, we conclude the $\mathcal{NP}$-hardness of our studied problem. □

5.8 Heterogeneous Tasks on Heterogeneous/Homogeneous Cores

In this section, we present an algorithm for two cases; that is, 1) power consumption is dependent on the task but not on the core it executes on, and, 2) power consumption is dependent on the task as well as the core on which the task executes.

The former case applies to the wide variety of multi-core chips currently available in the market, where several identical cores are available on the chip and the power consumption profile of tasks differs due to the resource access pattern. The latter case is a generalization to include many-core chips, such that a core might consume less power for one task and more power for another task, and there might exist another core in the system whose power consumption values are reversed for the same two tasks. This is the most general case, with all the previous cases being a special form of this case.

We present a variation of the Least Density First algorithm to solve both these cases. The variation from the Least Density First algorithm presented earlier is as follows. In the normal version of LDF the only decision criterion is based on density. However, in this algorithm, we also check the assignment of the slots. We start with the highest power consuming task and assign it the lowest density *free* slots on the core in which this task has been partitioned, and this process is repeated for all tasks. The extra step of checking the free slots is necessary as the globally lowest-density slots might already be occupied in the core of interest. Since we are only considering partitioned scheduling, i.e., each task is assigned to only one core, we do not need to check for the condition in which the same task is concurrently scheduled at two cores. The pseudo code for the procedure is presented in Algorithm 4.

Like in the previous section, the matrix A gives us the schedule of the cores. This schedule is feasible if π is less than or equal to the TDP.

This is also a polynomial time algorithm. We analyze its worst case time complexity as follows. The algorithm starts with two one-time operations of complexity q and mq, as before. Next, we conduct mn multiply operations and store non-zero values $\mathbf{Q}$ which can, at most, have as many members as total number

Algorithm 4: LDF with occupancy check

```
Input:
  Power matrix: $\mathbf{P} = [p_{i,j}]_{n\times m}$
  Partition matrix: $\mathbf{K} = [k_{i,j}]_{n\times m}$
  Utilization matrix: $\mathbf{U} = [u_{i,j}]_{n\times m}$
  Time slots in a period: $q$
Output:
  Peak power consumption: $\pi$
  Schedule assignment: $\mathbf{A} = [a_{j,t}]_{m\times q}$
Algorithm:
Density state: $\mathbf{D} = [d_t]_{1\times q} \leftarrow [0]_{1\times q}$ ;
$\mathbf{A} = [a_{j,t}]_{m\times q} \leftarrow [0]_{m\times q}$ ;
// Storing non-zero power values into an array
$\forall i \forall j$ if $(k_{i,j} \cdot p_{i,j}) \neq 0,$ append $(p_{i,j}, i, j)$ to Q;
Sort Q descending according to $p_{i,j}$;
while Q not empty do
    Pop head of Q $\rightarrow (p_{i,j}, i, j)$;
    // Among free slots, choosing required number of lowest
       density slots in set $S$
    $S = \{t : a_{j,t} = 0 \ \wedge \ d_t$ is among $\lceil k_{i,j} \cdot u_{i,j} \cdot q \rceil$ lowest density slots$\}$;
    for all $t \in S$ update $a_{j,t} = i$;
    for all $t \in S$ update $d_t = d_t + p_{i,j}$;
$\pi = \max(\mathbf{D})$;
```

of tasks, i.e., n. This is followed by sorting of **Q** with the complexity of $n \log n$. Then, we run the loop n times in which the following operations are performed. The first operation is to pop the first element of **Q**. This can be performed in constant time. The second operation consists of q comparisons to check for free slots, after which we choose the lowest valued elements from an array of q elements, i.e., the time complexity is upper bounded by $q \log q$. This is followed by two more operations, each having a worst case complexity of q. Therefore the total complexity can be given as $q + mq + mn + n \log n + n(q + q \log q + q + q) + q$. This can be simplified as $\mathcal{O}(mq + mn + n \log n + nq \log q)$.

5.9 Implicit Deadline Periodic Tasks

The solutions presented in Sections 5.6, 5.7 and 5.8 are applicable only to frame-based task sets. In this section, we present a method to extend the previous results to include implicit deadline periodic task sets. We include the following task types in the task model. Each task τ_i releases an infinite number of task instances (jobs) with period T_i and relative deadline D_i, where $D_i = T_i$. We assume the tasks are synchronized, that is, the first job of each task arrives at the same instant. Frame-based tasks are a special form of implicit deadline periodic tasks.

The core idea here is that an implicit deadline real-time task set can be feasibly scheduled as long as its utilization on each core does not exceed 100%. Therefore, the slack left after task partitioning can be reclaimed to halt the processing cores in a coordinated manner to decrease the peaks in power consumption.

Theorem 5.4. *A feasible, implicit deadline task set with periods $\{T_1, T_2, \cdots, T_n\}$, with cumulative utilization U can be feasibly scheduled with* Earlier-Deadline-First *(EDF) policy, if the processing core is put to sleep for $(1 - U)$ fraction of time in every Δ interval, where Δ is the greatest common divisor of $(T_1, T_2, \cdots, T_n)$ and is synchronized with the tasks.*

Proof. Consider an implicit deadline task set with periods $\{T_1, \cdots, T_n\}$ and worst-case execution times $\{C_1, \cdots, C_n\}$ which is feasibly schedulable with EDF policy. Consider that the sleep time of the processing core is an additional implicit deadline task, τ_s, with period Δ and execution time of $\Delta(1-U)$, where $U = \sum_{j=1}^{n} \frac{C_j}{T_j}$.

Assume that after introducing τ_s, the system cannot be feasibly scheduled using EDF policy and a job, $J_{i,k}$, of a task, τ_i, misses its absolute deadline, $d_{i,k}$. Suppose that t_0 is the last instant before $d_{i,k}$ when the system was idle. If such an instant does not exist, then t_0 denotes the starting time of the system. Since the system cannot be feasibly scheduled with EDF, then it must hold that:

$$\begin{aligned} d_{i,k} - t_0 &< \sum_{j=1}^{n} \left\lfloor \frac{d_{i,k} - t_0}{T_j} \right\rfloor C_j + \left\lfloor \frac{d_{i,k} - t_0}{\Delta} \right\rfloor \Delta(1-U) \\ \implies 1 &< \sum_{j=1}^{n} \frac{C_j}{T_j} + (1-U) \\ \implies 1 &< U + 1 - U \\ \implies 1 &< 1 \end{aligned}$$

We reach a contradiction in which the assumption that a task set, originally feasibly schedulable with EDF policy, becomes infeasible with the addition of τ_s with period Δ and execution time $\Delta(1-U)$, is invalid and the theorem is proven. □

Applicability of the earlier presented algorithms to implicit deadline periodic tasks is a more general result, but specifically in this case an overhead can be expected due to the additional switching transitions to and from the sleep mode. Since Δ is the greatest common divisor of the periods of all tasks, it can be small, and, as a sleep cycle has to be placed in each Δ, this can make the system infeasible if there is a timing overhead associated with putting the system to sleep. However, this overhead can be easily avoided by implementing the sleep mode using clock gating instead of powering down the system. Clock gating for conserving energy consumption has been well researched [43].

For periodic tasks, in the case of homogeneous tasks on homogeneous cores, the feasibility of real-time constraints originating from peak power consumption can be verified by the same test as introduced in Section 5.6. The peak power is given as $\pi = \lceil U \rceil \cdot p_{\text{const}}$, where U is the cumulative utilization and p_{const} is the power consumption of any core.

For the most general case, i.e., heterogeneity of either task set or both the task set and the processing cores, the test introduced in Section 5.8 for verifying the feasibility of real-time constraints for the second version of LDF (Algorithm 4) can be utilized repeatedly. Since there is a sleep period included in the schedule for every core in the system in every Δ interval, we can use LDF (Algorithm 4) within each Δ interval to coordinate the sleep periods of cores to minimize peak power. Peak power, π, for each Δ is known from Algorithm 4. To find the highest peak that can occur, the system must be analyzed for an interval equal to the hyper period of the system. A hyper period is that interval after which the system repeats itself and is equivalent to the least common multiple of the periods of the task set. To summarize, we employ the task partitioning algorithm as discussed previously (Section 5.5.1) and use Algorithm 4 with $\mathbf{P}$, $\mathbf{K}$ and $\mathbf{U}$ matrices obtained from the partitioning. Here, we set the number of slots, q, equal to Δ. Algorithm 4 returns the value of peak, π for this Δ. The same process is repeated for the next Δ intervals until the total analyzed period equals a hyper period. The highest peak among all Δ intervals belonging to the hyper period is used to determine the feasibility.

Similarly, for homogeneous tasks on heterogeneous cores, Algorithm 3 can be utilized repeatedly for one complete hyper period and the highest value of π can be compared against TDP for feasibility testing.

5.10 Results and Discussion

In this section, we present the results of our simulations. To evaluate our scheme, we use applications from the *Parsec* benchmark suite [16] running on

Table 5.1: Specifications for applications used in simulations

Application	Period [ms]	Dynamic power usage (W)	
		P54C	Alpha
x264	30	0.70	0.66
bodytrack	30	1.00	0.81
swaptions	450	0.60	0.74
blacksholes	900	0.50	0.70

the SCC platform and Alpha core that we simulated using gem5 and McPAT infrastructure. To highlight the difference between peak power minimization and average power minimization, i.e., energy minimization, we compare the presented algorithms with a well known energy minimization scheme. The details of the setup and the results are presented below.

5.10.1 Platform details

We use Intel's 48-core SCC platform [72] for power measurements. This platform is equipped with 48 Pentium (P54C) cores which are based on a 45 nm manufacturing process. The cores are distributed into evenly placed 24 tiles with 2 cores per tile. A network on chip in mesh topology allows inter-core communication. We use four applications from the *Parsec* benchmark suite [16] and obtain the peak power consumption individually for each, using the on-board instrumentation. Details of the applications used to collect the power traces are presented in the next section. SCC offers a tiled architecture with so-called voltage and frequency islands. The voltage can only be changed at the granularity of 8 cores and frequency at the granularity of 2 cores. As a precursor to future many-core processors, it shows that DPM based power control will remain an essential 'control knob' in future chips.

Since the SCC platform has homogeneous cores only, it cannot be used to measure the effect of core-heterogeneity. For this, we simulated a synthetic platform based on SCC's architecture and same dimensions, but with 24 Alpha cores [106] replacing 24 Pentium cores, one in each tile. Alpha cores are also based on the 45 nm manufacturing process and are simulated using gem5 [17]. The peak power measurements for the Alpha cores are obtained through simulation using McPAT. Alpha and Pentium cores differ in their power consumption, as well as computational performance, but are based on the same manufacturing technology. This makes them a good candidates for judging the efficacy of this work. We run the same applications on both types of cores to measure the power profiles for each application.

5.10.2 Real-time Workload

We generate synthetic workload for the simulations. Our scheme of workload generation is based on the widely used technique in the real-time community presented in [10]. We generate a randomized task set, where each task has three parameters: 1) utilization, 2) peak power, and 3) period. These three parameters are obtained as follows. For each task, the utilization is assigned randomly in the interval $(0, u_l)$, where u_l is the minimum of 1 and left over utilization in the platform. The peak power consumption is chosen among the actual measurements that were performed using the SCC platform and simulated Alpha core on gem5 architecture simulator. We use the traces collected from four applications from the *Parsec* benchmark suite [16] to sample their peak power consumption as shown in Table 5.1. To reduce evaluation time, we randomly selected four of the 13 applications in the Parsec benchmark suite and executed their single-core versions. Here, *x264* is an H.264 video encoder, *bodytrack* is a computer vision application that tracks human body whereas *swaptions* and *blacksholes* are financial analyses applications.

For the third parameter (the period of the real-time workload), we select the period for all tasks to be 30ms, in order to simulate a frame-based task set. Next, we use the actual periods as mentioned in Table 5.1 for implicit deadline periodic tasks. After generating the task set, we use the partitioning algorithm, presented in [14], to partition the tasks into the available cores. Since this algorithm guarantees to find a feasible mapping if, at most, half of the processing capacity is utilized, we use the system utilization of 24, where not mentioned otherwise.

5.10.3 Baseline Scheme

To compare our scheme, we use a well-known energy minimization procrastination scheme presented in [77]. The schedule obtained using the method of [77] optimally minimizes the energy consumption for tasks with real-time requirements. In essence, the energy minimization scheme procrastinates the tasks as much as possible without violating the performance constraints. In its original form, it activates all the cores toward the end of the schedule, causing a peak in the power consumption at the end of the period for frame-based tasks. To suppress this peak towards the end of the schedule, we *modify* the scheme by keeping m^* cores activated through out the period, starting with the most power consuming cores, and activating more in the end to meet performance requirements. Here, $m^* = \left\lfloor \sum_{j=1}^{m} \sum_{i=1}^{n} u_{i,j} \cdot k_{i,j} \right\rfloor$, i.e., the floor of the total system utilization. By keeping a subset of cores activated throughout the period, the power consumption is distributed over the length of the entire period and the peaks in power consumption are suppressed. This provides the basis for fair comparison.

5.10.4 Results

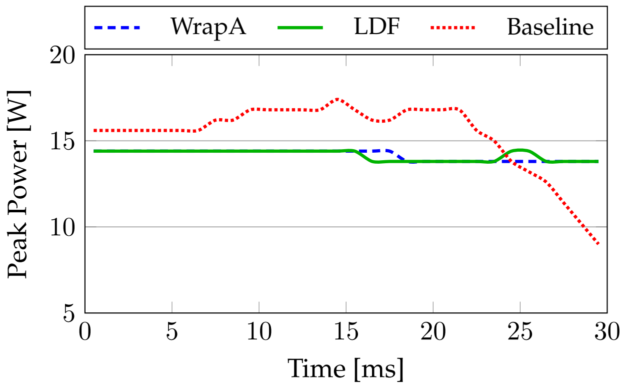
Our focus in this chapter has been on minimizing peak power. We present two important results in this regard.

First, we show a simple comparison of PPM against a well-known baseline scheme for energy minimization for both varieties of tasks and cores. We generate the schedule required for the different combinations of applications, considering their real-time requirements using our scheme and the baseline. Initially, we consider frame-based tasks with the period to be 30ms. To simulate the case of homogeneous task sets, we use only *swaptions* on all cores. To simulate the case of homogeneous cores, we use only the power data from the SCC platform.

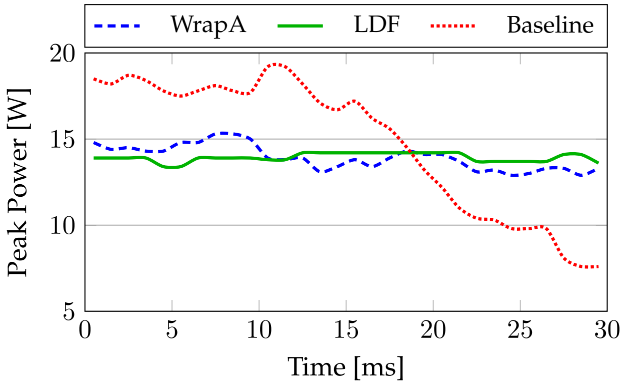
For frame-based tasks, the results are summarized in Figures 5.8 and 5.9. Here we can see that in all four cases our scheme produces a more balanced power consumption profile, as compared to the baseline. In the case of heterogeneous tasks or heterogeneous cores, since we prefer the least dense slots (LDF), the power consumption of our scheme gets distributed over the whole period. This helps to avoid peaks.

The wrap-around method (WrapA) was basically designed for homogeneous tasks on homogeneous cores and it solves this case optimally. In Figure 5.8a, it can be seen that the peak produced by WrapA is not higher than that of LDF, although not at the same point in the schedule. WrapA achieves a peak power consumption of 3 Watts less than the baseline scheme. In the rest of the three cases, that is, when either tasks or cores are non-homogeneous in their power consumption (Figures 5.8b, 5.9a and 5.9b), its optimality is not guaranteed. But, this method still fares better than the baseline scheme and its maximum deviation from the LDF remains less than 10%. The greedy approach employed in the LDF method achieves better results than both the baseline scheme and WrapA, in this case.

In the second case, we evaluate the effect of increasing the load on the peak power. The results are summarized in Figure 5.10. In this case, we only consider the general case of heterogeneous tasks on heterogeneous cores. As ex-



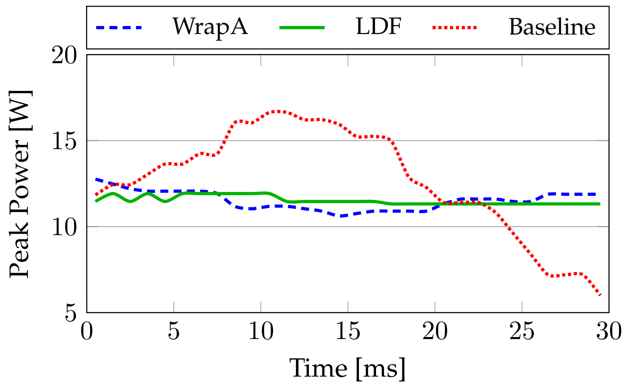
(a) homogeneous tasks



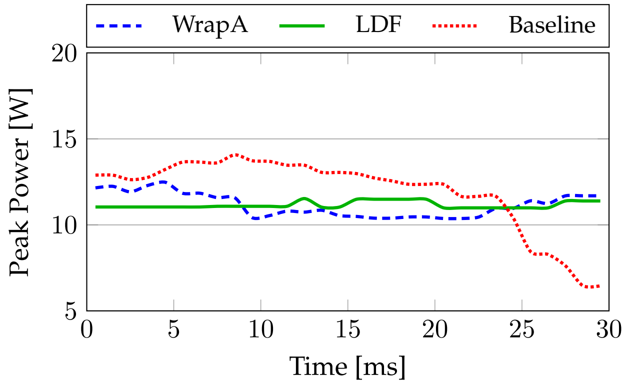
(b) heterogeneous tasks

Figure 5.8: Power consumption profile for each period for homogeneous cores (frame-based tasks)

pected, it can be seen that as the workload increases, the peak in power starts to grow for all three schemes. Here again, we observed that the maximum deviation of WrapA method does not exceed 10% (1.58 Watts) from the value achieved by LDF, whereas baseline scheme deviates up to 35.5% (6.08 Watts). The maximum deviation for both WrapA and the baseline scheme was observed at the total chip utilization of 35, with a general trend of higher deviations with increasing utilizations for both.



(a) homogeneous tasks



(b) heterogeneous tasks

Figure 5.9: Power consumption profile for each period for heterogeneous cores (frame-based tasks)

Non Frame-based, Periodic Tasks

The results for the non frame-based periodic tasks are presented in Figures 5.11 and 5.12. Here we use the periods and power profiles for the tasks as mentioned in Table 5.1. We assign the utilization randomly, according to the methodology introduced in [10]. For periodic tasks, we only present the results for non-homogeneous task sets on both homogeneous and heterogeneous cores, hence, we employ Algorithm 4 here. In this case, the difference between the *energy* minimization vs. *peak* power minimization becomes quite apparent. The baseline scheme used is designed with the perspective of energy minimization. In Figure 5.12, the baseline scheme's peak power consumption is 5 Watts more

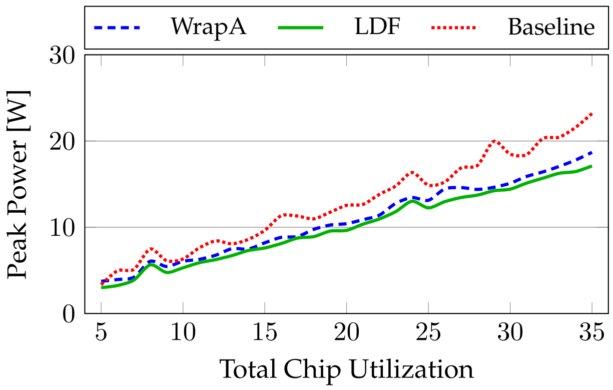


Figure 5.10: Effect of increasing workload on peak power consumption

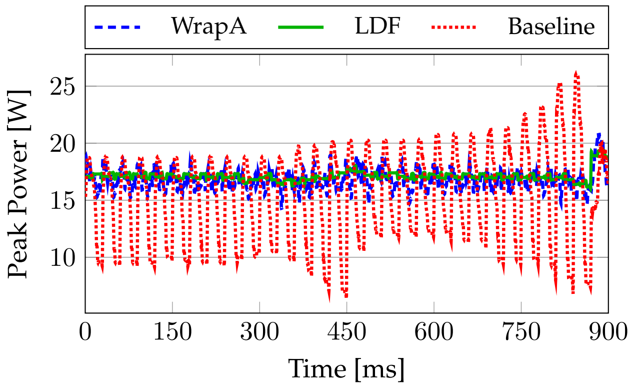


Figure 5.11: Periodic workload: heterogeneous tasks on homogeneous cores

than LDF and 3 Watts more than WrapA, whereas baseline consumes less energy, as shown in Figure 5.13. In the case of homogeneous cores (Figure 5.11), the difference in peak power consumption is even bigger. In both cases, LDF and WrapA keep a more balanced power profile, as in the case of frame based tasks. They are able to suppress the peaks, especially toward the end of the schedule, where baseline scheme causes a peak in order to fulfill the performance requirements of real-time tasks that have been back-logging during the procrastination. The peaks caused at every 30, 450 and 900 ms correspond to the periods of the tasks used for evaluation.

In Figure 5.12, the baseline scheme shows an oscillating pattern with the peaks being 30 ms apart. There are also clear bulges at the periods of 450 ms and

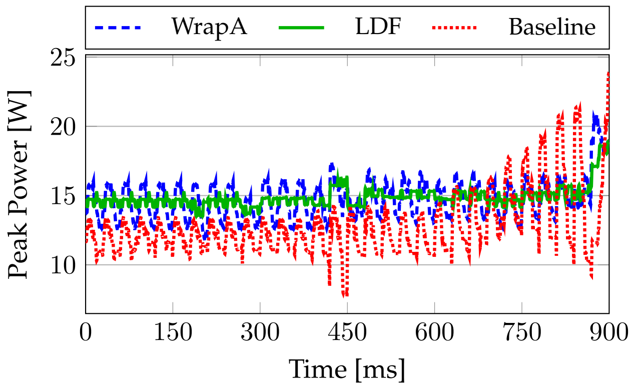


Figure 5.12: Periodic workload: heterogeneous tasks on heterogeneous cores

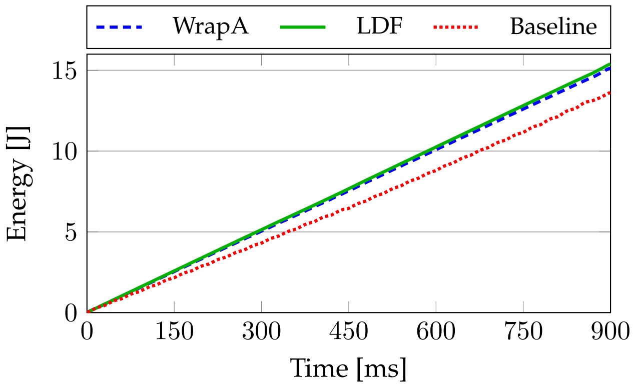


Figure 5.13: Cumulative energy consumption: Periodic workload, heterogeneous tasks on homogeneous cores

900 ms, whereas LDF and WrapA show much smoother trends. This also highlights another fundamental difference between energy minimization (i.e., average power minimization) and peak power minimization for many-core processors. For the former no coordination among the individual cores is required and each core, independently, decides its procrastination duration. Whereas, for the peak minimization, some form of coordination or synchronization is essential. Since, in the case of LDF and WrapA, cores' sleep cycles are coordinated, they result in a more balanced power profile in comparison with the baseline scheme.

5.11 Summary

In this chapter, we presented the congruence between the problem of peak power management in MapReduce clusters and the many-core architectures for executing the workload that has real-time constraints. We presented a solution to minimize peak power consumption for executing real-time tasks on many-core architectures or, equivalently in MapReduce clusters, which can help contain the power consumption within the TDP constraint. We argued that the peak power consumption of a real-time task set must also be verified when determining its scheduling feasibility. The presented peak power management scheme follows a two step procedure: Firstly, the tasks are partitioned on to the available cores and the schedule for each core is calculated. Afterward, our solution heuristically minimizes the peak power consumption for systems with *homogeneous tasks on homogeneous cores*, *either heterogeneous cores or tasks*, and with both *heterogeneous tasks on heterogeneous cores*. This is achieved by putting the cores into sleep mode at appropriate points in time, without affecting the tardiness of the real-time tasks. For this purpose, we presented algorithms with polynomial time complexity. We simulated our scheme using the power traces for two platforms: SCC and a heterogeneous core platform based on SCC design. Our results show the efficacy of our scheme in terms of peak power minimization.

6 Conclusion and Outlook

Environmental concerns are gaining importance, and consequently, the drive to reduce the anthropogenic environmental footprint is gathering momentum. This push toward environmental friendliness is powered by, among others, the evolving legislative landscape around the world to counter the major sources of pollution and global warming. One of the biggest contributors to environmental harm still remains the electricity generation process.

As large-scale consumers of electricity, global Internet services contribute significantly toward the environmental pollution. Such services often employ hundreds of thousands of servers at multiple locations around the globe and have high requirements of reliability, robustness and timeliness. Any attempt to optimize this huge system needs to consider a multitude of factors, including electricity consumption for cooling and infrastructure, fluctuating usage patterns, correlation of geographical distances, timeliness of responses via SLAs, among others. Spatially and temporally varying electricity prices that are affected by weather patterns, storage options and regulatory issues, in addition to demand and supply dynamics, add to the overall complexity. In this dissertation, we attempt to tackle these challenges in order to optimize the electricity consumption of such services.

In this chapter, we present a short summary of our contributions in Section 6.1 and identify areas for future investigations in Section 6.2.

6.1 Contribution Summary

Our goal, in the scope of this dissertation, was to present a model-based design methodology to optimize electricity usage in one of the fastest growing major consumers of electricity in the ICT sector, i.e. large-scale multi-site Internet services. We presented techniques to decrease the environmental footprint of such services while respecting budgetary and performance constraints. In this regard, our contributions fall into three areas, as follows.

In Chapter 3, we focused on maximizing the usage of green energy in a typical multi-site Internet service. We explored the gains that can be obtained by exploiting the temporal and spatial fluctuations in electricity prices and carbon market dynamics, while respecting budget and SLA constraints. We showed that this is an $\mathcal{NP}$-hard problem and presented a viable greedy heuristic for optimization. Our approach to increasing the share of green energy consisted of two steps: Intra DC optimization and a central optimization. This resulted in the basis for a general, holistic solution that can include factors like power required for cooling, power consumption of networking infrastructure, on-site renewable energy generation systems, multiple services with multiple SLAs and an intermix of heterogeneous and homogeneous server-based DCs. We evaluated these solutions with traces of electricity prices and typical Internet workloads. Our results showed that the environmental footprint of a large-scale Internet service can be decreased substantially by properly exploiting the variability of electricity prices and green energy production.

In Chapter 4, we focused on an electrical demand shifting problem, aiming to achieve financial and environmental benefits. In particular, we tackled the problem of peak minimization using a battery as an energy storage element. Including a battery buffer for energy storage at an intermediate or an end-consumer site can result in cost savings by offering the flexibility to defer or advance the demand to the period of lower cost. A controller is needed to decide when and how much energy to store or consume from the storage at any given time. We presented an analysis technique based on network calculus to help determine the most appropriate battery size for a system by analyzing the load traces from the past for this system. The technique, however, is limited to a class of controllers called 'monotonic' controllers. The presented scheme lays the groundwork for analyzing more general controllers, as well as 'load shifting', which is a more general form of peak minimization and more appropriate for integrating renewable energy sources into the electrical grid.

Finally, in Chapter 5, we concentrated on Intra DC level power optimizations (in contrast to the whole Internet service level). Particularly, we discussed the strong resemblance of MapReduce, a popular software model used in DCs, to the many-core processing chips. We presented a scheme to minimize peak power consumption for executing real-time tasks on many-core architectures and argued that the peak power consumption of a real-time task set must also be verified when deciding its scheduling feasibility. The scheme we presented addresses all possible cases that are expected to occur on many-core chips; that is, 1) homogeneous tasks on homogeneous cores, 2) either heterogeneous cores or tasks, and 3) with both heterogeneous tasks on heterogeneous cores. For this, we presented algorithms with polynomial time complexity and simulated our scheme using the power traces for two platforms: 48 core Intel's SCC and a heterogeneous core platform based on SCC design. Our results show the efficacy of our scheme.

6.2 Future Outlook

Energy and power management within Internet services, in particular, and ICT, in general, is an active topic of interest that is of growing importance. Because of the many variables involved, there are also many possible optimizations that can be exploited to reduce the environmental footprint and the cost of operation for large-scale Internet services.

In this section, we present some possible future directions for exploration that can be followed to build on the techniques presented in this dissertation.

Market Heterogeneity

Not unlike other commodities, electricity markets show inherent differences and peculiarities based on the differences of regional and legacy factors. The differences among electricity markets in various regions of the world include not only price differences originating from disparities in available electricity sources, but also the organizational and legislative dissimilarities. Regional legislation and regulatory principles have a bigger impact on the electricity market than physical factors, such as fuel availability. Because the earth's environment, however, is a shared resource, maximizing the usage of green energy is beneficial for the whole world, irrespective of where it is done. Hence, it will be advantageous to customize the presented green energy maximization scheme in the scope of other electricity markets, such as Europe and China.

Introducing Elasticity in Electrical Demand

In this dissertation, we concentrated on the special case of demand-shifting mechanisms, i.e., the peak minimization scheme in the electricity consumption. This holds merit for the majority of the world's conventional power generation setups, which are still powered by fossil fuel and carry negligible contributions from renewable energy sources. However, the share of the electricity derived from green energy sources is increasing, particularly in developed regions such as Europe. Hence, considering the intermittent nature of renewable energy sources, the next logical step is to find mechanisms that can 'mould' the electrical load profile to better fit the expected availability curve of renewable energy sources. In theory, this can even mean peak generation instead of peak shaving. For this, an extension to the peak shaving scheme presented in Chapter 4, is a logical next step.

Although this dissertation presented various model-based schemes to decrease the environmental footprint of Internet services in particular, and large consumers of electricity in general, there is significant room for future work in this direction.

List of Figures

List of Tables

Bibliography

[1] HA Aalami, M Parsa Moghaddam, and GR Yousefi. "Demand response modeling considering Interruptible/Curtailable loads and capacity market programs". In: *Applied Energy* 87.1 (2010), pp. 243–250.

[2] Nilmini Abeyratne, Reetuparna Das, Qingkun Li, Korey Sewell, Bharan Giridhar, Ronald G. Dreslinski, David Blaauw, and Trevor N. Mudge. "Scaling towards kilo-core processors with asymmetric high-radix topologies". In: *19th IEEE International Symposium on High Performance Computer Architecture, HPCA 2013, Shenzhen, China, February 23-27, 2013*. IEEE Computer Society, 2013, pp. 496–507. DOI: 10.1109/HPCA.2013.6522344.

[3] *AES to Help SCE Meet Local Power Reliability with PPA for 100 MW of Energy Storage in California*. [press release]. Accessed: 2016-09-02. Nov. 2014. URL: http://aesenergystorage.com/2014/11/05/aes-help-sce-meet-local-power-reliability-20-year-power-purchase-agreement-energy-storage-california-new-facility-will-provide-100-mw-interconnected-storage-equivalent-200-mw/.

[4] Markus Ahrens, Ladislav Kucera, and Rene Larsonneur. "Performance of a magnetically suspended flywheel energy storage device". In: *Control Systems Technology, IEEE Transactions on* 4.5 (1996), pp. 494–502.

[5] Mohamed H Albadi and EF El-Saadany. "Demand response in electricity markets: An overview". In: *IEEE power engineering society general meeting*. Vol. 2007. 2007, pp. 1–5.

[6] Amin Amini. "A Novel Approach to Forecast and Manage Electrical Maximum Demand". MA thesis. Purdue University Indianapolis, Indiana, Aug. 2017. URL: https://scholarworks.iupui.edu/bitstream/handle/1805/12825/AminAmini.pdf.

[7] Mike Andrea and Byron Wallace. "White Paper: Data Center Standards, Data Center Size And Density". In: *The Strategic Directions Group Pty Ltd* (Oct. 2014). URL: http://www.afcom.com/data-center-size-density-white-paper-download/.

[8] Apple Inc. *Apple Facilities Environmental Report*. 2012. URL: http://images.apple.com/environment/reports/docs/Apple_Facilities_Report_2012.pdf.

[9] Nikolay Archak, Vahab S. Mirrokni, and S. Muthukrishnan. "Mining advertiser-specific user behavior using adfactors". In: *Proceedings of the 19th International Conference on World Wide Web, (WWW 2010), Raleigh, North Carolina, USA, April 26-30*. Ed. by Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti. ACM, 2010, pp. 31–40. DOI: 10.1145/1772690.1772695.

[10] T. P. Baker. *Comparison of empirical success rates of global vs. paritioned fixed-priority and EDF scheduling for hard real time*. Tech. rep. TR-050601, 2005.

[11] Amotz Bar-Noy, Matthew P. Johnson, and Ou Liu. "Peak shaving through resource buffering". In: *Approximation and Online Algorithms* (2009), pp. 147–159.

[12] Amotz Bar-Noy, Yi Feng, Matthew P. Johnson, and Ou Liu. "When to Reap and When to Sow - Lowering Peak Usage with Realistic Batteries". In: *Experimental Algorithms, 7th International Workshop, WEA 2008, Provincetown, MA, USA, May 30-June 1, Proceedings*. 2008, pp. 194–207. DOI: 10.1007/978-3-540-68552-4_15.

[13] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition*. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2013. ISBN: 9781627050098.

[14] Sanjoy K. Baruah. "Task Partitioning Upon Heterogeneous Multiprocessor Platforms". In: *10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004), 25-28 May, Toronto, Canada*. IEEE Computer Society, 2004, pp. 536–543. DOI: 10.1109/RTTAS.2004.1317301.

[15] Mohamed Ben-Daya, M Darwish, and K Ertogral. "The joint economic lot sizing problem: Review and extensions". In: *European Journal of Operational Research* 185.2 (2008), pp. 726–742.

[16] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. "The PARSEC benchmark suite: characterization and architectural implications". In: *17th International Conference on Parallel Architecture and Compilation Techniques (PACT 2008), Toronto, Ontario, Canada, October 25-29*. Ed. by Andreas Moshovos, David Tarditi, and Kunle Olukotun. ACM, 2008, pp. 72–81. DOI: 10.1145/1454115.1454128.

[17] Nathan L. Binkert, Bradford M. Beckmann, Gabriel Black, Steven K. Reinhardt, Ali G. Saidi, Arkaprava Basu, Joel Hestness, Derek Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. "The gem5 simulator". In: *SIGARCH Computer Architecture News* 39.2 (2011), pp. 1–7. DOI: 10.1145/2024716.2024718.

[18] Spyros Blanas, Jignesh M. Patel, Vuk Ercegovac, Jun Rao, Eugene J. Shekita, and Yuanyuan Tian. "A Comparison of Join Algorithms for Log Processing in MaPreduce". In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. SIGMOD '10. Indianapolis, Indiana, USA: ACM, 2010, pp. 975–986. ISBN: 978-1-4503-0032-2. DOI: 10.1145/1807167.1807273.

[19] Tom Boden, Bob Andres, and Gregg Marland. "Global CO_2 Emissions from Fossil-Fuel Burning, Cement Manufacture, and Gas Flaring: 1751-2011". In: *Carbon Dioxide Information Analysis Center, Oak Ridge National Laboratory, US Department of Energy* (2015). URL: http://cdiac.ornl.gov/ftp/ndp030/global.1751_2011.ems.

[20] B. Bohnenstiehl, A. Stillmaker, J. Pimentel, T. Andreas, B. Liu, A. Tran, E. Adeagbo, and B. Baas. "A 5.8 pJ/Op 115 Billion Ops/sec, to 1.78 Trillion Ops/sec 32 nm 1000-Processor Array". In: *Symposium on VLSI Circuits*. Honolulu, HI, June 2016.

[21] Haseeb Bokhari, Haris Javaid, Muhammad Shafique, Jörg Henkel, and Sri Parameswaran. "darknoc: Designing energy-efficient network-on-chip with multi-vt cells for dark silicon". In: *Proceedings of the 51st Annual Design Automation Conference*. ACM. 2014, pp. 1–6.

[22] Brian Bonlender. *State of the Data Center Industry*. Tech. rep. Department of Commerce, State of Washington, Jan. 2018.

[23] Joos-Hendrik Böse, Artur Andrzejak, and Mikael Högqvist. "Beyond online aggregation: parallel and incremental data mining with online map-reduce". In: *MDAC '10: Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud*. Raleigh, North Carolina, USA: ACM, 2010. ISBN: 978-1-60558-991-6.

[24] Jean-Yves Le Boudec and Dan-Cristian Tomozei. "A Demand-Response Calculus with Perfect Batteries". In: *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance - 16th International GI/ITG Conference, MMB & DFT 2012, Kaiserslautern, Germany, March 19-21, 2012. Proceedings*. 2012, pp. 273–287. DOI: 10.1007/978-3-642-28540-0_23.

[25] George EP Box, Gwilym M Jenkins, and Gregory C Reinsel. "Time series analysis: forecasting and control". In: vol. 734. ISBN: 978-0-470-27284-8. John Wiley & Sons, 2011. Chap. Forecasting, pp. 137–179.

[26] D.S. Callaway and I.A. Hiskens. "Achieving controllability of electric loads". In: *Proceedings of the IEEE* 99.1 (2011), pp. 184–199.

[27] Francesco Caravelli, Ali Ashtari, Cozmin Ududec, James Requeima, Tiziana Di Matteo, and Tomaso Aste. "PJM and MISO electricity markets price data". In: (May 2015). URL: https://figshare.com/articles/PJM_and_MISO_electricity_markets_price_data/1412608/1.

[28] Nicholas Carr. *Avatars consume as much electricity as Brazilians*. Dec. 2006. URL: http://www.roughtype.com/archives/2006/12/avatars_consume.php.

[29] Che-Wei Chang, Jian-Jia Chen, Waqaas Munawar, Tei-Wei Kuo, and Heiko Falk. "Partitioned Scheduling for Real-time Tasks on Multiprocessor Embedded Systems with Programmable Shared Srams". In: *Proceedings of the Tenth ACM International Conference on Embedded Software*. EMSOFT '12. Tampere, Finland: ACM, 2012, pp. 153–162. ISBN: 978-1-4503-1425-1. DOI: 10.1145/2380356.2380384.

[30] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, Amin Vahdat, and Ronald P. Doyle. "Managing Energy and Server Resources in Hosting Centres". In: *Proceedings of the 18th ACM Symposium on Operating System Principles, SOSP 2001, Chateau Lake Louise, Banff, Alberta, Canada, October 21-24, 2001*. 2001, pp. 103–116. DOI: 10.1145/502034.502045.

[31] Jian-Jia Chen, Heng-Ruey Hsu, and Tei-Wei Kuo. "Leakage-Aware Energy-Efficient Scheduling of Real-Time Tasks in Multiprocessor Systems". In: *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2006), 4-7 April, San Jose, California, USA*. IEEE Computer Society, 2006, pp. 408–417. DOI: 10.1109/RTAS.2006.25.

[32] Jian-Jia Chen, Kai Huang, and Lothar Thiele. "Power management schemes for heterogeneous clusters under quality of service requirements". In: *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC), TaiChung, Taiwan, March 21-24*. 2011, pp. 546–553. DOI: 10.1145/1982185.1982304.

[33] Jian-Jia Chen, Heng-Ruey Hsu, Kai-Hsiang Chuang, Chia-Lin Yang, Ai-Chun Pang, and Tei-Wei Kuo. "Multiprocessor Energy-Efficient Scheduling with Task Migration Considerations". In: *16th Euromicro Conference on Real-Time Systems (ECRTS 2004), 30 June - 2 July, Catania, Italy, Proceedings*. IEEE Computer Society, 2004, pp. 101–108. DOI: 10.1109/ECRTS.2004.20.

[34] City of Burbank. *Adopted Citywide Fee Schedule (Resolution 16-28,846)*. June 2016. URL: https://www.burbankwaterandpower.com/images/FeeSchedules/FY16-17_FeeSchedule_ADOPTED.pdf.

[35] Wesley J Cole, Joshua D Rhodes, William Gorman, Krystian X Perez, Michael E Webber, and Thomas F Edgar. "Community-scale residential air conditioning control for effective grid management". In: *Applied Energy* 130 (2014), pp. 428–436. ISSN: 0306-2619. DOI: http://dx.doi.org/10.1016/j.apenergy.2014.05.067.

[36] *ComEd's Hourly Pricing Program*. URL: https://hourlypricing.comed.com/live-prices/ (visited on 09/05/2016).

[37] Intel Corporation. "Intel 64 and IA-32 Architectures. Software Developer's Manual. Volume 3B: System Programming Guide, Part 2." In: *Intel Manuals* (Sept. 1, 2016).

[38] Paulo Cortez, Miguel Rio, Miguel Rocha, and Pedro Sousa. "Internet Traffic Forecasting using Neural Networks". In: *Proceedings of the International Joint Conference on Neural Networks, IJCNN, part of the IEEE World Congress on Computational Intelligence, WCCI*. (Vancouver, BC, Canada). IEEE, July 21, 2006, pp. 2635–2642. DOI: 10.1109/IJCNN.2006.247142. URL: http://dx.doi.org/10.1109/IJCNN.2006.247142.

[39] *Cost-Competitiveness-Solarbuzz*. 2013. URL: http://solarbuzz.com/facts-and-figures/markets-growth/cost-competitiveness.

[40] Robert I. Davis and Alan Burns. "A survey of hard real-time scheduling for multiprocessor systems". In: *ACM Computing Surveys* 43.4 (2011), p. 35. DOI: 10.1145/1978802.1978814.

[41] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". In: *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*. OSDI'04. San Francisco, CA: USENIX Association, 2004, pp. 10–10. URL: http://dl.acm.org/citation.cfm?id=1251254.1251264.

[42] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

[43] Timothy J Dell, Bruce G Hazelzet, Mark W Kellogg, and Christopher P Miller. *Power management on a memory card having a signal processing element*. US Patent 6,327,664. 2001.

[44] Gary Demasi. *Official Google Blog: Google buys next 20 years worth of wind power*. Apr. 2011. URL: http://googleblog.blogspot.com/2011/04/oklahoma-where-wind-comes-sweepin-down.html.

[45] R.H. Dennard, F.H. Gaensslen, V.L. Rideout, E. Bassous, and A.R. LeBlanc. "Design of ion-implanted MOSFET's with very small physical dimensions". In: *Solid-State Circuits* 9.5 (Oct. 1974), pp. 256–268.

[46] Jason Howard Saurabh Dighe, Sriram R. Vangal, Gregory Ruhl, Nitin Borkar, Shailendra Jain, Vasantha Erraguntla, Michael Konow, Michael Riepen, Matthias Gries, Guido Droege, Tor Lund-Larsen, Sebastian Steibl, Shekhar Borkar, Vivek K. De, and Rob F. Wijngaart. "A 48-Core IA-32 Processor in 45 nm CMOS Using On-Die Message-Passing and DVFS for Performance and Power Scaling". In: *Solid-State Circuits* 46.1 (2011), pp. 173–183. DOI: `10.1109/JSSC.2010.2079450`.

[47] Bruce Dunn, Haresh Kamath, and Jean-Marie Tarascon. "Electrical energy storage for the grid: A battery of choices". In: *Science* 334.6058 (2011), pp. 928–935.

[48] The Economist. "Packing Some Power". In: *The Economist* (Mar. 2012). URL: `http://www.economist.com/node/21548495`.

[49] LLC Energy Tariff Experts. *Can Solar Help Me Avoid Demand Charges?* Accessed: 2016-09-02. June 2014. URL: `http://energytariffexperts.com/blog/2014/6/18/can-solar-help-me-avoid-demand-charges`.

[50] Hadi Esmaeilzadeh, Emily R. Blem, Renée St. Amant, Karthikeyan Sankaralingam, and Doug Burger. "Dark Silicon and the End of Multicore Scaling". In: *IEEE Micro* 32.3 (2012), pp. 122–134. DOI: `10.1109/MM.2012.17`.

[51] Ahmad Faruqui, Ryan Hledik, Sam Newell, and Johannes Pfeifenberger. "The power of five percent: how dynamic pricing can save $35 billion in electricity costs". In: *The Brattle Group, discussion paper* (2007).

[52] Nathan Fisher, Jian-Jia Chen, Shengquan Wang, and Lothar Thiele. "Thermal-Aware Global Real-Time Scheduling on Multicore Systems". In: *15th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2009, San Francisco, CA, USA, 13-16 April*. IEEE Computer Society, 2009, pp. 131–140. DOI: `10.1109/RTAS.2009.34`.

[53] Flett Exchange, LLC. *New Jersy Spot Prices for SREC*. 2019. URL: `www.flettexchange.com` (visited on 01/16/2019).

[54] John Gantz and David Reinsel. "The Digital Universe in 2020: Big Data, Bigger Digital Shadows and Biggest Growth in the Far East". In: *International Data Group -IDC- White Paper* (Dec. 2012). URL: `https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf`.

[55] Qin Gao and Stephan Vogel. "Training Phrase-Based Machine Translation Models on the CloudOpen Source Machine Translation Toolkit Chaski". In: *The Prague Bulletin of Mathematical Linguistics* 93 (2010), pp. 37–46. URL: `http://ufal.mff.cuni.cz/pbml/93/art-gao-vogel.pdf`.

[56] Michael R. Garey and David S. Johnson. "Computers and Intractability: A Guide to the Theory of NP-completeness". In: *Freeman&Co, San Francisco* (1979). ISBN-10: 0716710455.

[57] Golden Valley Electric Association. *GVEA Ni-Cd BESS Project*. URL: http://www.gvea.com/energy/bess (visited on 06/12/2016).

[58] Google. "Google's Green PPAs: What, How, and Why - Rev 02". In: *Google White Papers* (Apr. 2011). URL: http://static.googleusercontent.com/external_content/untrusted_dlcp/www.google.com/en/us/green/pdfs/renewable-energy.pdf.

[59] Sriram Govindan, Anand Sivasubramaniam, and Bhuvan Urgaonkar. "Benefits and limitations of tapping into stored energy for datacenters". In: *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*. IEEE. 2011, pp. 341–351.

[60] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. "Learning influence probabilities in social networks". In: *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*. Ed. by Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu. ACM, 2010, pp. 241–250. DOI: 10.1145/1718487.1718518.

[61] R.L. Graham. "Bounds on multiprocessing timing anomalies". In: *SIAM Journal on Applied Mathematics* 17 (1969), pp. 263–269.

[62] Steve Greenberg, Evan Mills, Bill Tschudi, Peter Rumsey, and Bruce Myatt. "Best practices for data centers: Lessons learned from benchmarking 22 data centers". In: *Proceedings of the ACEEE Summer Study on Energy Efficiency in Buildings in Asilomar, California* 3 (2006), pp. 76–87.

[63] Peter Greenhalgh. "Big. little processing with arm cortex-a15 & cortex-a7". In: *ARM White Paper* (2011).

[64] Raphael Guerra, Julius C. B. Leite, and Gerhard Fohler. "Attaining soft real-time constraint and energy-efficiency in web servers". In: *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), Fortaleza, Ceara, Brazil, March 16-20, 2008*. 2008, pp. 2085–2089. DOI: 10.1145/1363686.1364189.

[65] Christophe Guille and George Gross. "A conceptual framework for the vehicle-to-grid(V2G) implementation". In: *Energy Policy* 37.11 (2009), pp. 4379–4390.

[66] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. "Toward Dark Silicon in Servers". In: *IEEE Micro* 31.4 (2011), pp. 6–15. DOI: 10.1109/MM.2011.77.

[67] Jin Heo, Dan Henriksson, Xue Liu, and Tarek F. Abdelzaher. "Integrating Adaptive Components: An Emerging Challenge in Performance-Adaptive Systems and a Server Farm Case-Study". In: *Proceedings of the 28th IEEE Real-Time Systems Symposium (RTSS 2007), 3-6 December, Tucson, Arizona, USA*. IEEE Computer Society, 2007, pp. 227–238. DOI: `10.1109/RTSS.2007.48`.

[68] Charlotte J Hiatt. *A primer for disaster recovery planning in an IT environment*. ISBN: 1-878289-81-0. IGI Global, 2000.

[69] Stacey Higginbotham. *Whose data centers are more efficient? Facebook's or Google's?* Tech. rep. Gigaom Research. Mar. 2012. URL: `https://gigaom.com/2012/03/26/whose-data-centers-are-more-efficient-facebooks-or-googles`.

[70] Dustin Hillard, Stefan Schroedl, Eren Manavoglu, Hema Raghavan, and Chris Leggetter. "Improving ad relevance in sponsored search". In: *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*. Ed. by Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu. ACM, 2010, pp. 361–370. DOI: `10.1145/1718487.1718532`.

[71] Urs Hölzle. *Official Google Blog: Powering a Google search*. Jan. 2009. URL: `http://googleblog.blogspot.com/2009/01/powering-google-search.html`.

[72] Jason Howard, Saurabh Dighe, Sriram R. Vangal, Gregory Ruhl, Nitin Borkar, Shailendra Jain, Vasantha Erraguntla, Michael Konow, Michael Riepen, Matthias Gries, Guido Droege, Tor Lund-Larsen, Sebastian Steibl, Shekhar Borkar, Vivek K. De, and Rob F. Wijngaart. "A 48-Core IA-32 Processor in 45 nm CMOS Using On-Die Message-Passing and DVFS for Performance and Power Scaling". In: *J. Solid-State Circuits* 46.1 (2011), pp. 173–183. DOI: `10.1109/JSSC.2010.2079450`.

[73] Hussein Ibrahim, Adrian Ilinca, and Jean Perron. "Energy storage systems—characteristics and comparisons". In: *Renewable and Sustainable Energy Reviews* 12.5 (2008), pp. 1221–1250.

[74] Uptime Institute. *2014 Data Center Industry Survey*. Nov. 2014. URL: `https://journal.uptimeinstitute.com/2014-data-center-industry-survey/`.

[75] Intel Corporation. *Desktop 3rd Generation Intel Core Processor Family - Thermal Mechanical Specifications and Design Guidelines*. Jan. 2013.

[76] International Energy Agency (IEA). *CO_2 Emissions from Fuel Combustion 2018 (with 2016 data)*. Tech. rep. ISBN: 978-92-64-27819-6. Oct. 2018. URL: `https://webstore.iea.org/co2-emissions-from-fuel-combustion-2018`.

[77] Ravindra Jejurikar, Cristiano Pereira, and Rajesh K. Gupta. "Leakage aware dynamic voltage scaling for real-time embedded systems". In: *Proceedings of the 41th Design Automation Conference, DAC'04, San Diego, CA, USA, June 7-11*. Ed. by Sharad Malik, Limor Fix, and Andrew B. Kahng. ACM, 2004, pp. 275–280. DOI: 10.1145/996566.996650.

[78] James M Kaplan, William Forrest, and Noah Kindler. *Revolutionizing data center energy efficiency*. Tech. rep. McKinsey & Company, July 2008. URL: http://www.sallan.org/pdf-docs/McKinsey_Data_Center_Efficiency.pdf.

[79] Joe Kava. *Measuring to improve: comprehensive, real-world data center efficiency numbers*. Google Official Blog. Mar. 2012. URL: https://googleblog.blogspot.com/2012/03/measuring-to-improve-comprehensive-real.html.

[80] Willett Kempton and Jasna Tomić. "Vehicle-to-grid power fundamentals: calculating capacity and net revenue". In: *Journal of Power Sources* 144.1 (2005), pp. 268–279.

[81] Sharath Kodase, Shige Wang, Zonghua Gu, and Kang G. Shin. "Improving Scalability of Task Allocation and Scheduling in Large Distributed Real-Time Systems Using Shared Buffers". In: *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2003), May 27-30, Toronto, Canada*. IEEE Computer Society, 2003, pp. 181–188. DOI: 10.1109/RTTAS.2003.1203050.

[82] Paul Kohlenbach and Uli Jakob. *Solar Cooling: The Earthscan Expert Guide to Solar Cooling Systems*. Earthscan Expert. ISBN: 978-0415639750. Routledge, Aug. 2014.

[83] Jonathan Koomey. "Growth in data center electricity use 2005 to 2010". In: *A report by Analytical Press, completed at the request of The New York Times* (2011), p. 9.

[84] Chuck Lam. *Hadoop in Action*. 1st. Greenwich, CT, USA: Manning Publications Co., 2010. ISBN: 1935182196, 9781935182191.

[85] Kien Le, Ricardo Bianchini, Thu D. Nguyen, Ozlem Bilgir, and Margaret Martonosi. "Capping the brown energy consumption of Internet services at low cost". In: *International Green Computing Conference, Chicago, IL, USA, 15-18 August*. 2010, pp. 3–14. DOI: 10.1109/GREENCOMP.2010.5598305.

[86] Kien Le, Ricardo Bianchini, Margaret Martonosi, and T Nguyen. "Cost- and energy-aware load distribution across data centers". In: *SOSP Workshop on Power Aware Computing and Systems (HotPower)*. Oct. 2009, pp. 1–5.

[87] Kien Le, Ozlem Bilgir, Ricardo Bianchini, Margaret Martonosi, and Thu D. Nguyen. "Managing the cost, energy consumption, and carbon footprint of internet services". In: *SIGMETRICS 2010, Proceedings of the 2010 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, New York, USA, 14-18 June 2010*. 2010, pp. 357–358. DOI: 10.1145/1811039.1811085.

[88] J-Y Le Boudec and D-C Tomozei. "Demand response using service curves". In: *ISGT Europe*. IEEE. 2011, pp. 1–8.

[89] Jean-Yves Le Boudec and Patrick Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*. Springer, 2001.

[90] C. Le Quéré, R. M. Andrew, P. Friedlingstein, S. Sitch, J. Hauck, et al. "Global Carbon Budget 2018". In: *Earth System Science Data* 10.4 (2018), pp. 2141–2194. DOI: 10.5194/essd-10-2141-2018. URL: https://www.earth-syst-sci-data.net/10/2141/2018/.

[91] J. Lee, B. Yun, and K. Shin. "Reducing Peak Power Consumption in Multi-Core Systems Without Violating Real-Time Constraints". In: *Parallel & Distributed Sys. '13.* (). ISSN: 1045-9219. DOI: 10.1109/TPDS.2013.131.

[92] Jie Li et al. "Towards Optimal Electric Demand Management for Internet Data Centers". In: *IEEE Transactions on Smart Grid* (2012), pp. 183–192. ISSN: 1949-3053. DOI: 10.1109/TSG.2011.2165567.

[93] Jing Li, Jian Jia Chen, Kunal Agrawal, Chenyang Lu, Chris Gill, and Abusayeed Saifullah. "Analysis of federated and global scheduling for parallel real-time tasks". In: *2014 26th Euromicro Conference on Real-Time Systems*. IEEE. 2014, pp. 85–96.

[94] Na Li, Lijun Chen, and Steven H Low. "Optimal demand response based on utility maximization in power networks". In: *Power and Energy Society General Meeting*. IEEE. 2011, pp. 1–8.

[95] Shen Li, Shaohan Hu, and Tarek F. Abdelzaher. "The Packing Server for real-time scheduling of MapReduce workflows". In: *21st IEEE Real-Time and Embedded Technology and Applications Symposium, Seattle, WA, USA, April 13-16, 2015*. IEEE Computer Society, 2015, pp. 51–62. DOI: 10.1109/RTAS.2015.7108416.

[96] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. "The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing". In: *TACO* 10.1 (2013), p. 5. DOI: 10.1145/2445572.2445577.

[97] Jia-Chun Lin, Fang-Yie Leu, Ying-Ping Chen, and Waqaas Munawar. "Impact of MapReduce Task Re-execution Policy on Job Completion Reliability and Job Completion Time". In: *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*. May 2014, pp. 712–718. DOI: `10.1109/AINA.2014.87`.

[98] Xiao Ling, Yi Yuan, Dan Wang, Jiangchuan Liu, and Jiahai Yang. "Joint scheduling of MapReduce jobs with servers: Performance bounds and experiments". In: *Journal of Parallel and Distributed Computing* 90-91 (Apr. 2016), pp. 52–66. DOI: `10.1016/j.jpdc.2016.02.002`.

[99] Yang Liu, Xiaohong Jiang, Huajun Chen, Jun Ma, and Xiangyu Zhang. "MapReduce-Based Pattern Finding Algorithm Applied in Motif Detection for Prescription Compatibility Network". In: *Advanced Parallel Processing Technologies, 8th International Symposium, APPT 2009, Rapperswil, Switzerland, August 24-25, 2009, Proceedings*. Ed. by Yong Dou, Ralf Gruber, and Josef M. Joller. Vol. 5737. Lecture Notes in Computer Science. Springer, 2009, pp. 341–355. DOI: `10.1007/978-3-642-03644-6_27`.

[100] Rudolf Lohner. "ForHLR II - energy - and resource efficient computing at KIT". In: *4th Workshop Energy for Sustainable Science at Research Infrastructures ESS*. (Bucharest, Romania). Nov. 23, 2017. URL: `https://indico.eli-np.ro/event/1/contributions/61/attachments/83/121/6_2017-11-23_ESS-WS_ForHLR-KIT.pdf`.

[101] Steve Lohr. *Google Schools Its Algorithm - NYTimes.com*. Mar. 2011. URL: `http://www.nytimes.com/2011/03/06/weekinreview/06lohr.html?pagewanted=1\&\_r=1\&hpw`.

[102] Jianying Luo, Lei Rao, and Xue Liu. "Data center energy cost minimization: a spatio-temporal scheduling approach". In: *INFOCOM, 2013 Proceedings IEEE*. IEEE. 2013, pp. 340–344.

[103] V. Mathew et al. "Energy-aware load balancing in content delivery networks". In: *INFOCOM*. Mar. 2012, pp. 954–962.

[104] Suzanne Matthews and Tiffani L. Williams. "MrsRF: an efficient MapReduce algorithm for analyzing large collections of evolutionary trees". In: *BMC Bioinformatics* 11.S-1 (2010), p. 15. DOI: `10.1186/1471-2105-11-S1-S15`.

[105] Angus McCrone, Ulf Moslener, Eric Usher, Christine Grüning, and Virginia Sonntag-O'Brien. *Global Trends in Renewable Energy Investment 2015*. Tech. rep. Frankfurt School –UNEP Collaborating Centre for Climate & Sustainable Energy Finance and Bloomberg New Energy Finance, Mar. 2015. URL: `http://fs-unep-centre.org/publications/global-trends-renewable-energy-investment-2015`.

[106] E. J. McLellan and D. A. Webb. "The Alpha 21264 Microprocessor Architecture". In: *ICCD '98*. ISBN: 0-8186-9099-2.

[107] Robert McNaughton. "Scheduling with deadlines and loss functions". In: *Management Science* 6.1 (1959), pp. 1–12.

[108] Shengwei Mei, Xuemin Zhang, and Ming Cao. *Power grid complexity*. ISBN: 978-3-642-16210-7. Springer Science & Business Media, 2011.

[109] Aditya Mishra, David Irwin, Prashant Shenoy, and Ting Zhu. "Scaling distributed energy storage for grid peak reduction". In: *4th international conference on Future energy systems*. ACM. 2013, pp. 3–14.

[110] Waqaas Munawar and Jian-Jia Chen. "Peak power demand analysis and reduction by using battery buffers for monotonic controllers". In: *2013 23rd International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), Karlsruhe, Germany, September 9-11, 2013*. IEEE, 2013, pp. 255–258. DOI: 10.1109/PATMOS.2013.6662185.

[111] Waqaas Munawar, Jian-Jia Chen, and Minming Li. "Minimizing Environmental Footprints of Data Centers under Budget and Service Requirement Constraints". In: *SMARTGREENS 2014 - Proceedings of the 3rd International Conference on Smart Grids and Green IT Systems, Barcelona, Spain, 3-4 April*. 2014, pp. 222–232. DOI: 10.5220/0004934202220232.

[112] Waqaas Munawar, Heba Khdr, Santiago Pagani, Muhammad Shafique, Jian-Jia Chen, and Jörg Henkel. "Peak Power Management for scheduling real-time tasks on heterogeneous many-core systems". In: *20th IEEE International Conference on Parallel and Distributed Systems, ICPADS, Hsinchu, Taiwan, December 16-19*. IEEE Computer Society, 2014, pp. 200–209. DOI: 10.1109/PADSW.2014.7097809.

[113] Prashanth Mundkur, Ville Tuulos, and Jared Flatow. "Disco: A Computing Platform for Large-scale Data Analytics". In: *Proceedings of the 10th ACM SIGPLAN Workshop on Erlang*. Erlang '11. Tokyo, Japan: ACM, 2011, pp. 84–89. ISBN: 978-1-4503-0859-5. DOI: 10.1145/2034654.2034670.

[114] Thannirmalai Somu Muthukaruppan, Mihai Pricopi, Vanchinathan Venkataramani, Tulika Mitra, and Sanjay Vishin. "Hierarchical power management for asymmetric multi-core in dark silicon era". In: *The 50th Annual Design Automation Conference, DAC '13, Austin, TX, USA, May 29 - June 07*. ACM, 2013, 174:1–174:9. DOI: 10.1145/2463209.2488949.

[115] Dean Nelson, Michael Ryan, Serena DeVito, KV Ramesh, Petr Vlasaty, Brett Rucker, B. Da y Nelson, et al. "The role of modularity in datacenter design". In: *Sun BluePrints Online* (May 2010). White paper. URL: `http://www.oracle.com/technetwork/articles/systems-hardware-architecture/moddatactrdesign-163921.pdf`.

[116] *North Carolina Renewable Energy Tracking System (NC-RETS)*. 2015. URL: `http://www.ncrets.org/` (visited on 02/13/2015).

[117] NYISO. *New York Independent System Operator*. 2015. URL: `http://www.nyiso.com/`.

[118] Open Access Same-time Information System (OASIS). *California Independent System Operator (CAISO)*. 2015. URL: `http://oasis.caiso.com` (visited on 05/03/2018).

[119] Alexandre Oudalov, Rachid Cherkaoui, and Antoine Beguin. "Sizing and Optimal Operation of Battery Energy Storage System for Peak Shaving Application". In: *Lausanne Power Tech*. IEEE. July 2007, pp. 621–625. DOI: `10.1109/PCT.2007.4538388`.

[120] Santiago Pagani, Heba Khdr, Waqaas Munawar, Jian-Jia Chen, Muhammad Shafique, Minming Li, and Jörg Henkel. "TSP: Thermal Safe Power - Efficient power budgeting for many-core systems in dark silicon". In: *International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS*. (Uttar Pradesh, India). Ed. by Radu Marculescu and Gabriela Nicolescu. ACM, Oct. 12, 2014, pp. 1–10. DOI: `10.1145/2656075.2656103`.

[121] Peter Palensky and Dietmar Dietrich. "Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads". In: *IEEE Transactions on Industrial Informatics* 7.3 (2011), pp. 381–388. DOI: `10.1109/TII.2011.2158841`.

[122] Yiqun Pan, Rongxin Yin, and Zhizhong Huang. "Energy modeling of two office buildings with data center for green building design". In: *Energy and Buildings* 40.7 (2008), pp. 1145–1152.

[123] Linh TX Phan, Zhuoyao Zhang, Boon Thau Loo, and Insup Lee. *Realtime MapReduce scheduling*. Tech. rep. MS-CIS-10-32. Department of Computer & Information Science, University of Pennsylvania, 2010.

[124] Marco Piemontesi and Cord Dustmann. "Energy storage systems for UPS and energy managment at consumer level". In: *International Stationary Battery Conference*. BATTCON '10. Hollywood, FL, USA.

[125] *Power Smart Pricing - Realtime pricing tariff for residential customers*. URL: `https://www.powersmartpricing.org/prices/` (visited on 09/05/2016).

[126] Asfandyar Qureshi. "Plugging Into Energy Market Diversity". In: *7th ACM Workshop on Hot Topics in Networks - HotNets-VII, Calgary, Alberta, Canada, October 6-7*. 2008, pp. 49–54. URL: `http://conferences.sigcomm.org/hotnets/2008/papers/9.pdf`.

[127] Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John V. Guttag, and Bruce M. Maggs. "Cutting the electric bill for internet-scale systems". In: *Proceedings of the ACM SIGCOMM 2009 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Barcelona, Spain, August 16-21*. 2009, pp. 123–134. DOI: `10.1145/1592568.1592584`.

[128] Bharathwaj Raghunathan and Siddharth Garg. "Job arrival rate aware scheduling for asymmetric multi-core servers in the dark silicon era". In: *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*. ACM. 2014, p. 14.

[129] Bharathwaj Raghunathan, Yatish Turakhia, Siddharth Garg, and Diana Marculescu. "Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors". In: *DATE '13*. Grenoble, France. ISBN: 978-1-4503-2153-2.

[130] Lei Rao, Xue Liu, Le Xie, and Wenyu Liu. "Minimizing Electricity Cost: Optimization of Distributed Internet Data Centers in a Multi-Electricity-Market Environment". In: *INFOCOM'10 29th IEEE International Conference on Computer Communications*. (San Diego, CA, USA). IEEE, Mar. 15, 2010, pp. 1145–1153. DOI: `10.1109/INFCOM.2010.5461933`.

[131] Shafiqur Rehman, Luai M Al-Hadhrami, and Md Mahbub Alam. "Pumped hydro energy storage system: a technological review". In: *Renewable and Sustainable Energy Reviews* 44 (2015), pp. 586–598.

[132] Lazar Rozenblat. *Renewable Energy Sources: Cost Comparison*. July 2016. URL: `http://www.renewable-energysources.com` (visited on 06/05/2018).

[133] G Sudha Sadasivam and G Baktavatchalam. "A novel approach to multiple sequence alignment using hadoop data grids". In: *MDAC '10: Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud*. Raleigh, North Carolina, USA: ACM, 2010. ISBN: 978-1-60558-991-6.

[134] San Diego Gas and Electric (SDG&E). *Energy Tariff–San Diego Gas and Electric*. Aug. 2016. URL: `http://www2.sdge.com/tariff/com-elec/ALTOUPrimary.pdf`.

[135] John Sartori and Rakesh Kumar. "Distributed peak power management for many-core architectures". In: *Design, Automation and Test in Europe, DATE 2009, Nice, France, April 20-24*. Ed. by Luca Benini, Giovanni De Micheli, Bashir M. Al-Hashimi, and Wolfgang Müller. IEEE, 2009, pp. 1556–1559. DOI: 10.1109/DATE.2009.5090910.

[136] Phillip F Schewe. *The Grid:: A Journey Through the Heart of Our Electrified World*. ISBN 978-0-309-10260-5. National Academies Press, 2007.

[137] Oliver Schmidt, Adam Hawkes, Ajay Gambhir, and Iain Staffell. "The future cost of electrical energy storage based on experience rates". In: *Nature Energy* 2.8 (2017), p. 17110.

[138] Amip J Shah and Nikhil Krishnan. "Optimization of global data center thermal management workload for minimal environmental and economic burden". In: *Components and Packaging Technologies, IEEE Transactions on* 31.1 (2008), pp. 39–45.

[139] Souther California Edison. *Price Schedule GS-2*. Jan. 2014. URL: https://www.sce.com/NR/sc3/tm2/pdf/ce30-12.pdf.

[140] FERC Staff. *Energy Primer - A Handbook of Energy Market Basics*. Tech. rep. pp. 35–102. Federal Energy Regulatory Commission (FERC), Nov. 2015.

[141] Nicholas Stern, Alex Bowen, and John Whalley. *The Global Development of Policy Regimes to Combat Climate Change*. Vol. 4. World Scientific Publishing Co. Pte. Ltd., 2014.

[142] Christopher Stewart and Kai Shen. "Some joules are more precious than others: Managing renewable energy in the datacenter". In: *Proceedings of the Workshop on Power Aware Computing and Systems (HotPower)*. 2009.

[143] Goran Strbac. "Demand side management: Benefits and challenges". In: *Energy Policy* 36.12 (2008). Foresight Sustainable Energy Management and the Built Environment Project, pp. 4419–4426. DOI: 10.1016/j.enpol.2008.09.030.

[144] Michael B. Taylor. "Is dark silicon useful?: harnessing the four horsemen of the coming dark silicon apocalypse". In: *DAC*. San Francisco, California, 2012. ISBN: 978-1-4503-1199-1.

[145] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. "Hive - A Warehousing Solution Over a Map-Reduce Framework". In: *PVLDB* 2.2 (2009), pp. 1626–1629. URL: http://www.vldb.org/pvldb/2/vldb09-938.pdf.

[146] W Pitt Turner IV, John H Seader, and Vincent E Renaud. "Data center site infrastructure tier standard: Topology". In: *Uptime Institute* (2010).

[147] UNFCCC. Conference of the Parties (COP). *Adoption of the Paris Agreement. Proposal by the President.* Dec. 12, 2015. URL: `http://unfccc.int/resource/docs/2015/cop21/eng/l09.pdf`.

[148] United Nations Framework Convention on Climate Change (UNFCCC). "Protocol, Koyoto". In: *Kyoto Protocol, Kyoto* (1997).

[149] Sandra Upson. "The greening of Google". In: *Spectrum, IEEE* 44.10 (2007), pp. 24–28.

[150] Guido Urdaneta et al. "Wikipedia Workload Analysis for Decentralized Hosting". In: *Elsevier Computer Networks* (July 2009).

[151] Rahul Urgaonkar, Bhuvan Urgaonkar, Michael J. Neely, and Anand Sivasubramaniam. "Optimal power cost management using stored energy in data centers". In: SIGMETRICS '11. San Jose, California, USA: ACM, pp. 221–232. ISBN: 978-1-4503-0814-4. DOI: `10.1145/1993744.1993766`. URL: `http://doi.acm.org/10.1145/1993744.1993766`.

[152] U.S. Energy Information Administration. *Electric power monthly*. Tech. rep. Washington, DC, USA, Aug. 2016. URL: `http://www.eia.gov/electricity/monthly/pdf/epm.pdf`.

[153] U.S. Energy Information Administration, (EIA). *Levelized Cost and Levelized Avoided Cost of New Generation Resources in the Annual Energy Outlook 2016*. Aug. 2016. URL: `http://www.eia.gov/forecasts/aeo/pdf/electricity_generation.pdf`.

[154] U.S. Office of Enforcement. *OE ENERGY MARKET SNAPSHOT - Northeast States Version*. Tech. rep. Federal Energy Regulatory Commission, May 2010.

[155] Ward Van Heddeghem, Sofie Lambert, Bart Lannoo, Didier Colle, Mario Pickavet, and Piet Demeester. "Trends in worldwide ICT electricity consumption from 2007 to 2012". In: *Computer Communications* 50 (2014), pp. 64–76.

[156] Akshat Verma, Pradipta De, Vijay Mann, Tapan Kumar Nayak, Amit Purohit, Gargi Dasgupta, and Ravi Kothari. "BrownMap: Enforcing Power Budget in Shared Data Centers". In: *Middleware 2010 - ACM/IFIP/USENIX Proceedings of 11th International Middleware Conference*. (Bangalore, India). Ed. by Indranil Gupta and Cecilia Mascolo. Vol. 6452. Lecture Notes in Computer Science. Springer, Nov. 29, 2010, pp. 42–63. DOI: `10.1007/978-3-642-16955-7_3`.

[157] Guozhang Wang, Marcos Vaz Salles, Benjamin Sowell, Xun Wang, Tuan Cao, Alan Demers, Johannes Gehrke, and Walker White. "Behavioral Simulations in MapReduce". In: *Proceedings of VLDB Endow.* 3.1-2 (Sept. 2010), pp. 952–963. ISSN: 2150-8097. DOI: `10.14778/1920841.1920962`.

[158] Shengquan Wang, Jian-Jia Chen, Zhenjun Shi, and Lothar Thiele. "Energy-Efficient Speed Scheduling for Real-Time Tasks under Thermal Constraints". In: *15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA, Beijing, China, 24-26 August*. 2009, pp. 201–209. DOI: 10.1109/RTCSA.2009.29.

[159] Shengquan Wang, Waqaas Munawar, Jun Liu, Jian-Jia Chen, and Xue Liu. "Power-Saving Design for Server Farms with Response Time Percentile Guarantees". In: *2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium, Beijing, China, April 16-19*. 2012, pp. 273–284. DOI: 10.1109/RTAS.2012.35.

[160] Chun You and Kavitha Chandra. "Time series models for internet data traffic". In: *Local Computer Networks (LCN)*. IEEE. Oct. 18, 1999. DOI: 10.1109/LCN.1999.802013.

[161] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Spark: Cluster Computing with Working Sets". In: *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA*. Ed. by Erich M. Nahum and Dongyan Xu. USENIX Association, June 2010. URL: https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets.

[162] Xiyue Zhang and K. Max Zhang. "Demand Response, Behind-the-Meter Generation and Air Quality". In: *Environmental Science & Technology* 49.3 (Feb. 2015), pp. 1260–1267. DOI: 10.1021/es505007m.

[163] Yanwei Zhang, Yefu Wang, and Xiaorui Wang. "GreenWare: Greening Cloud-Scale Data Centers to Maximize the Use of Renewable Energy". In: *Middleware 2011 - ACM/IFIP/USENIX Proceedings of 12th International Middleware Conference*. (Lisbon, Portugal). Dec. 12, 2011, pp. 143–164. DOI: 10.1007/978-3-642-25821-3_8.

[164] Weibin Zhao, David Olshefski, and Henning G Schulzrinne. *Internet quality of service: An overview*. Tech. rep. CUCS-003-00. Department of Computer Science, Columbia University, 2000.

[165] Weizhong Zhao, Huifang Ma, and Qing He. "Parallel K-Means Clustering Based on MapReduce". In: *Proceedings of First International Conference on Cloud Computing, (CloudCom 2009)*. (Beijing, China). Ed. by Martin Gilje Jaatun, Gansen Zhao, and Chunming Rong. Vol. 5931. Lecture Notes in Computer Science. Springer, Dec. 1, 2009, pp. 674–679. DOI: 10.1007/978-3-642-10665-1_71.

[166] D. Zhu, S. Yue, N. Chang, and M. PedraG. "Toward a Profitable Grid-Connected Hybrid Electrical Energy Storage System for Residential Use". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.7 (July 2016), pp. 1151–1164. ISSN: 0278-0070. DOI: 10.1109/TCAD.2015.2501296.