# Cost and Energy Efficient Job Scheduling Algorithm for Cloud Data Centre

Masters Thesis

Harshit Kapoor
Student No. 666810
Supervisor: Dr. Adel Nadjaran Toosi

Master of Science (Computer Science)
Research Project (75 Points)
COMP60002, COMP60003 & COMP60004
June 2017

Department of Computing and Information Systems
THE UNIVERSITY OF MELBOURNE

## Abstract:

As a result of an increased interest in the storage and processing of large amounts of data or "Big Data", the IT industry has entered a data-intensive age. A single computer cannot process this much data. Therefore, we need data centers with multiple physical or virtual machines to satisfy this need of the market. While these data centers provide us the cloud computing functionalities, they also require a significant amount of brown (fossil-based) energy to function. Consequently, this results in an increase in electricity costs as well as damages to the environment. Extensive research is being performed to replace brown energy with green (renewable) energy sources such as wind, hydropower, solar etc. to power up these data centers.

Today, many companies are moving to adopt green energy as their primary source to power up their cloud data centers. Therefore, it has become essential to develop a mechanism to use cloud data centers in environmentally friendly manner and increase the usage of green energy.

In this thesis, we propose a job- scheduling algorithm for a cloud data center that is cost and energy efficient. The main problem that we are trying to solve with this algorithm is matching energy required by the workload with the available green energy in a cloud data center. We also use this algorithm with real-time workload and present the obtained results. We try many different scheduling techniques to find the most optimal technique to maximize usage of green energy and minimize overall cost. The results of our experiments show that using least running time first (LRTF) and least nodes first scheduling policy we can maximize the usage of green energy. Whereas, using first come first serve (FCFS) scheduling policy, we can minimize the time required to execute the workload in a data center.

*I certify that*

- *This thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.*

- *Where necessary I have received clearance for this research from the University's Ethics Committee and have submitted all required data to the Department.*

- *The thesis is about 17093 words in length.*

## Acknowledgement

I would like to thank my thesis supervisor, Dr. Adel Nadjaran Toosi, for his endless guidance and patience throughout the span of my research project.

I would also like to express my profound gratitude to my guru Pt. Gopal Dutt; my father, Kapil Kapoor; my mother, Rashmi Kapoor; my brother, Yogesh Kapoor and all of my friends who supported me at every step in these last two years.

## Table of Contents

## List of figures

## List of tables

## 1. Introduction

In 2007, as per the reports released by Google [1], their cloud data centers consumed about 0.01 percent of the global electricity [2]. The same year, according to the U.S. Energy Information Administration, the global electricity consumption was estimated to be around 19,710 billion kilowatt-hours. Using the estimates provided by Google, their data centers actually consumed the same amount of electrical energy as that of entire Turkey's. The alarming fact about this data was that all of this energy was actually from coal, which is brown energy.

This data was presented in campaigns by the [3] GreenPeace organization, in order to prevent environmental degradation around the world. This degradation was due to significant amounts of brown energy being used to power up these server factories i.e. data centers. The campaigns led Facebook to entirely switch to use of renewable forms of energy as their electricity source. Google followed the same fashion after extensive research on use and implementation of green algorithms. However, the hidden problem does not lie with these monster Internet companies, but small and medium-sized enterprises, which employ small data centers spanning from a few tens of servers to a few hundreds of servers.
Unfortunately, the inefficiencies of these small and medium-sized data centers impact the environment in a very harmful way. Mostly, these data centers consume brown energy sourced electricity i.e. the electricity produced using carbon intensive methods, for example, burning of coal and hence supplied through electricity grid. As per the executive summary presented by one of the environment protection groups in the UK, a single server has the same carbon footprint as that of a SUV vehicle [4]. The increase in carbon footprint results in global warming all around the globe consequently increases Earth's temperature. We cannot avoid the fact that, today every market on our planet has become data intensive. Therefore, these data centers and cloud computing have become essential. Therefore, predicting that there will be a huge increase in demand and establishment of small/medium data centers should not be incorrect.
However, this concern has come to light in the past few years. Many private and government led initiatives have been setup to promote the use of renewable energy or green energy over the use of brown energy. Many countries have come up with solutions and initiatives [5] [6], which involve incentives from government to provide or use green energy. Data centers also have started participating in this emerging change of promoting the use of green energy. [7, 8] Some of the data centers are generating their own electricity using either solar or wind energy or a combination of both, to power up the data centers.

Many scholars and researchers have researched and are researching on this topic of efficiently using green energy in data centers to reduce the use of brown energy and hence, carbon footprint. Some of these works have been discussed in the literature survey section. Most of these studies deal with using only the renewable form of energy to power up the data centers and are not flexible to accommodate brown energy as well. Some of these researches deal with load

balancing and leverage the geographical locations of data centers to schedule the execution of workload in such a way that the cheapest electricity cost is incurred. To achieve this, some have used migration of data as well. These algorithms are called green algorithms, which are nothing but recipes to decrease the consumption of coal energy and replacing it with renewable sources of energy.
However, in our thesis, the argument that we have presented is that we can formulate an algorithm flexible enough, which uses both green energy and brown energy. The main aim of the algorithm proposed in this paper is to increase the use of green energy to its maximum potential. In green energy deficient cases and to prevent job execution from violating its deadline, we do not restrict our strategy from using brown energy. All of this has to be done with the minimum cost possible. We try to minimize brown energy as much as possible by matching the energy required by the workload with the available green energy.
To make our algorithm efficient, we have focused on different scheduling policies in this thesis. These scheduling policies are the core machinery of our algorithm and are responsible mainly for scheduling of the jobs before the jobs are sent for allotment of time slots in which they will execute, as they arrive in the system. We have analyzed the overall algorithm with four different scheduling policies, on the basis of various factors such as maximum green energy used, minimum brown energy used and minimum cost incurred.

The remainder of the thesis is as follows: the next section provides the literature survey, discussing all the works which inspired this thesis. Next section includes our motivation to engage and perform this research and background to this thesis including the description of various terms and concepts used. After which, we discuss the problem definition along with the thesis statement. The next section has the proposed scheduling algorithm with its detailed explanation, which is followed by results and analysis of the results. Finally, we conclude the thesis with a summary and a conclusion, which includes the steps, which could be followed in future to improve the proposed algorithm.

## 2. Literature Survey

**Leveraging electricity prices to control overall cost.** In [9], the author has proposed the use of geographical load balancing with multiple data centers for cost savings. They talk about how the current workload can be segregated and migrated between multiple data centers and what part of it can be executed and delayed, without violating the deadline. In order to save cost, they mainly focus on dynamic electricity price changes in different regions. They use prediction model for future electricity prices to reduce the overall cost. Their main focus is on delaying the workload in a manner such that the cost of executing the workload after delaying it for a given time interval without violating its deadline is lesser than the cost associated with executing the workload at the current time. In order to calculate the cost associated with a workload, they mainly focus on electricity prices and use a future electricity price prediction model to calculate the cost incurred by the delayed execution of the workload.
However, our main aim is to increase the usage of renewable energy and decrease the use of brown energy. We are talking about a single datacenter. We use prediction model, which is mostly based on future weather predictions. As per the availability of green energy we formulate a workflow, which consists of multiple jobs scheduled in such a way that they maximize the use green energy. While doing this, our algorithm uses electricity prices to calculate the overall cost of the workflow and minimizes the cost as well (We assume that green energy is much cheaper than the brown energy). The factors, which affect a workflow in our algorithm, are:

- Green energy prediction
- Day/Night electricity prices
- Number of processors (cores) available in the data center, which can be engaged to execute the jobs.
- Deadline of the jobs.

We use a penalty model as well, as in if a job exceeds its deadline in a given workflow, then we apply a penalty on that workflow in terms of cost and consequently, our algorithm returns us the workflow with the minimum cost of executing the current jobs.

**Utilization of green energy & saving energy in datacenters.** In [10], The main focus in this paper is to increase the utilization of energy by reducing its wastage, which occurs due to varying workload. The algorithm tries to reduce the overall cost of the energy required to execute the workload. In order to achieve the aim, the author takes the same approach of delaying the workload but here in order to save energy, capacity provisioning is done. The suggested approach determines how many servers are to be kept active to execute the workload along with the determination of how much workload can be delayed, within the bounds of deadline suggested by Service Level Agreement (SLA), such that the energy utilization is most optimized. The workload consists of mainly interactive and batch jobs. The costs considered in this approach are operational and switching costs. Operational is mainly related to executing the workload where as switching cost is related to switching the server state between on and off. The main aim is to determine the number of active servers in a particular data center

and hence forth, dispatching partial workload to that data center. This is capacity provisioning.
However, we consider only batch jobs. Our cost model considers the electricity prices and the penalty applied to the workflow. The main assumption we take is that price for green energy is 0, and therefore the workflow with maximum green energy will presumably have the lowest cost. Hence, the cheapest workflow schedule will be given as the output. In our algorithm, to save up brown energy and maximize the utilization of green energy, the algorithm tries to allocate the maximum number of jobs in those time slots where we have maximum green energy availability along with the minimum cost incurred.

In [7], the solution to the problem of energy consumption in a virtualized datacenter has been presented. To achieve this aim, the suggested approach is to turn off all the idle machines whenever they are not in use or is not required to run the required VM's at hand. It mainly describes a scheduling policy based on a score for each host, which has the capability of running a virtual machine (VM). The score represents the VM hosting capability of a host, which is based on factors such as costs involved due to virtualization, reliability, dynamic SLA enforcement and power consumption. This paper mainly talks about the virtualized data center and redistribution of VM's spawned in order to accommodate the workload. However, in our approach, we mainly discuss how the workload (jobs) can be executed maximizing the use of green energy and minimizing the cost involved. As per this approach, whenever the state of the whole virtualized data center changes (for example when a VM is spawned), the redistribution of VMs takes place and the workload is consolidated to a lesser number of machines and spare servers/host/machines are henceforth turned off.
In our algorithm we have used various scheduling policies such as Least slack time first (LSTF), First come first serve (FCFS), Least running time first and least cores first. We compare the working of our overall algorithm with all these policies to analyze the consumption of green energy and cost involved. Our concern is with workload consisting of jobs rather than spawning of VM in a virtualized environment. We don't use score based mechanism. Rather, we use cost-based model to give the cheapest workflow to execute the jobs. This required an assumption that green energy is considered to be free and brown energy has different on peak and off peak prices.

Prior research shows that a lot of work has been done in this direction of energy efficiency. Work done in [11] promotes the utilization of renewable energy to make green data centers. The work performed in this paper mainly focuses on dispatching of request amongst multiple data centers in order to maximize the use of renewable energy, without violating the budget constraints. Our work mainly focuses on a single data center and how to create a workflow that is the order in which jobs should be executed, at a single data center in order to maximize the renewable energy used with minimalistic cost. This paper talks about a middleware, which dispatches requests to different data centers at different geographical locations. However, since we are talking about only one data center, therefore our main focus is to delay/ re-arrange the jobs in such a way that we maximize green energy. The research that we have conducted also

includes comparison and analysis of different scheduling policies with our algorithm to determine which of the policies maximize the utilization of green energy, minimize the cost, execute the maximum number of jobs and which waste the most amount of green energy. [12] Strengthens the idea of maximizing the use of renewable energy using the green energy predictions. They even consider the use of varying electricity prices. But the difference in our research and the research presented in this paper is that we deal with batch jobs. We have chosen batch jobs rather than interactive small tasks/requests because usually batch jobs run for longer periods of time. Since these jobs are bags of tasks and run for longer periods of time, their deadlines tend to be not that strict and hence we can schedule these jobs more efficiently in terms of maximizing the utilization of green energy and minimizing the cost involved. Their focus is on multiple data centers whereas we deal with a single datacenter and try to produce the most optimized workflow for that data center.
In [13], an infrastructure for the simulation to evaluate the cost of energy for a data center powered by green energy sources, called ReRack has been proposed. The proposed infrastructure mainly focuses on analyzing and modeling of a consumed power by a data center and the available green energy dependent on the location of the data center. It also finally optimizes the available green energy power as per the workload available. The main task of this proposed simulation model is to determine the cost associated if a given workload and with a given energy availability, will execute at a given data center. It also optimizes the use of different forms of renewable energy sources in order to minimize the overall cost. However, our research is more about scheduling the jobs in such a way that we can maximize the use of renewable energy with minimum cost (which includes penalties if any) without violating the SLA of a job. We find the most efficient job schedule for a particular data center with the available green energy traces and workload description. One of the other contrasting points is that this methodology mainly proposes of adjusting the mixing of different forms of renewable energy sources in order to decrease the overall cost. Whereas, our main focus is on scheduling the jobs with different available scheduling policies and analyzing their respective results.
The research presented in [14], tackles the same problem of minimizing brown energy consumption along with minimizing the cost of execution of a workload. However, the research conducted in this paper mainly deals with multiple data centers located in different time zones. The workload consists of Internet service requests. They propose a request distribution model which takes advantage of data centers located in different regions and hence, in different time zones. These data centers being in different time zones, the request is distributed in such a way that, once the workload has been divided into fractions, the maximum number of fractions is directed to data centers having maximum green energy and then to data centers with cheaper electricity costs. We, in our research, mainly focus on single data center and workload, which consists of jobs, which are bags of tasks and requests. We leverage the fact that jobs, having many tasks, usually have flexible deadlines, whereas interactive requests have comparatively stricter deadlines, and therefore they can be scheduled to attain the desired outcome of utilizing green energy to its maximum potential with cheapest possible cost.

Research in [15] shows the approaches, which are required to design sustainable high-performance data centers, to tackle the problem of utilizing green energy and decreasing the carbon footprints for a data center. They highlight that due to using of green energy and electricity from a power grid, unwanted overheads and tuning activities get introduced at a data center. This weakens up the performance of the data center to a certain extent. They propose a switching technique between grid energy and available green energy, called ISwitch. It handles hybrid energy sourced data centers. ISwitch divides the overall computational load into two fractions. One fraction that can be powered by grid energy (conventional source of energy) and the other one, which can be powered by green energy sources. These fractions are dynamically computed. However, the underlying idea still deals with switching the servers to different power states and migration of data.

**Intermittent power supplies in datacenters.** In [16], the author tackles the problem of intermittent supply of power to a date center. Since it is very difficult to predict renewable forms of energy, which include solar energy, wind energy etc. and hence to have a consistent and steady flow of power, it is very important to tackle the problem of inconsistent flow of power. The solution that the paper has provided in order to tackle this problem is Blink, which is a power driven application independent hardware software platform. It works by transitioning the servers to different states. These states are high power state when the server is active and low power state when they are inactive. The proposal also says that the data center uses only the renewable energy and is not connected to any kind of brown energy source such as a power grid. However, the algorithm that we have analyzed and proposed does not deal with changing the states of the servers. Rather we mainly are dependent on the scheduling policies, which are the driving force of this algorithm. These scheduling policies help to delay/ re-arrange certain jobs so that desired effect of maximizing green energy can be achieved. Also, our algorithm is not dependent on green energy only because of its intermittency. It is very important to not violate the Quality of Service (QOS) and therefore, in the time slots, when sufficient green energy is not available the algorithm either doesn't accept the jobs or uses brown energy with additional costing.

**Other scheduling approaches.** The scheduler that we have proposed and analyzed with different scheduling policies has some unique characteristics, differentiating it from all the other schedulers listed in [17].
Our main objective is to increase the utilization of green energy while minimizing the overall cost of execution of workload. We exploit the fact that generally batch jobs have loose deadlines and therefore they can be made to wait in order to find the best spot for their execution in the workflow. While generating the workflow, various factors which affect our algorithm are:

- Running time of a job
- Number of cores/processors required
- Deadline of the job

When we focus on cost, we focus mainly on electricity prices and the power consumption of the job. The approach that we have adopted to prevent deadline violation while generating the workflows is that we apply penalties on all those workflows (cost wise), which have deadline violations for any of the jobs. We also analyze the working of our algorithm with different scheduling policies such as:

- Least slack time first
- First come first serve
- Least running time first
- Least number of cores first

Some of the metrics that we have used to analyze the above said scheduling policies are:

- Maximum number of jobs scheduled
- Maximum green energy used
- Maximum brown energy used
- Maximum green energy wasted
- Minimum cost workflow
- Most number of times deadline violated

The work done in [18] is quite similar to what we have done in our research. However, in this paper, the author talks about a load balancing technique for multiple data centers located in different locations, and henceforth exploiting the temporal variations to utilize on-site power and available green energy. The technique proposed in this paper does not use any kind of future predictions of availability of green energy and its sources, electricity prices and demands (requirement) as per the workload. As discussed earlier, the analysis that we have done and the algorithm that we have proposed is designed for a single data center and instead of taking advantage of temporal variations of multiple data centers, we exploit the fact that batch jobs tend to have loose deadlines and therefore can be rearranged or made to wait without violation of their deadlines.

# 3. Motivation & Background

## 3.1 Motivation

As per Kyle York, board member of Cloud App, the Internet is the single biggest thing our species will ever create [19]. The amount of data that the Internet holds today is unimaginable. This indicates that today the world is actually shifting to a data intensive age. Hence, to accommodate this sudden increase of data, we are in a state where the cloud has become a necessity. For instance, at CERN, Europe's particle physics laboratory [20], computer scientists made an in-house cloud to manage and process the data generated from the Large Hadron Collider. Another example being the cloud used at NASA, which handles the data generated from the cosmological studies.

Additional reasons as for why the cloud is regarded as the fifth utility after gas, electricity, telephony and water, is that nowadays scientists do not have the time to wait for results, unlike in the past when waiting for results of the same data size could take days to receive. Computer scientist Mark Howison [21] carried out a research on marine animals where he had to analyze Ribonucleic acid (RNA) from animals related to jellyfish and coral. After the experiment was carried out, the team discovered that the local super computer at Brown University, Rhode Island was not reliable enough to carry out the processing of data. Therefore, Mark could not wait for the university team to fix this issue and therefore invoked virtual machines from Amazon. Because of Amazon's user-friendly interface, it took him only two hours to learn how to operate the cloud environment. Consequently, within 14 hours and US $61, his data analysis was completed. Therefore it becomes really important to devise strategies which could help us enjoy the benefits of cloud computing. Whilst preventing the environmental damage caused by the brown energy consumption of these cloud data centers.

Our research primarily focuses on devising an algorithm that schedules the received jobs in such a way which maximizes the use of green energy and minimizes the cost incurred to execute those jobs in a data center using green and brown energy to power itself up. The main motivation behind formulating and analyzing the algorithm is to conserve the environment and to reduce the carbon footprint of any data center, which schedules the jobs using this algorithm.

The aim of our research is to develop an energy efficient algorithm, which could be integrated with any cloud management studio, for example Aneka. In this way, these cloud management studios could be made more cost and energy efficient.

We know that the Internet is the single biggest thing our species has come across. But when we issue such statements about the network, it also becomes important that we deal with the responsibilities associated with it. All these technological advancements are to make our future sustainable and secure. All the technological benefits will mean nothing if we have no future at all.

Many scholars and researchers have done a significant amount of research around the problem of how to efficiently use renewable energy to power up cloud data centers. Mostly the research done has focused on the scheduling of interactive requests. The solutions proposed by some of them include distributing the requests amongst multiple data centers, which are geographically located at different locations and hence leveraging temporal difference. Some of the solutions include efficiently switching the states of the servers from idle to active. In this thesis, we have examined an alternate path to this problem and mainly focused on jobs. I hope that results and analysis presented in this thesis will bring us one step closer to the solution of the above-discussed problem.

## 3.2 Background

**Powering data centers with renewable energy (RE).** Renewable forms of energy are hailed as the most environmental friendly forms of energy. Their usage minimizes the generation of secondary wastage, carbon footprint, and other environmental impacts if they are used appropriately and judiciously. [22] On the other hand, non-renewable source of energy is on the brink of extinction. This is because of the rising trend of energy consumption. The speed at which these resources are being consumed has had serious environmental effects such as the release of poisonous gases, which have led to the phenomenon called global warming. This is why renewable energy is being regarded as the clean source of energy [23]. Another major reason to switch to a renewable source of energy is that due to a surge in consumption of energy, the prices of energy being produced by non-renewable forms of energy are soaring every day. This indicates that non-renewable sources of energy are being consumed at a faster pace than they are being replaced in the environment [24].
Sun is considered to be the primary source of renewable or green energy. Mainly, it provides us with sunlight, constituting of light and heat, which contributes to solar energy. Other major forms [24] of renewable energy are the wind, water, tidal and geothermal. Renewable energy can satisfy energy demands based on current and future economic trend.
However, after discussing some of their advantages, one of the most important disadvantages of using renewable energy and its sources is the intermittency in their supply. The sources of renewable energy chiefly depend upon the environmental conditions, which affects their supply and makes it intermittent. Therefore, it is very important to build a reliable renewable energy supply system.
The availability of these sources varies from place to place and their availability is not guaranteed at a certain place when they are needed [25]. For example, hydropower sources vary much depend upon the geography of the area and are mostly present near seashores and in remote areas. Tidal energy can also be obtained from places close to the seashore. Whereas wind energy can be harnessed maximum at higher altitudes where wind velocity is relatively higher and sufficient to produce electricity. These energy sources require very costly

machinery to generate electricity. This also leads to increase in the cost of bringing the electricity to cities using expensive electricity wires. One of the other problems that electricity produced by these sources is the inefficiency of storing large amounts of electricity generated by them, for later use. All of the above factors are obstacles to the growth of renewable energy and its uses. [25] As per the current records, net renewable energy production worldwide is 10% out of which 8% is in the United States.

**Comparing costs of RE sources.** Even though the sources of RE such as water, sunlight etc. are available in abundance in nature but still, to harness the electricity from these resources is still not cheap. This is because; these energy sources require highly advanced machinery and equipment to extract electricity from them.

Often, the cost associated with an RE source and its electricity production constitutes of the total cost to build and operate a power plant, initial investment, continuous maintenance, return on investment, continuous operation and fuel required to operate the whole facility. The overall cost is called Levelized energy costs (LEC) [25].

| **Power Plant Type** | **Cost $/kWh** |
|---|---|
| Coal | $0.095-0.15 |
| Natural Gas | $0.07-0.14 |
| Nuclear | $0.095 |
| Wind | $0.07-0.20 |
| Solar PV | $0.125 |
| Solar Thermal | $0.24 |
| Geothermal | $0.05 |
| Biomass | $0.10 |
| Hydro | $0.08 |

**Table 1: Cost of electricity produced by different plant types as taken from US EIA statistics and analysis from Annual Energy Outlook.**

The content of above table is based on the data provided from US Energy Information Administration (EIA) statistics and analysis from Annual Energy Outlook [25]. The data shows average LEC in dollars per kilowatt-hour of renewable and non-renewable sources of energy. The above data shows that RE sources are expensive than NRE sources. All the points, which have been discussed above, contribute in the same direction.

Even though there are many hurdles, as discussed above, in using RE as the source of electricity, still many countries and their governments are promoting the use of RE by introducing lucrative incentives for the companies using RE.

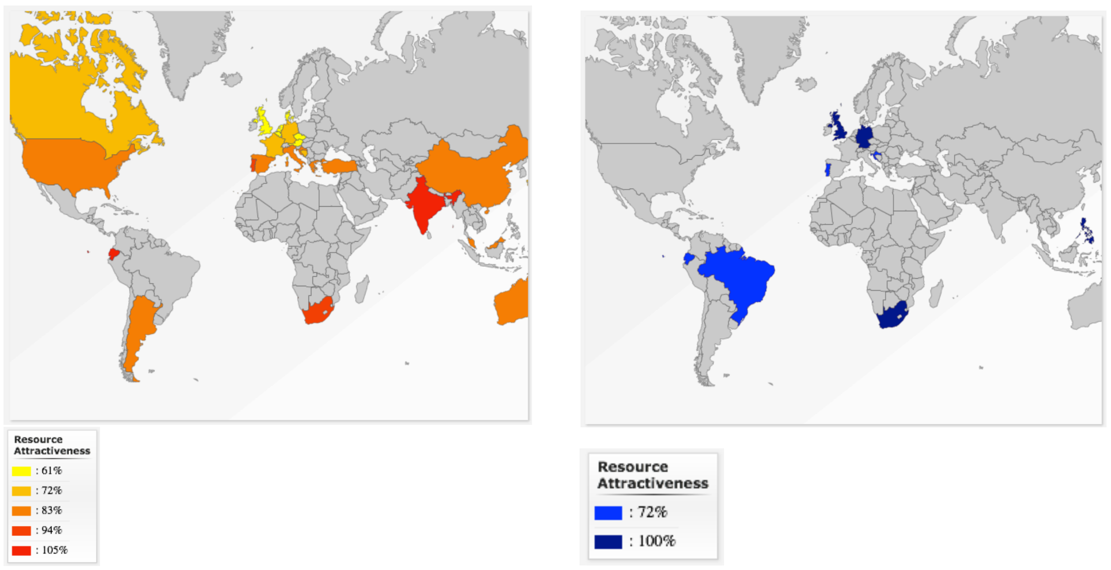

Figure 1: Country wise incentives based on solar energy and wind energy.

The two world maps presented above show all those countries, which have introduced incentives to encourage companies to use RE [26]. The left map shows countries having incentives based on solar energy and the right one shows countries having incentives based on wind energy.

These incentives include [26] [27] [28] [29]:

- Generation of electricity based incentives, for example, the producer will have to pay a least possible cost per kWh for a given period of time.
- Incentives based on investment such as tax credits for investment, VAT exemption, loan guarantees and interest-free loans.
- A legal framework to be streamlined and hurdle free which includes building regulations and streamlined planning process.
- Feed-in tariffs, which allow anyone to install a solar power system on their rooftops and then sell generated electricity back to the grid, for which power company/government pays rebate or some premium rate.

**Converting RE into electricity.** There are many processes to convert different forms of RE to electricity but we will mainly talk about solar and wind energy conversion.
To convert solar energy into electricity, solar panels are used, which produce direct current. It is then converted to alternate current by passing it through an inverter and hence, is supplied for residential/commercial use. These solar panels consist of silicon semiconductors or photovoltaic cells, which convert sunlight into electricity. When sunlight falls on the surface of these panels, the photos transfer their energy to electrons present in these silicon semiconductors. This results in negatively charged electrons to gather at one side of the cell, thus resulting in the creation of electric voltage. The current thus produced can be channeled through wiring multiple solar cells and is collected for further use.
However, wind energy requires a different mechanism to generate electricity. The kinetic energy in wind is used to generate electricity. To realize the generation of electricity, wind turbines are used. As the wind, with a certain

velocity passes through the blades of the turbine, it results in spinning of the blades and hence, the shaft. To produce electricity, a generator connected to the shaft of the turbine is used. The generator uses this motion of the shaft to rotate the rotor, which results in the creation of electromagnetic induction due to oppositely charged magnets and loops of copper wire around them.

**Supplying RE produces electricity for commercial use (data center).** As discussed earlier, once electricity from RE sources is produced it can be stored in batteries for further usage. The electricity produced is direct current. Therefore, in order to use it for datacenter purposes, an inverter has to be used, to convert the generated DC into alternating current (AC). Extra electricity produced using these means can be fed to a grid as well.
However, the study that we have done required the data center to be connected to a grid as well. This is to minimize the interruption experienced due to the intermittent availability of RE. Therefore, in the times when RE is scarce or if the electricity produced by RE cannot satisfy the demands of the workload, our algorithm uses electricity from the grid (Brown electricity) to satisfy the demand.

**Using grid electricity to reduce the cost.** Generally, datacenters have some kind of agreements with power companies, which provide them grid electricity. These agreements usually give variable pricing to data centers. There are two types of pricing, off-peak electricity price, and on-peak electricity price. Off peak electricity price is cheaper than on peak electricity price. It is generally available at night whereas on peak electricity price is available in the daytime. Therefore, in order to reduce the overall cost to execute a given workload, it will be highly beneficial for a datacenter if it can push the workload execution to nighttime, in order to avail the benefits of cheaper off-peak electricity pricing.

The algorithm that we have proposed in this thesis does not consider battery system. This means that whatever green energy is produced, if not consumed will be wasted. But our algorithm minimizes the use of green energy with different kinds of logical incentives, which we will discuss in later sections of this thesis. While scheduling the jobs, we have taken into consideration the environmental effect on green energy availability and therefore, chosen 30 minutes as our time slot period. This is because after analyzing historical data and traces of green energy, we could see that availability of green energy varies substantially in every 30-45 minutes. Our major aim is to schedule the jobs in such a way that they don’t violate their deadlines, use maximum green energy and incur the minimum cost.

## 4. Problem Definition

As discussed above, cloud computing has become one of the necessities of our lives. Today's market which dwells on data needs specialized infrastructure to gather, store and process the data, whose volume multiplies with each passing second.

In an ideal situation, the algorithms employed to run the cloud management infrastructure should aim to execute the tasks or jobs allotted to the cloud data centers in the most efficient manner. They should aim to execute the bag of jobs with the minimum cost incurred by the data center. They should aim at increasing the amount of renewable energy used to execute the jobs and henceforth, minimizing the use of a brown or non-renewable form of energy. Data centers also should not just depend upon the availability of green energy but should also be flexible enough to use brown energy when available green energy is insufficient to satisfy the energy requirements. Also, in the ideal world, the runtime of the jobs, when they are being submitted to the system for execution, would be consciously estimated along with flexible deadlines and not tighter deadlines.

However, when we discuss the current situation, there are many problems, which exist and need to be looked into. Coal deposits are decreasing day by day. This will result in an increase in the electricity cost produced using brown energy. As per the example, which was discussed in the earlier section, cloud data centers around the world consume a very large amount of electricity to function. If we continue with this pace, it will lead to the extinction of non-renewable sources of energy such as coal. Although, currently the cost of using green energy is higher than brown energy, but we predict that in future, it will decrease, keeping in mind the multiple government initiatives to encourage the power companies to do the same. Another big concern associated with the use of brown energy is the carbon footprint, which is left behind by these cloud data centers. This carbon footprint is depleting the environment around us and hence, is one of the most important causes for greenhouse effect and rise of Earth's surface temperature.

Another problem which needs to be dealt with equal importance, is how to tackle the intermittence in the availability of energy using renewable resources. Since, most of the energy produced by renewable and environment-friendly methods depend upon environmental factors such as availability of sunlight, low tides and high tides, velocity of wind etc., it is very difficult to guarantee their presence at a certain place and at a certain time when required.

Another problem that persists and is related to the jobs submitted by institutions to the cloud management studios is, in order to get their work (job or bag of jobs) done quickly or on a priority basis, an unconscious estimate of the runtime is done. Hence, the resources which are employed for the execution of the said job get wasted because they are used for more time than required. To get their jobs prioritized, these institutions sometimes recklessly give tighter deadlines and hence, other jobs are deprived of the resources and time required to run them.

Many researchers have proposed solutions to the above-mentioned problems but mostly, they have considered and leveraged the scheduling of jobs geographically. These solutions have underestimated the cost required to

transfer the data and metadata required to execute a given job at a different geographical location in order to leverage the cheap electricity prices of that location to reduce the overall cost of executing that job.
When we talk about the financial cost of using green energy as the primary source of electricity to power up the cloud data centers, we expect that the prices of electricity unit will go down. This belief has been bolstered by several governments generous initiatives that have been implemented all over the world in different countries which encourage the use of green energy for example central and state initiatives which can reduce capital cost up to 60% [30]. In-house generation of electricity using solar panels, wind mills etc. is also becoming popular all over the world as demonstrated by small and medium-sized data centers [11, 31]. We have observed that in the last 20 years, the cost of solar energy has decreased by 7 times [16, 32]. We believe that this trend will continue in future as well. If the carbon taxes which have been introduced across Asia and Europe, spread throughout the world, the financial cost associated with green energy generation should further dip down [33].
In this thesis, we propose a job-scheduling algorithm for cloud data centers in order to maximize the usage of green energy and minimize the incurred cost to execute a given job. The algorithm focuses on single data center and is flexible enough to accommodate brown energy, whenever green energy is not sufficient to satisfy the needs of a given workload. We also compare and analyze different scheduling policies, when used in association with our algorithm.
Some of the benefits of the proposed algorithm, other than maximizing green energy and minimizing the cost, are that the cost model that we have considered doesn't only depend upon the electricity prices. It focuses on and is directly proportional to the running time and nodes requirement provided in the script when a job is submitted to the algorithm. This will help to keep the unnecessary longer running times and requirement of extra nodes in check. Our algorithm also leverages upon the fact that brown electricity prices in the night are cheaper than in the day.
Optimization of utilization of green energy has become crucial for running cloud data centers in order to protect the environment. In this thesis, we propose an efficient method to do the same with its analysis and feasibility. We have incorporated different scheduling policies with our algorithm and discussed their advantages and disadvantages individually when they are used in association with the proposed algorithm.

## 4.1 Thesis Statement

In this thesis, we emphasize and encourage the use of green energy in order to power up cloud data centers. By this we mean, that a data center which is using renewable energy sources to power itself up and receiving jobs with deadlines. We present an efficient way of scheduling jobs in a cloud data center, which maximizes the use of green energy and simultaneously reduces the overall cost of executing a workload.

# 5. Scheduling Algorithm

## 5.1 Introduction

In this thesis, we have proposed a job scheduling strategy for a cloud data center. This strategy can be integrated with different cloud management platforms to efficiently schedule jobs in a cloud data center. The proposed algorithm utilizes green energy, which is used to power up the data center. The source of this green energy could be anything such as solar energy, wind energy or any other form. The algorithm proposed in this thesis is flexible enough and in green energy deficient times, it uses brown energy to satisfy the energy demand of jobs. In this manner, it does not only rely on green energy availability.
The work proposed in this thesis is greatly influenced by [34] and their strategy has been taken as the base model, upon which our work has been built on.

The proposed algorithm schedules the job in such a way that it maximizes the use of green energy and minimizes the cost incurred to execute these jobs. We have also included a basic cost model in our algorithm to calculate and reduce the cost required to execute given jobs to its minimum which will be discussed in detail in later sections of this thesis.
The basic machinery of this scheduling algorithm is the inner scheduling strategy that we have used which decides which job has to be scheduled next. To make our study meaningful and more helpful towards the cause, we have used, compared and analyzed the overall algorithm in a simulated environment with four different scheduling strategies namely:

- LSTF – Least slack time first: which picks up the job with the minimum slack time that is the difference between the current time and the latest possible start time of the given job.
- FCFS – First Come First Serve – which picks up a job depending upon its arrival time and schedules it.
- LRTF – Least Running Time First – Which picks up the job depending upon its running time and schedules it.
- Least Nodes First – Which picks up the job depending upon the number of nodes required to execute the given job and schedules it.

We have presented a detailed analysis and comparison of our algorithm with the above mentioned strategies based on factors such as cost incurred in each case, amounts of green energy used and wasted in each case, quantities of brown energy used and wasted in each case, number of jobs scheduled and number of jobs rejected in each cases respectively.
We have kept in mind that the most important factor when it comes to cloud management is the deadline of a job. Therefore, we have presented the solution which schedules the jobs in such a way that deadline of a job is not violated, even if we have to compromise a bit on cost.
The main contributions of this thesis are:

1. We propose a greedy strategy to schedule the jobs in a cloud data center in such a way that minimizes cost and maximizes the use of green energy.
2. We show the working of our algorithm integrated with four different scheduling strategies with their detailed analysis. We leave the decision in customer's hand for which strategy to use, since all the strategies have their own advantages and disadvantages.
3. We evaluate our proposed algorithm in a simulated environment with real world traces of available renewable energy, electricity prices and real one day long workloads of a data center.

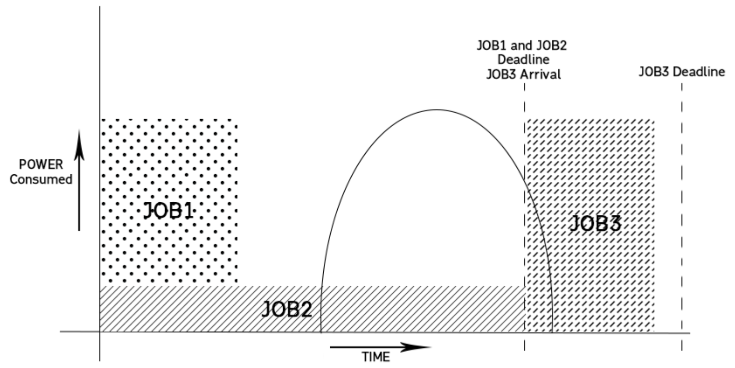

Figure 2: A conventional scheduler

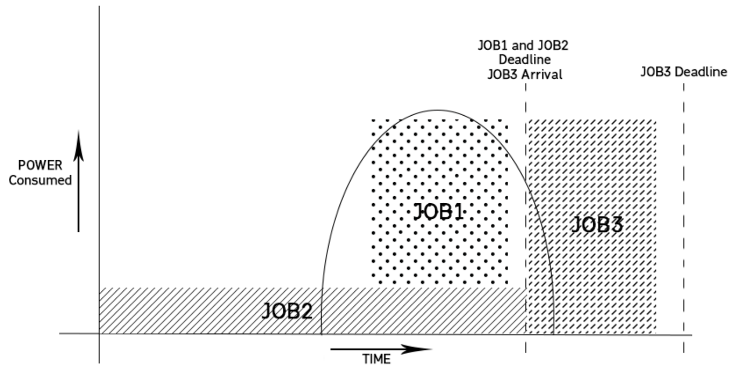

Figure 3: Scheduling using the proposed job scheduling algorithm

The above present figures show a very high-level comparison between a conventional job scheduler and the job scheduling algorithm which has been proposed in this thesis.
The blocks in the above figures represent three different jobs consuming a specified amount of energy over a certain period of time. JOB1 and JOB2 arrive at the starting of the time whereas JOB3 arrives at the same time as the deadline of the other two jobs. The elliptical line shows the availability of green energy during that time of the day. We assume that JOB3 requires a maximum number of nodes, then JOB1 and finally JOB2, which requires the least number of nodes for their execution respectively. We also assume that these jobs can run only in the fixed time slots as shown in the figures.
Figure 2 shows a conventional scheduler which schedules the jobs as they arrive in the system. It does not make any intelligent decision based on the availability

of green energy. This results in the use of more of brown energy rather than green energy, even though when green energy is available. Any scheduler, which would not consider the availability of green energy and associated electricity prices when scheduling the jobs, would presumably behave in the same fashion.
Figure 3 shows scheduling of the same jobs using the greedy job scheduling algorithm which has been proposed in this thesis. As shown in the figure, even though JOB1 and JOB2 arrive at the same time, but they are not executed as soon as they arrive. Execution of JOB1 is delayed but its deadline is never violated. The execution of JOB1 is delayed to utilize the available green energy to its maximum potential and hence use minimum brown energy. It also tries to shift the execution of the jobs towards off-peak electricity timings, since in the off-peak time we have cheapest electricity prices. We can delay the execution of the jobs but never violate the deadlines of the jobs respectively. In green energy deficient time slots, the algorithm uses brown energy to satisfy the energy requirements of the jobs in a given workflow.

In the next subsections of this thesis, we provide a detailed description of the proposed scheduling algorithm, along with our integrated green energy and cost model. Then, we describe how the proposed algorithm schedules the given jobs according to the available green energy and the cost model. We also describe the various scheduling policies used in integration with the algorithm and their advantages and disadvantages respectively.

## 5.2 Overview

The main purpose of the proposed scheduling algorithm is to schedule the jobs in such a way that maximum amount of green energy is used and minimum amount of green energy is wasted. The result or the schedule generated is strategized in such a way that minimum cost is incurred without violating the deadlines of the jobs.
Our algorithm requires the user to give the initial inputs in the form of a script, as in the case of any other jobs or tasks scheduler in cloud environments. The script provided at the beginning consists of four relevant pieces of information. The information to be provided in the script is job ID, the deadline of the job, the running time of the job, the number of nodes required to execute the job and arrival time of the job respectively. The output provided by the algorithm at the end is a workflow consisting of jobs. This workflow is a schedule suggesting which job should be allotted in which time slots keeping all the criteria in the account.
At the beginning of the algorithm, it first checks for any other queued or pending jobs from previous scheduling window. Scheduling window is the time frame within which we are scheduling the jobs. The scheduling window in the proposed greedy job scheduling algorithm is one day (24 hours) long. Once it has figured out the pending or queued jobs, it tries to schedule them first before beginning with the fresh jobs submitted to the algorithm. In our algorithm the scheduling window has been divided into 48-time slots, each time slot being half an hour (30 minutes) long. We have taken 30 minutes as the time slot length to keep weather changes in check. Weather can change rapidly and especially in a country like Australia, 30 minutes are more than enough for a sunny weather to

change into a rainy one. This will directly affect the availability of green energy and hence, the scheduling of the jobs. In our algorithm, the first slot always starts with either the current time slot or if there are multiple jobs given in the script with different arrival times, then it is the arrival time of the most recent job out of all of them.

We use a very specific cost model to minimize the cost. The model has been inspired by work done in [34]. In order to push our scheduling algorithm towards using the maximum amount of green energy, green energy price has been set to zero. Therefore, any workflow, which will be generated with maximum green energy usage, will have the minimum cost. In cases when sufficient green energy is unavailable to satisfy the energy requirement of the jobs, our algorithm switches to brown energy. The cost of brown energy has been set to normal electricity prices, which is cheaper in off peak time. Considering and leveraging this fact, whenever brown energy is required to execute a bunch of jobs, the proposed algorithm tries to push them towards night time to avail cheaper brown energy electricity prices to bring down the overall cost of a workflow. Therefore, time of usage or time of execution of the job affects the brown energy price and hence, the overall cost incurred.

Another addition to our cost model is penalties. For a given job, high-cost penalties are applied to the time slots which violate the deadline of the job. We have included this particular condition because when the algorithm generates more than one workflow (giving the user the option to choose between multiple workflow options) for a job, all the workflows will automatically get discarded which include time slots violating the deadline of the job. This is because of their very high-cost requirement due to penalty applied. Hence, this will result in execution of the jobs staying within their deadlines.

To schedule the jobs and feed them to the cost based workflow modeling, we have used four different strategies to analyze the performance of the algorithm with each of them. The first one used is LSTF i.e. the least slack time first. Slack time in the simplest words is the difference between the latest possible start time of a job and the current time. The second strategy used is FCFS i.e. first come first serve which is self-explanatory and is based on the arrival times of the jobs. The third strategy used is LRTF i.e. least running time first which is dependent on the running time of the jobs as provided by the user in the initial script. The last strategy used is least nodes first, which is dependent on the number of nodes required to execute the job, which is again provided by the user in the initial script. If a job is scheduled then the next job will be scheduled only on the remaining free time slots if, the occupied time slots do not meet the requirement of the next job in terms of nodes requirement.

As per the algorithm we assume that if a job has started on a certain node or a set of nodes, then it will finish on the same node or the same set of nodes. In order to save up energy, all the nodes which are not in use i.e. idle, are not activated and hence, wastage of energy is prevented.

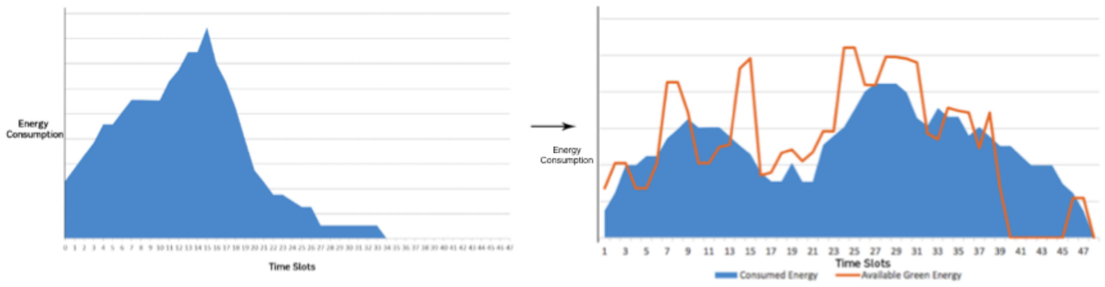

Figure 4: Figure showing how the proposed algorithm spreads out the energy consumption over time to maximize the usage of Green Energy

In the above figure, the graph on the left shows the energy consumed by a single day real world workload. It shows the total energy requirement and consumption of all the jobs in a one-day scheduling window. We assume that the scheduling window starts at 0th time slot. We also assume that the deadline of all the jobs submitted is the end of the day. Since no scheduling policy or energy aware policy is used, all the jobs get scheduled as soon as they arrive and we can see the energy peak around 15th time slot. The same load when is provided to the proposed algorithm along with real world traces of green energy availability, using least nodes first policy, it spreads the jobs in such a way that the total energy consumption is spread across the whole scheduling window and the maximum amount of green energy is used.

## 5.3 Algorithm

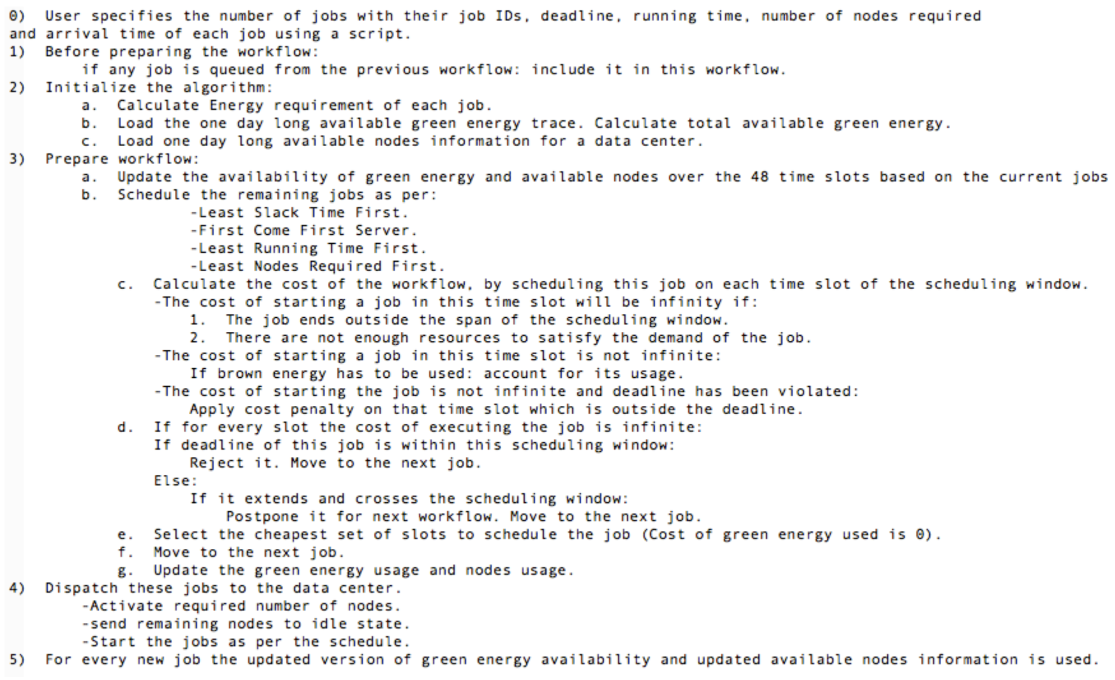
```
0)  User specifies the number of jobs with their job IDs, deadline, running time, number of nodes required
and arrival time of each job using a script.
1)  Before preparing the workflow:
        if any job is queued from the previous workflow: include it in this workflow.
2)  Initialize the algorithm:
        a.  Calculate Energy requirement of each job.
        b.  Load the one day long available green energy trace. Calculate total available green energy.
        c.  Load one day long available nodes information for a data center.
3)  Prepare workflow:
        a.  Update the availability of green energy and available nodes over the 48 time slots based on the current jobs
        b.  Schedule the remaining jobs as per:
                -Least Slack Time First.
                -First Come First Server.
                -Least Running Time First.
                -Least Nodes Required First.
            c.  Calculate the cost of the workflow, by scheduling this job on each time slot of the scheduling window.
            -The cost of starting a job in this time slot will be infinity if:
                1.  The job ends outside the span of the scheduling window.
                2.  There are not enough resources to satisfy the demand of the job.
            -The cost of starting a job in this time slot is not infinite:
                If brown energy has to be used: account for its usage.
            -The cost of starting the job is not infinite and deadline has been violated:
                Apply cost penalty on that time slot which is outside the deadline.
            d.  If for every slot the cost of executing the job is infinite:
            If deadline of this job is within this scheduling window:
                Reject it. Move to the next job.
            Else:
                If it extends and crosses the scheduling window:
                    Postpone it for next workflow. Move to the next job.
            e.  Select the cheapest set of slots to schedule the job (Cost of green energy used is 0).
            f.  Move to the next job.
            g.  Update the green energy usage and nodes usage.
4)  Dispatch these jobs to the data center.
        -Activate required number of nodes.
        -send remaining nodes to idle state.
        -Start the jobs as per the schedule.
5)  For every new job the updated version of green energy availability and updated available nodes information is used.
```

Figure 5: Cost and energy efficient job scheduling algorithm for a cloud data center.

The proposed algorithm is described above step by step. The algorithm has been inspired by work done in [34]. Line 0 describes the initial user input which user feeds to the algorithm in the form of a script. Each line of the script represents a different job which has to be scheduled. Each line of the script should have the job ID, its deadline, running time of the job, a number of nodes required and the arrival time of the job in the system. A single job in the script looks like:

JOB11 12/08/2016 18:05:00 15799 3 12/07/2016 18:05:00

In the above example, JOB11 is the job ID. 12/08/2016 18:05:00 represents the job deadline time. 15799 represent the running time of this job in seconds. The number of slots this job will use is calculated by:

$$(15799/60)/30 = 8.77 \approx 9$$

Figure 6: Slot Calculation

(15799/60) is divided by 30 because each slot is 30 minutes long. 3 represent the number of nodes required i.e. throughout the 9 slots, which this job requires to execute, it will use 3 nodes in each slot. Finally, 12/07/2016 18:05:00 represents the arrival time of the job in the system. The algorithm works in a very strict fashion such that first, it will calculate the latest possible start time with the help of expected running time, arrival time and deadline provided by the user. For example in this case, the expected running time of the job (15799/60 = 263.31 minutes and 263.31/60 = 4.38 hours) is 4 hours and 22 minutes approximately. So it cannot start at any time later than 12/08/2016 13:43:00. And hence, keeping this latest possible start time, the algorithm will start the scheduling of each job.
Line 1 and 2 represents the initialization part of the algorithm. Before starting with preparing the workflow schedule for each of the jobs to be executed, the algorithm does some basic initializations. The algorithm checks if there is any job from the previous workflow which was not scheduled in the last scheduling window. It checks if the deadline of that job is in this scheduling window. If yes, it proceeds with including that queued job to the list of jobs, which are to be scheduled in this window. Lines 2a, 2b and 2c show some basic initializations, which include loading up of the available green energy information. Another initialization is regarding the number of available nodes in the data center for which scheduling is taking place. Both of these initializations take place for 48-time slots that are one-day long scheduling window. Each time slot spans for 30 minutes. The most important initialization in this step is of the calculation of the total energy required by each job. In order to calculate the amount of energy required by each job the below-described formula [34] has been used:

**Job Energy Constant (JEC)** = [JGT + JIT + JST + SE + SC]
**Individual Job Energy Requirement** = [JEC + (JRT *Job Run Time as provided by the user in the script)]

Formula 1: Formulae used to calculate individual job energy requirement

The various acronyms used in the above formulae are Job Gather Time (JGT), which is the average time required to gather the various resources required to run a job on a node. Job Initialization Time (JIT) is the average time required to initialize a job which might include initializing objects, variables, data structures etc. for the execution of the job. Job Split Time (JST) is the average time required to split the job over multiple time slots if required. Switch Energy (SE) is the average energy required to change the state of a node from active to idle or vice versa. It is the same energy required to alter the state of the server as well, as in

activating or deactivating the server. Server Consumption (SC) is the average energy consumed by one server. All of these are constants and their values are per second. The sum of all these constants gives us the resultant as Job Energy Constant. To calculate the individual job's energy requirement, JEC is added to the product of running time of a job provided by the user in the initial script and average energy required to a run a job per second over a node (constant).
Line 3 shows the steps to generate a workflow for the jobs fed to the algorithm through the script.
Line 3a describes the very first step of workflow creation. The algorithm updates the availability of green energy over the 48-time slots. This takes place after subtracting the green energy which has already been consumed by the previously scheduled jobs in this scheduling window from the overall green energy available for the data center. The same takes place with information regarding the availability of nodes. If one node has been allotted to a job, or if a job is already running on a node then that node will not be used for any other job i.e. no other job will be scheduled on that node. In our algorithm, we have assumed that the nodes are not shared between two jobs. A node which is already in use or has been allotted will not be used for further scheduling process.
Once all the updates and initializations have been done, then the algorithm proceeds to order the jobs for scheduling into a workflow. In our algorithm and experiments, we have used four different ordering policies to test the performances and various other factors. The four policies used are:

- ***Least Slack Time First***
  As per this policy, the difference between the current time in the scheduling window and the latest possible start time of the job is calculated, which is the slack time of a job. The job having the least slack time is sent for scheduling first. For example, if there are two jobs Job A and job B. Running time of job A is 1 hour i.e. 3600 seconds and running time of job B is 2 hours i.e. 7200 seconds and both end at 5 am. Let's say current time is 12 am. Then the latest possible start time for job A would be 4 am and job B would be 3 am. The slack time of job A would be 4 hours (4 am – 12 am) and slack time for job B would be 3 hours (3 am – 12 am). As per LSTF policy, job B would be sent before job A for scheduling.

- ***First Come First Serve***
  Jobs are reordered on the basis of their arrival times. The job with the arrival time closest to the current time is sent for scheduling first. For example, if there are two jobs, job A and job B and their arrival times are 12 am and 12.30 am respectively. In this case, Job A will be scheduled before job B.

- ***Least Running Time First***
  In this ordering policy, jobs are re ordered on the basis of their running time provided in the initial script. Job with smallest running time is sent for scheduling first. For example, if there are two jobs, A and B and A's running time is 3600 seconds and B's running time is 7200 seconds, then A will be sent for scheduling before B as per LRTF.

- ***Least Nodes First***
  As per this policy, jobs are reordered as per the number of nodes required to execute them. A job requiring the least number of nodes is sent for scheduling first. For example, if there are two jobs, A and B and A's nodes requirement is 5 and B's nodes requirement is 3, then B will be sent for scheduling before A as per this ordering policy.

In step 3b, as soon as one job is ordered it is then sent for scheduling and is removed from the initial list of jobs which are to be scheduled. Step 3c in the algorithm shows how the cost for a particular job is calculated and what factors affect the overall cost. When a job reaches step 3c, the cost for executing the job at each time slot is calculated for every available slot in the scheduling window. In order to prioritize green energy over brown energy, we have assumed that the (i) cost of using green energy is 0. If the job ends in this scheduling window and we have enough resources to satisfy its resource requirement, the (ii) algorithm accounts for brown energy as well wherever the brown energy is used. (iii) If the cost of execution is not infinity but the time slot schedule of a job ends outside the deadline of the job, all the slots outside the deadline of the job will be penalized with a cost penalty.
The cost of scheduling a job will be infinite only in two cases:

- If the job ends outside this scheduling window.
- The data center does not have enough resources, in terms of nodes availability to satisfy the demands of a job to be executed.

Infinity cost is a representation of the inability of the algorithm or the data center to accommodate a job in this scheduling window.
The time slot schedule with the cheapest cost is selected as output and included in the overall schedule as the workflow. In the overall schedule, which is prepared by the algorithm, if there is more than one schedule of the time slots with the same amount of cost, then the algorithm tries to allot those slots to the execution of this job, which consume a minimum amount of brown energy and the maximum amount of green energy. This way it spreads out the whole energy consumption over the time slots in a consistent way. A job will not be scheduled in this window if it satisfies any of the below-mentioned scenarios:

- All the schedules generated for the job have cost as infinity. This indicates that either the data center is out of resources or job ends outside this scheduling window.
- If the job's deadline is in this scheduling window, but none of the slots are free or cannot accommodate this job, then also this job will be rejected and won't be scheduled.

For all those cases, in which job gets rejected, the user can either submit the job for the next round of scheduling or if the job is queued, the algorithm will automatically try to schedule it in the next round. Once the job has been scheduled, the overall green energy availability and nodes availability information is updated. This updated information is used for further scheduling of the next job. This is represented by lines 3e to 3g.
Step 4 and 5 show the final steps of the algorithm. Once the workflow has been prepared for all the jobs submitted to the algorithm, they are sent for the execution to the data center. As per the per time slot node requirement of the

workflow, data center switches nodes from deactivated/idle state to activated/ active state. On these active nodes, the jobs are started as per the schedule and hence the whole workflow is executed in the same fashion.

## 6. Performance Evaluation

### 6.1 Experimental Setup

**Software**. We have evaluated the algorithm which has been proposed in this paper by writing its code in C# language. The code involves approximately 800 lines of coding. To run different policies and analyze them, four separate codes were written, each with a different ordering policy. We created a SLURM [17] like script in order to feed workload information into the algorithm. We created this script in Microsoft Notepad only and the program picked up the JOB information from this text file. Another Microsoft Notepad file was used in order to provide renewable energy availability information to the algorithm. This information about renewable energy availability was provided slot wise to the algorithm.

**Workload.** In order to test our algorithm, we used two workloads. The first workload consists of 10 JOBs. These JOBs were part of workload 2, which will be discussed in the next section, but with stricter deadlines. The purpose of using this workload was to study the initial results of the algorithm and the behavior of the energy graph. The sole purpose of using this workload was to check if the algorithm was giving the desired results or not. This workload comprised of a diverse bag of JOBs. The initial script for this workload is present below:

```
JOB1 12/07/2016 13:55:00 2509 03 12/07/2016 12:00:00
JOB2 12/07/2016 14:00:00 2000 01 12/07/2016 13:00:00
JOB3 12/08/2016 18:00:00 20000 05 12/07/2016 18:00:00
JOB4 12/07/2016 15:00:00 3600 01 12/07/2016 13:00:00
JOB5 12/07/2016 15:55:00 3000 01 12/07/2016 13:00:00
JOB6 12/07/2016 14:00:00 2000 01 12/07/2016 13:00:00
JOB7 12/07/2016 16:00:00 1800 01 12/07/2016 13:00:00
JOB8 12/07/2016 15:00:00 3600 01 12/07/2016 13:00:00
JOB9 12/07/2016 15:55:00 3000 01 12/07/2016 13:00:00
JOB10 12/07/2016 13:55:00 2509 03 12/07/2016 12:00:00
```

Script 1: Initial Script for 10 JOBs of workload 1

As can be seen from this script it consisted of 7 JOBs requiring only one node for their processing, 2 JOBs requiring 3 nodes and one JOB requiring 5 nodes for the same. As per this workload, two JOBs arrived at 12/07/2016 12:00:00, 7 JOBs arrived at 12/07/2016 13:00:00 and one JOB arrived at 12/07/2016 18:00:00. The deadline time for each of these JOBs vary, earliest being JOB 1 and JOB 10 with 12/07/2016 13:55:00 and the latest being JOB 3 with a deadline as 12/08/2016 18:00:00. The running time for the JOBs in this workload greatly varies too. The least running time is 1800 seconds, which is 30 minutes for JOB 7. The maximum running time being 20,000 seconds for JOB 3, which is 333.33 minutes or 5.55 hours. The scheduling window for this workload spans over 48-time slots; each time slot is 30 minutes long and hence, in totality is 24 hours long.

The other workload that we used consisted of one day long actual workload trace provided in [18] for the experiment. This workload consisted of 40 JOBs each with the deadlines ending within the 24 hours scheduling window. The detailed workload is present below, with the initial script, which was fed to the algorithm:

```
JOB2 12/08/2016 18:00:00 48575 1 12/07/2016 18:00:00
JOB3 12/08/2016 18:00:00 33934 2 12/07/2016 18:00:00
JOB8 12/08/2016 18:02:00 60332 1 12/07/2016 18:02:00
JOB11 12/08/2016 18:05:00 15799 3 12/07/2016 18:05:00
JOB13 12/08/2016 18:12:00 35523 1 12/07/2016 18:12:00
JOB9 12/08/2016 18:13:00 33269 1 12/07/2016 18:13:00
JOB21 12/08/2016 18:13:00 27490 1 12/07/2016 18:13:00
JOB23 12/08/2016 18:17:00 30188 1 12/07/2016 18:17:00
JOB30 12/08/2016 18:29:00 17537 1 12/07/2016 18:29:00
JOB31 12/08/2016 18:34:00 30317 1 12/07/2016 18:34:00
JOB37 12/08/2016 18:38:00 28063 1 12/07/2016 18:38:00
JOB16 12/08/2016 19:03:00 25217 1 12/07/2016 19:03:00
JOB52 12/08/2016 19:22:00 12399 4 12/07/2016 19:22:00
JOB54 12/08/2016 19:34:00 20208 1 12/07/2016 19:34:00
JOB62 12/08/2016 19:51:00 20579 1 12/07/2016 19:51:00
JOB35 12/08/2016 20:01:00 23219 1 12/07/2016 20:01:00
JOB59 12/08/2016 20:02:00 16761 1 12/07/2016 20:02:00
JOB50 12/08/2016 20:24:00 7833 5 12/07/2016 20:24:00
JOB15 12/08/2016 21:01:00 11876 1 12/07/2016 21:01:00
JOB77 12/08/2016 21:20:00 21046 1 12/07/2016 21:20:00
JOB78 12/08/2016 21:42:00 12850 3 12/07/2016 21:42:00
JOB82 12/08/2016 21:56:00 13367 1 12/07/2016 21:56:00
JOB25 12/08/2016 22:30:00 11719 4 12/07/2016 22:30:00
JOB95 12/08/2016 23:00:00 16273 1 12/07/2016 23:00:00
JOB45 12/08/2016 23:01:00 12888 1 12/07/2016 23:01:00
JOB113 12/08/2016 23:11:00 19518 1 12/07/2016 23:11:00
JOB128 12/08/2016 23:33:00 22722 1 12/07/2016 23:33:00
JOB130 12/08/2016 23:39:00 13705 1 12/07/2016 23:39:00
JOB118 12/08/2016 23:46:00 13909 1 12/07/2016 23:46:00
JOB55 12/09/2016 0:14:00 5971 5 12/08/2016 0:14:00
JOB41 12/09/2016 0:14:00 18214 1 12/08/2016 0:14:00
JOB12 12/09/2016 0:36:00 11504 1 12/08/2016 0:36:00
JOB143 12/09/2016 0:39:00 9614 2 12/08/2016 0:39:00
JOB72 12/09/2016 0:57:00 4822 3 12/08/2016 0:57:00
JOB151 12/09/2016 1:12:00 20309 1 12/08/2016 1:12:00
JOB87 12/09/2016 1:34:00 8234 2 12/08/2016 1:34:00
JOB155 12/09/2016 1:35:00 5133 2 12/08/2016 1:35:00
JOB156 12/09/2016 1:39:00 10052 1 12/08/2016 1:39:00
JOB1 12/09/2016 1:42:00 18183 1 12/08/2016 1:42:00
JOB89 12/09/2016 2:14:00 14630 1 12/08/2016 2:14:00
```

**Script 2: Initial script for day long JOBs scheduling experiment of workload 2**

This workload represents actual real world JOBs which were submitted to the cloud system for scheduling. In the overall workload there are 29 JOBs which required 1 node for their execution, 4 JOBs which required 2 nodes for their

execution, 3 JOBs which required 3 nodes, 2 JOBs which required 4 and 2 JOBs which required 5 nodes for their execution. The arrival time of these JOBs varies, the earliest being JOB 2 with arrival time as 12/07/2016 18:00:00 and the last one to arrive being JOB 89 with arrival time as 12/08/2016 2:14:00. The deadlines for these JOBs being one day long that is if the arrival time of the JOB is 12/07/2016 18:00:00 then the deadline of the JOB is 12/08/2016 18:00:00, i.e. 24 hours window to schedule a JOB. The running time for these JOBs varies in time. The smallest running time being 4822 seconds, which is 1.34 hours for JOB 72. The largest running time being 60,332 seconds, which is 16.75 hours for JOB 8. The scheduling window for this workload also spans over 48-time slots; each time slot is 30 minutes long and hence, in totality is 24 hours long.
As it is clear from the scripts above, that we tried to cover as many variations with respect to the JOBs, their running times, their arrival times, their deadlines and nodes requirement as possible in the workloads. In the results section, we will be covering the behavior of the algorithm with respect to both the workloads and how the results varied, with respect to the 4 ordering policies.

**Renewable Energy and power consumption.** The renewable energy information which was fed into the algorithm in each experiment is a real world trace of available renewable energy. The source of this trace is Lyon, from where we got this trace for experimental purposes as is discussed in [35]. The trace spans over the 24 hour time period and fits in 48-time slots. The trace provided to us was a standardized form of available renewable energy i.e. spanning from 0 to 1, where 0 denotes unavailability of renewable energy and when available renewable energy is enough to satisfy the overall energy demand, then it is 1. The trace is plotted in graph 2.
In order to compute the amount of energy required by each JOB, as discussed earlier, the various constants used were [34] JOB initialization time, which uses 140W, the splitting of a JOB requires 90W, gathering of a JOB (after split) requires 102W, the switching used in the process requires 55W and a server consumes 30W. A JOB running constantly consumes 105W. All of these values have been taken from [34] and are in W per hour. When these values were converted into their corresponding per second values, they are:

| | |
|---|---|
| Job Initialization Time | 0.039 W |
| Job Split Time | 0.025 W |
| Job Gather Time | 0.0283 W |
| Switch Energy | 0.0153 W |
| Server Consumption | 0.008 W |
| Job Running Time | 0.0292 W |

Table 2: Energy consumption values, in watts per second

A server when in the idle state takes about 8.6W per hour as per [34].

**Electricity Prices.** We have used a very popular model for electricity pricing in the algorithm, the on/off-peak electricity pricing. The on peak time is when there is huge demand for electricity. The off-peak time is when the demand for electricity is below a certain threshold. Generally, the on peak time is mostly in the daytime and at this time electricity prices are higher. Whereas, in the off-peak time, which is nighttime, the electricity prices are cheaper than the on peak timing. In Australia, the on peak timings are 7 AM to 10 PM and off peak timings are 10 PM to 7 AM [12, 36]. The on-peak price in Victoria is $1.20 and the off-peak electricity tariff is $0.84 [13, 37].

**Penalty.** In our algorithm, we have adopted a penalizing strategy to demotivate the unconscious estimation of the deadline of a JOB by the user. Any time slot, which will be on the schedule for a JOB, such that it violates the deadline of the JOB, will be penalized. The sole motivation of this strategy is to increase the cost of the workflow, which includes time slots, which violate the deadline for a JOB, such that eventually, their overall cost will shoot up. This will help the algorithm to choose a cheaper workflow over the expensive one and hence, will prevent it from choosing the workflow having time slots which violate JOB's deadline. **The penalty included in our algorithm is $100 per slot.**

**Standardization of green energy trace.** In order to standardize the available energy trace values, the strategy which we adopted was that we plotted the graph of the energy required by the JOBs, such that the JOBs were ordered in the order of their arrival time, without any kind of ordering policy. The graph (Workload 2 case) is present below:

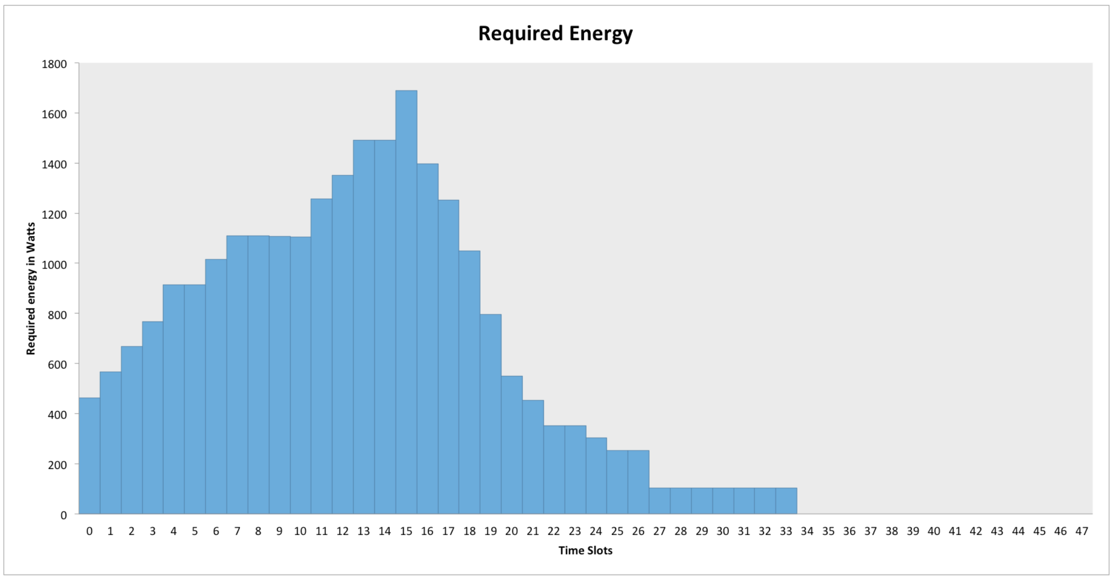

Figure 7: Overall energy requirement if the jobs were ordered in the order of their arrival time.

Once this figure was created, we found the area under the curve. The area under this figure was 24528.85 time slot-watt. The second step was to plot the graph of the available renewable energy trace. The figure is present below:

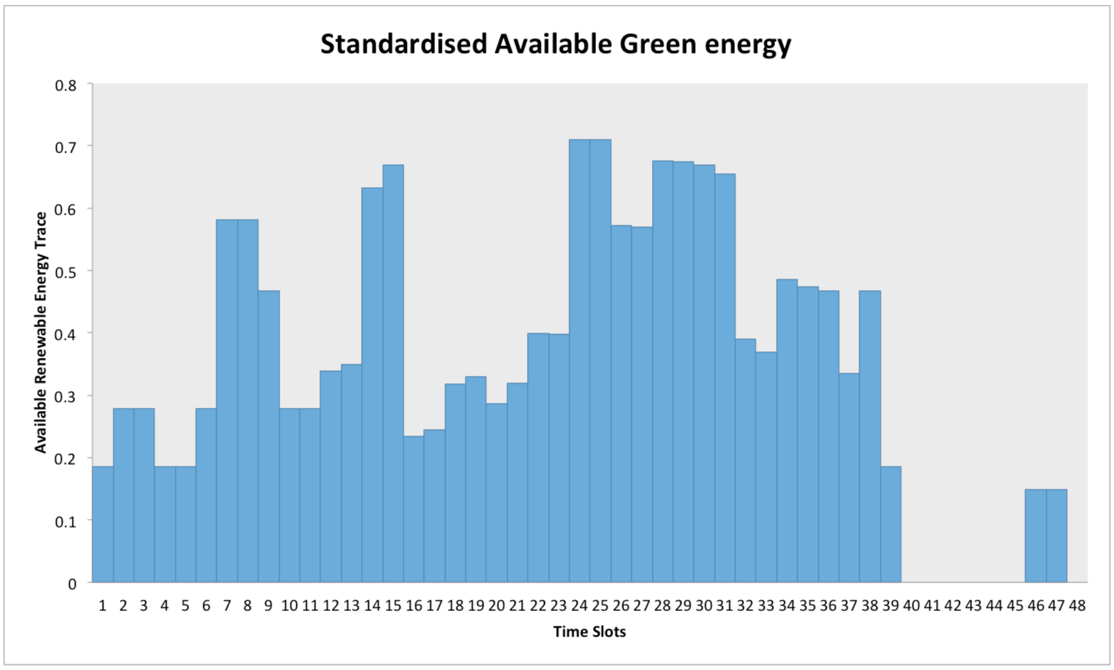

**Figure 8: Standardised available green energy trace**

We found the area under this curve as well. The area under figure 7 was 16.70 time slot-watt. These figures were plotted with the help of MINITAB [14, 38] and area was found with the help of Microsoft Excel. In order to standardize the renewable energy trace values, we multiplied each trace value with a factor of 1468.26 (24528.85/16.70 = 1468.26) by equating the two areas in the two figures present above. The resulting trace values, which were fed to the algorithm, are present in the below table:

| TIME SLOTS | RENEWABLE ENERGY AVAILABLE (Standardized values) | Resultant values in Watts (Standardized values * standardization factor) |
|---|---|---|
| 0 | 0.19 | 272.10 |
| 1 | 0.28 | 409.42 |
| 2 | 0.28 | 409.42 |
| 3 | 0.19 | 272.10 |
| 4 | 0.19 | 272.10 |
| 5 | 0.28 | 409.42 |
| 6 | 0.58 | 853.18 |
| 7 | 0.58 | 853.18 |
| 8 | 0.47 | 685.97 |
| 9 | 0.28 | 409.42 |
| 10 | 0.28 | 409.42 |
| 11 | 0.34 | 498.15 |
| 12 | 0.35 | 512.05 |
| 13 | 0.63 | 928.04 |
| 14 | 0.67 | 982.78 |
| 15 | 0.23 | 342.94 |

| | | |
|---|---|---|
| 16 | 0.24 | 358.58 |
| 17 | 0.32 | 465.85 |
| 18 | 0.33 | 483.22 |
| 19 | 0.29 | 420.67 |
| 20 | 0.32 | 469.32 |
| 21 | 0.40 | 584.93 |
| 22 | 0.40 | 584.06 |
| 23 | 0.71 | 1041.71 |
| 24 | 0.71 | 1041.71 |
| 25 | 0.57 | 838.89 |
| 26 | 0.57 | 836.28 |
| 27 | 0.68 | 991.32 |
| 28 | 0.67 | 990.45 |
| 29 | 0.67 | 981.76 |
| 30 | 0.65 | 960.91 |
| 31 | 0.39 | 572.00 |
| 32 | 0.37 | 540.72 |
| 33 | 0.49 | 712.91 |
| 34 | 0.47 | 696.40 |
| 35 | 0.47 | 685.97 |
| 36 | 0.34 | 492.07 |
| 37 | 0.47 | 685.97 |
| 38 | 0.19 | 272.10 |
| 39 | 0.00 | 0.00 |
| 40 | 0.00 | 0.00 |
| 41 | 0.00 | 0.00 |
| 42 | 0.00 | 0.00 |
| 43 | 0.00 | 0.00 |
| 44 | 0.00 | 0.00 |
| 45 | 0.15 | 218.70 |
| 46 | 0.15 | 218.70 |
| 47 | 0.12 | 168.85 |

**Table 3: Resultant available renewable energy values which were fed to the algorithm**

The method was applied for Workload 1 case also and the factor to be multiplied with the standardized green energy values was much smaller i.e. 174.092.

**Selecting the appropriate number of nodes.** In figure 6, when we plotted the JOBs without any ordering policy and just on the basis of their arrival time, we could see that most of the JOBs were scheduled in time slot 15.The total number of nodes required to execute the overall schedule is 49, such that none of the JOBs get left out because of unavailability of resources. Hence, the number of nodes present in the data center, for simulation purposes, in our algorithm is 49.

**Output.** The output of the algorithm looks somewhat like:

```
Job 130 Requirement:  1 Energy Requirement: 400.3016
```

```
 0 1 2 3 4 5 6 7
 Cost of this list is: 360.144

 Job 12 Requirement:  1 Energy Requirement: 336.0324
40 41 42 43 44 45 46
Cost of this list is: 201.944
```

The output shown above is just a very small part of the overall workflow. It shows how Job 130 and Job 12 should be scheduled in the overall workflow, the energy requirements of these two JOBs respectively and if these schedules are chosen for these two JOBs then the cost for executing these JOBs respectively.

## 6.2 Experimental Results

In this section, we will analyze the results obtained from the experiments. We will mainly elaborate the workflow obtained in each of the policies, in each of the two cases i.e. workload 1 and workload 2. The focus of this detailed analysis would be the green energy usage behavior of each scenario. We will focus on the cost associated with the resultant schedule. We will discuss and analyze the results case wise.

**Case 1: Workload 1**

As discussed earlier, in this case, 10 JOBs were used chiefly to test the behavior of the algorithm with respect to the availability of green energy. As per the initial script provided in the beginning, 10 JOBs were ordered in different orders because of four different ordering policies used in scheduling. The tables present below show the order of the JOBs in each of the four cases, with different ordering policies, along with a schedule of the time slots for their execution in each of the cases.

| **Job Names (In the order of scheduling)** | **Time Slots Schedule** |
|---|---|
| JOB2 | 3,4 |
| JOB6 | 2,3 |
| JOB1 | 4,5 |
| JOB10 | 4,5 |
| JOB4 | 5,6 |
| JOB8 | 5,6 |
| JOB5 | 6,7 |
| JOB9 | 6,7 |
| JOB7 | 1 |
| JOB3 | 7,8,9,10,11,12,13,14,15,16,17,18 |

Table 4: LSTF Workflow schedule for workload 1

| Job Names (In the order of scheduling) | Time Slots Schedule |
|---|---|
| JOB1 | 3,4 |
| JOB10 | 4,5 |
| JOB2 | 3,4 |
| JOB4 | 5,6 |
| JOB5 | 6,7 |
| JOB6 | 3,4 |
| JOB7 | 1 |
| JOB8 | 4,5 |
| JOB9 | 6,7 |
| JOB3 | 5,6,7,8,9,10,11,12,13,14,15,16 |

Table 5: FCFS Workflow schedule for workload 1

| Job Names (In the order of scheduling) | Time Slots Schedule |
|---|---|
| JOB7 | 3 |
| JOB2 | 3,4 |
| JOB6 | 1,2 |
| JOB1 | 4,5 |
| JOB10 | 4,5 |
| JOB5 | 6,7 |
| JOB9 | 6,7 |
| JOB8 | 1,2 |
| JOB4 | 5,6 |
| JOB3 | 6,7,8,9,10,11,12,13,14,15,16,17 |

Table 6: LRTF Workflow schedule for workload 1

| Job Names (In the order of scheduling) | Time Slots Schedule |
|---|---|
| JOB2 | 3,4 |
| JOB4 | 5,6 |
| JOB5 | 6,7 |
| JOB6 | 1,2 |
| JOB7 | 1 |
| JOB8 | 2,3 |
| JOB9 | 6,7 |
| JOB1 | 3,4 |
| JOB10 | 4,5 |
| JOB3 | 7,8,9,10,11,12,13,14,15,16,17,18 |

Table 7: Least Nodes First for workload 1

The figures present below show the above scheduling in each case, along with a comparison with available green energy. The orange bars show energy per watt consumed in a given time slot. The blue line shows the trend of the available green energy (standardized form).

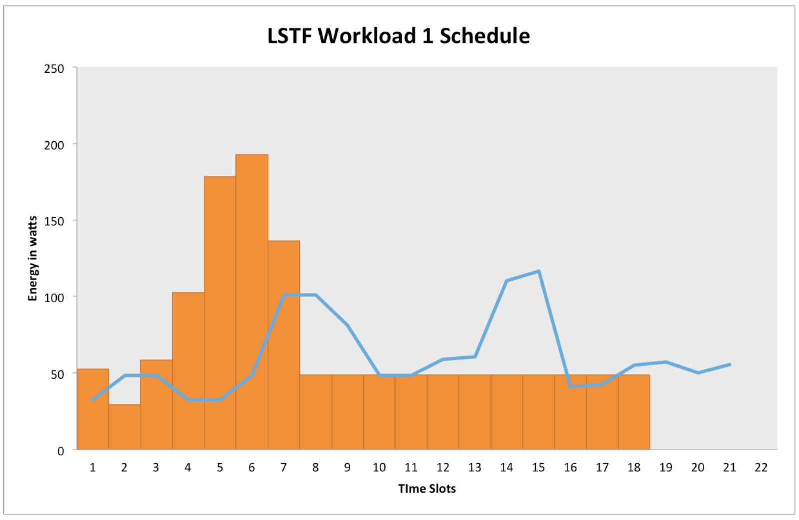

Figure 9: Schedule generated through LSTF for workload 1

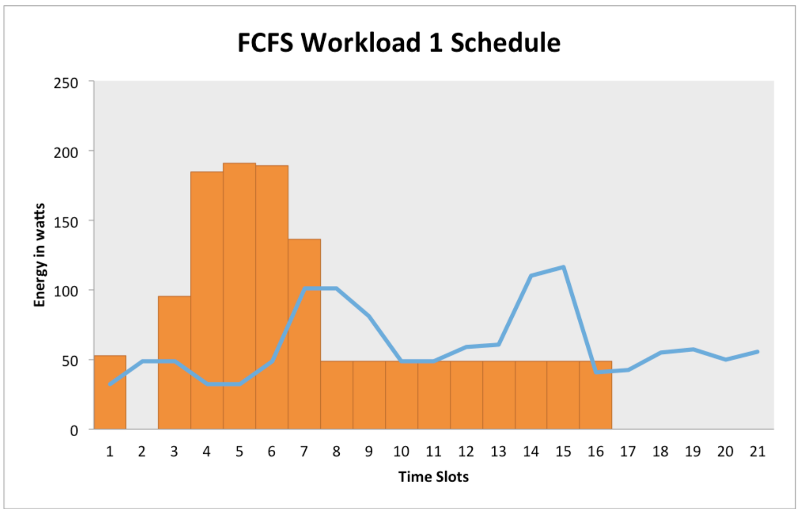

Figure 10: Schedule generated through FCFS for workload 1

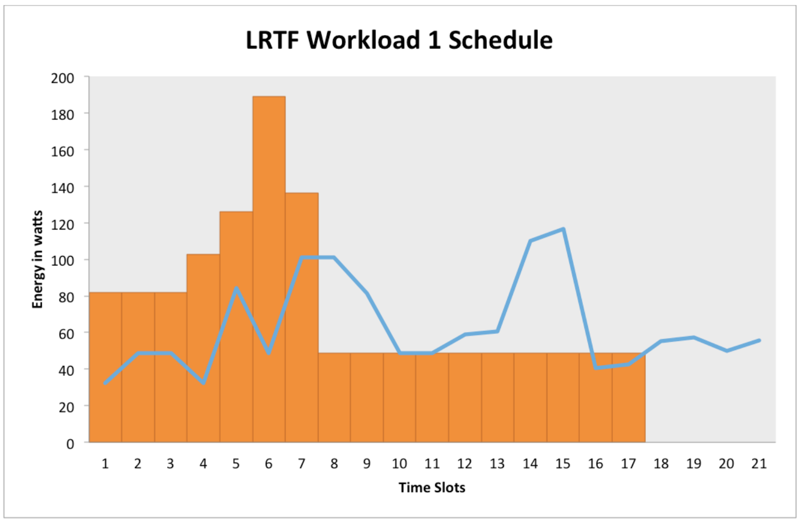

Figure 11: Schedule generated through LRTF for workload 1

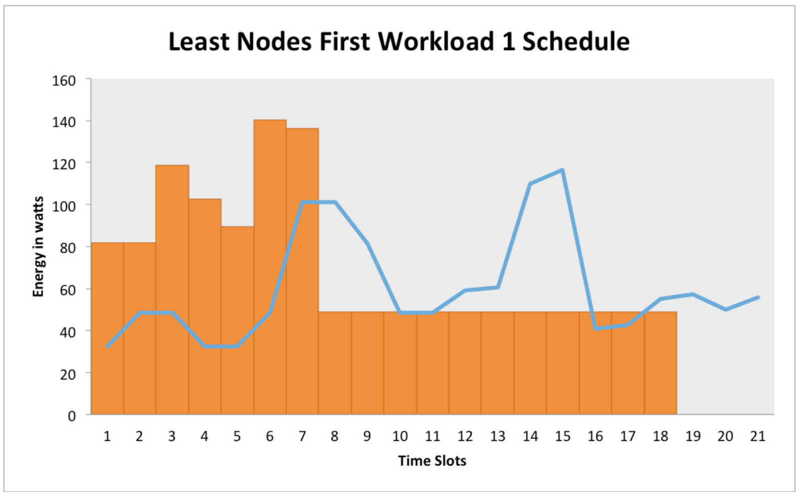

Figure 12: Schedule generated through least nodes first policy for workload 1

The above figures show the energy consumed per slot, when the 10 JOBs or workload 1 was scheduled as per each policy.

From the above figures, it can be clearly concluded that mostly after time slot 8, the plots have a similar trend. This is why in our analysis we will mainly focus on the plot of the figures before time slot 8. Again, this experiment was conducted to check the behavior of the overall algorithm. Our major analysis has been done with workload 2, which follows after this section since it consists of real world workload. One of the most important observations from the above figures is that in all the four cases, one or more than one JOB (s) could have been scheduled in time slot 8,9,14 and 15 because, in all of the cases, there is a considerable amount of green energy which is unused in these time slots. This is one of the main drawbacks of the experiment, of selecting such a small number of JOBs for a workload with stricter deadlines and also one of the other reasons, which motivated us to conduct an experiment with the real workload, and is presented in the next case.

The tables present below show the total green energy, total brown energy and overall cost of the schedule created by each of the policies respectively.

| Ordering Policy | Total Green Energy Used (in Watts) |
|---|---|
| Least Slack Time First | 845.28 |
| First Come First Server | 724.85 |
| Least Running Time First | 867.92 |
| Least Nodes First | 864.59 |

Table 8: Total green energy used by each policy for workload 1

| Ordering Policy | Total Brown Energy Used (in Watts) |
|---|---|
| Least Slack Time First | 441.18 |
| First Come First Server | 561.63 |
| Least Running Time First | 418.57 |
| Least Nodes First | 421.89 |

Table 9: Total green energy used by each policy for workload 1

| Ordering Policy | Total Cost of the Overall Schedule (in $) |
|---|---|
| Least Slack Time First | 529.40 |
| First Come First Server | 673.96 |
| Least Running Time First | 502.28 |
| Least Nodes First | 506.27 |

Table 10: Total cost of the overall schedule as per each policy

Some of the important points which have been highlighted from the above data are the maximum green energy was used by least running time first (LRTF) policy. The minimum amount of green energy in this experiment was used by first come first serve (FCFS) policy. Another point to be focused on is the minimum brown energy was used by Least running time first and maximum brown energy was used by first come first serve. We could also see that the disparity amongst LRTS, LSTF and Least nodes first is not that much, when compared to the results of FCFS. The same is highlighted in cost table (rounded off up to 2 places) as well, where we can see that maximum cost was incurred by FCFS's work schedule where as the minimum cost was incurred by LRTF for the same workload. This cost was calculated by taking green energy as 0 and brown energy as $1.20 per watt, which is the day price for electricity. In this case, since the number of JOBs was very less that is 10, and these JOBs had strict deadlines, all of these JOBs were scheduled in the first 20 slots of the overall schedule produced by all of the policies.
Another observation that can be made from the figures is that even though FCFS is the most expensive and least effective policy when it comes to the cost incurred to execute all the JOBs or amount of green energy consumed, however, it took the least number of slots to schedule and execute all the slots. That is, in the schedule generated by FCFS, the workflow ends at slot 16 whereas, for the same workload, LSTF and least nodes first take the maximum number of slots to schedule all the JOBs which are 18-time slots.
From these observations we can conclude that for workload 1, that is workload consisting of fewer JOBs, LRTF was the most efficient with respect to usage of green energy and hence, the cost incurred to execute the overall schedule. But for a user, for whom time is more important than the overall cost of the execution schedule and utilization of green energy, FCFS will be the more efficient one.

**Case 2: Workload 2:**
As discussed earlier, in this case, a total of 40 real time JOBs [18] were used and fed to the algorithm. The initial script which was used is present under the workload title. The standardized values of the green energy used in this experiment have been also provided in the previous section. We will discuss the results of all the four policies, when this real world workload was provided to the algorithm. Table 13 shows the energy required by each of the JOB, which was calculated before the scheduling of these JOBs.

| JOB IDs | Energy Required (in Watts) | JOB IDs | Energy Required (in Watts) |
|---|---|---|---|

| | | | |
|---|---|---|---|
| JOB2 | 1418.5056 | JOB78 | 375.3356 |
| JOB3 | 990.9884 | JOB82 | 390.432 |
| JOB8 | 1761.81 | JOB25 | 342.3104 |
| JOB11 | 461.4464 | JOB95 | 475.2872 |
| JOB13 | 1037.3872 | JOB45 | 376.4452 |
| JOB9 | 971.5704 | JOB113 | 570.0412 |
| JOB21 | 802.8236 | JOB128 | 663.598 |
| JOB23 | 881.6052 | JOB130 | 400.3016 |
| JOB30 | 512.196 | JOB118 | 406.2584 |
| JOB31 | 885.372 | JOB55 | 174.4688 |
| JOB37 | 819.5552 | JOB41 | 531.9644 |
| JOB16 | 736.452 | JOB12 | 336.0324 |
| JOB52 | 362.1664 | JOB143 | 280.8444 |
| JOB54 | 590.1892 | JOB72 | 140.918 |
| JOB62 | 601.0224 | JOB151 | 593.1384 |
| JOB35 | 678.1104 | JOB87 | 240.5484 |
| JOB59 | 489.5368 | JOB155 | 149.9992 |
| JOB50 | 228.8392 | JOB156 | 293.634 |
| JOB15 | 346.8948 | JOB1 | 531.0592 |
| JOB77 | 614.6588 | JOB89 | 427.3116 |

Table 11: Energy requirement of each JOB in workload 2

Table 14 represents the order in which the JOBs were sent for scheduling as per the different policies.

| LSTF | LRTF | FCFS | Least Nodes First |
|---|---|---|---|
| JOB2 | JOB72 | JOB2 | JOB2 |
| JOB3 | JOB155 | JOB3 | JOB8 |
| JOB8 | JOB55 | JOB8 | JOB13 |
| JOB11 | JOB50 | JOB11 | JOB9 |
| JOB13 | JOB87 | JOB13 | JOB21 |
| JOB9 | JOB143 | JOB9 | JOB23 |
| JOB21 | JOB156 | JOB21 | JOB30 |
| JOB23 | JOB12 | JOB23 | JOB31 |
| JOB30 | JOB25 | JOB30 | JOB37 |
| JOB31 | JOB15 | JOB31 | JOB16 |
| JOB37 | JOB52 | JOB37 | JOB54 |
| JOB16 | JOB78 | JOB16 | JOB62 |
| JOB52 | JOB45 | JOB52 | JOB35 |
| JOB54 | JOB82 | JOB54 | JOB59 |
| JOB62 | JOB130 | JOB62 | JOB15 |
| JOB35 | JOB118 | JOB35 | JOB77 |
| JOB59 | JOB89 | JOB59 | JOB82 |
| JOB50 | JOB11 | JOB50 | JOB95 |
| JOB15 | JOB95 | JOB15 | JOB45 |

| | | | |
|---|---|---|---|
| JOB77 | JOB59 | JOB77 | JOB113 |
| JOB78 | JOB30 | JOB78 | JOB128 |
| JOB82 | JOB1 | JOB82 | JOB130 |
| JOB25 | JOB41 | JOB25 | JOB118 |
| JOB95 | JOB113 | JOB95 | JOB41 |
| JOB45 | JOB54 | JOB45 | JOB12 |
| JOB113 | JOB151 | JOB113 | JOB151 |
| JOB128 | JOB62 | JOB128 | JOB156 |
| JOB130 | JOB77 | JOB130 | JOB1 |
| JOB118 | JOB128 | JOB118 | JOB89 |
| JOB55 | JOB35 | JOB55 | JOB3 |
| JOB41 | JOB16 | JOB41 | JOB143 |
| JOB12 | JOB21 | JOB12 | JOB87 |
| JOB143 | JOB37 | JOB143 | JOB155 |
| JOB72 | JOB23 | JOB72 | JOB72 |
| JOB151 | JOB31 | JOB151 | JOB11 |
| JOB87 | JOB9 | JOB87 | JOB78 |
| JOB155 | JOB3 | JOB155 | JOB52 |
| JOB156 | JOB13 | JOB156 | JOB25 |
| JOB1 | JOB2 | JOB1 | JOB50 |
| JOB89 | JOB8 | JOB89 | JOB55 |

**Table 12: Jobs in their scheduling order as per each of the policy**

Table 15 shows the overall resultant schedule for each of the JOB which was created by each policy.

| JOB IDs | LSTF | FCFS | LRTF | Least Nodes First |
|---|---|---|---|---|
| JOB2 | 14-40 | 14-40 | 14-40 | 14-40 |
| JOB3 | 18-36 | 20-38 | 18-36 | 18-36 |
| JOB8 | 10-44 | 10-44 | 10-44 | 10-44 |
| JOB11 | 31-39 | 31-39 | 31-39 | 31-39 |
| JOB13 | 0-19 | 0-19 | 0-19 | 18-37 |
| JOB9 | 0-18 | 0-18 | 0-18 | 0-18 |
| JOB21 | 30-45 | 30-45 | 30-45 | 30-45 |
| JOB23 | 2-18 | 2-18 | 18-34 | 2-18 |
| JOB30 | 9-18 | 9-18 | 9-18 | 8-17 |
| JOB31 | 1-17 | 1-17 | 1-17 | 1-17 |
| JOB37 | 2-17 | 2-17 | 18-33 | 31-46 |
| JOB16 | 2-16 | 2-16 | 2-16 | 32-46 |
| JOB52 | 8-14 | 8-14 | 0-6 | 8-14 |
| JOB54 | 0-11 | 0-11 | 4-15 | 32-43 |
| JOB62 | 0-11 | 0-11 | 14-25 | 33-44 |
| JOB35 | 21-34 | 21-34 | 21-34 | 21-34 |
| JOB59 | 2-11 | 2-11 | 36-45 | 6-15 |
| JOB50 | 41-45 | 32-36 | 20-24 | 1-5 |

| | | | | |
|---|---|---|---|---|
| JOB15 | 39-45 | 39-45 | 1-7 | 22-28 |
| JOB77 | 18-30 | 18-30 | 18-30 | 18-30 |
| JOB78 | 23-30 | 23-30 | 23-30 | 23-30 |
| JOB82 | 39-46 | 23-30 | 39-46 | 7-14 |
| JOB25 | 2-8 | 2-8 | 0-6 | 0-6 |
| JOB95 | 0-9 | 0-9 | 0-9 | 36-45 |
| JOB45 | 2-9 | 2-9 | 27-34 | 6-13 |
| JOB113 | 18-29 | 18-29 | 18-29 | 2-13 |
| JOB128 | 21-34 | 21-34 | 21-34 | 21-34 |
| JOB130 | 25-32 | 25-32 | 25-32 | 25-32 |
| JOB118 | 0-7 | 0-7 | 10-17 | 4-11 |
| JOB55 | 26-29 | 26-29 | 18-21 | 26-29 |
| JOB41 | 18-29 | 18-29 | 18-29 | 18-29 |
| JOB12 | 40-46 | 40-46 | 1-7 | 24-30 |
| JOB143 | 2-7 | 2-7 | 2-7 | 2-7 |
| JOB72 | 28-30 | 28-30 | 23-25 | 6-8 |
| JOB151 | 18-30 | 18-30 | 18-30 | 0-12 |
| JOB87 | 25-29 | 25-29 | 41-45 | 25-29 |
| JOB155 | 43-45 | 32-34 | 24-26 | 7-9 |
| JOB156 | 2-7 | 2-7 | 2-7 | 24-29 |
| JOB1 | 22-33 | 22-33 | 22-33 | 21-32 |
| JOB160 | 22-40 | 22-40 | 22-40 | 21-39 |

Table 13: This table shows the time slots in which each job was scheduled by the respective policies for the workload 2

The four figures present below show the spread of energy consumption throughout the span of 48-time slots, along with available green energy. Each figure belongs to one of the four ordering policies. The blue vertical bars show energy per watt consumed in a given time slot. The light blue line shows the trend of the available green energy (standardized form).

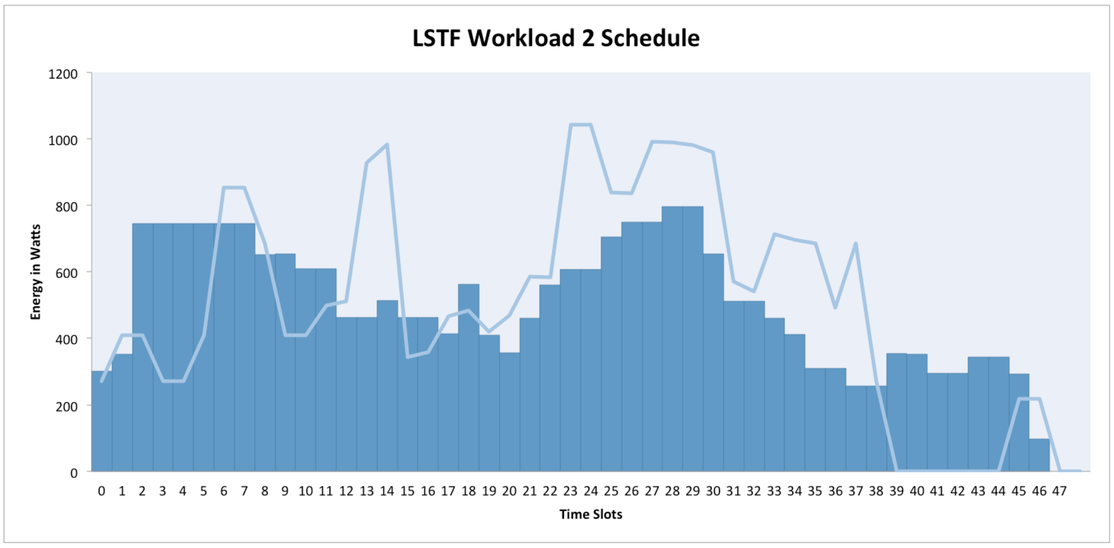

Figure 13: LSTF energy spread as per its schedule for workload 2

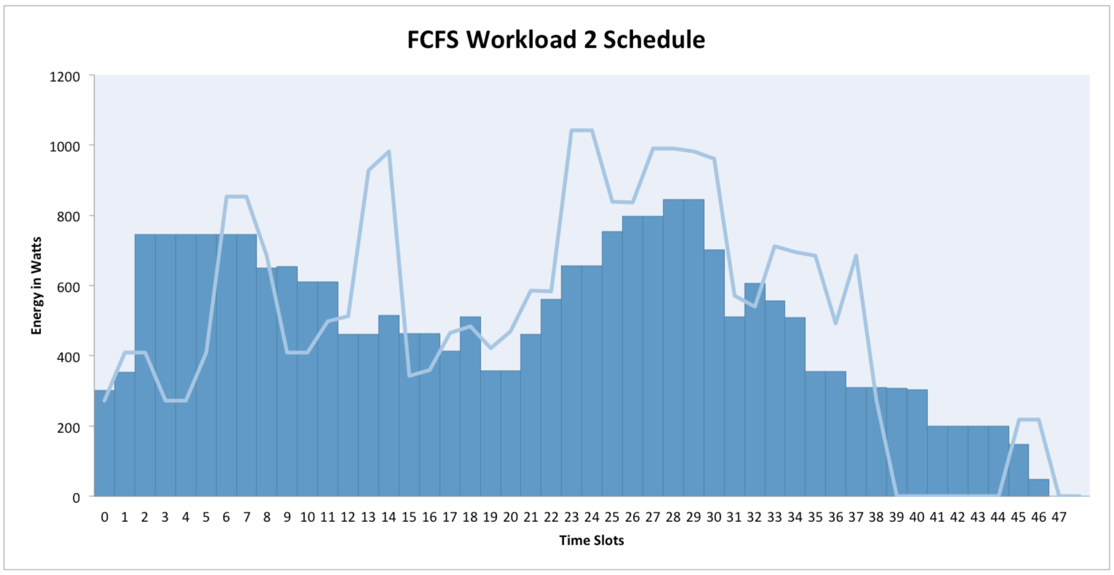

**Figure 14: FCFS energy spread as per its schedule for workload 2**

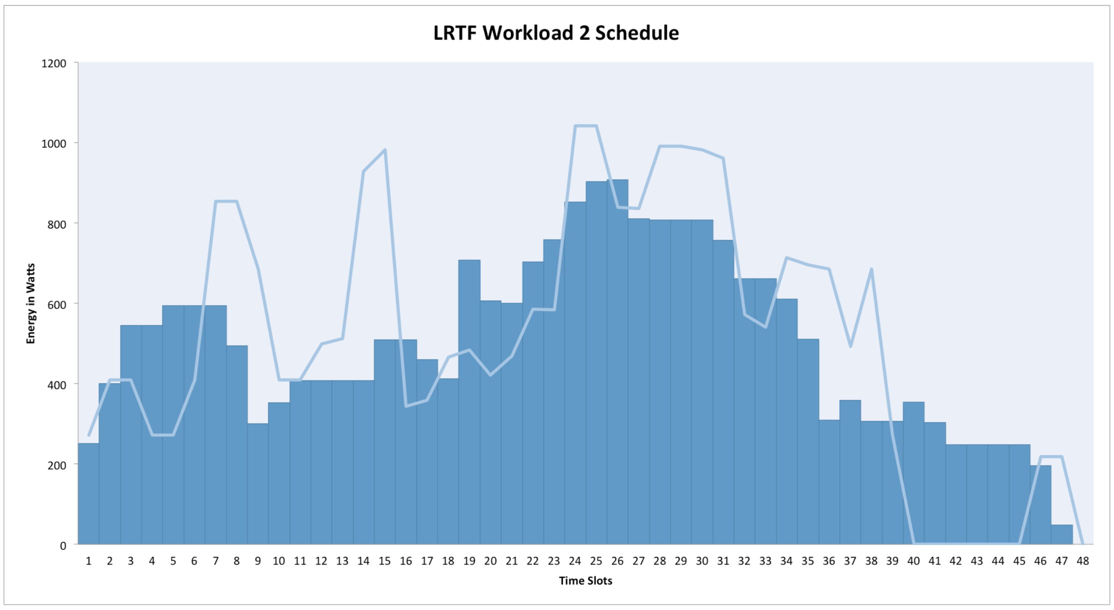

**Figure 15: LRTF energy spread as per its schedule for workload 2**

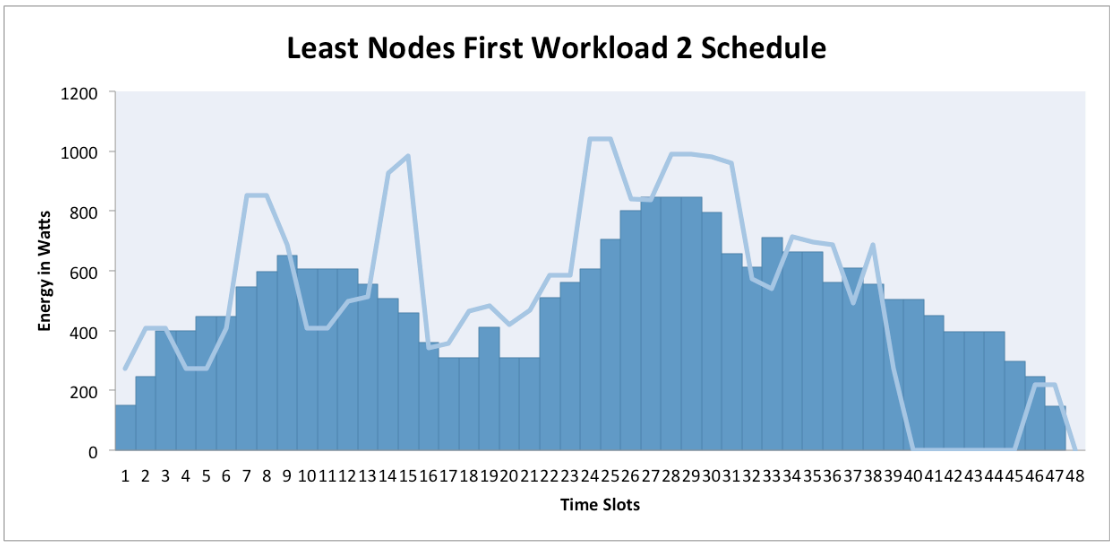

Figure 16: Least Nodes First energy spread as per its schedule for workload 2

One of the most important observations that could be made from the above figures is that the algorithm tries the spread the jobs in such a manner that it will be able to spread the energy consumption throughout the scheduling window to utilize maximum green energy as possible.

The tables below show the green energy consumption, brown energy consumption and total overall cost of the schedule, with respect to each of the ordering policies for workload 2.

| Ordering Policy | Total Green Energy Used (in Watts) |
|---|---|
| Least Slack Time First | 19393.89 |
| First Come First Server | 19871.21 |
| Least Running Time First | 19857.06 |
| Least Nodes First | 20180.34 |

Table 14: Total green energy used by each policy for workload 2

| Ordering Policy | Total Brown Energy Used (in Watts) |
|---|---|
| Least Slack Time First | 4565.60 |
| First Come First Server | 3900.84 |
| Least Running Time First | 3980.56 |
| Least Nodes First | 3941.48 |

Table 15: Total green energy used by each policy for workload 2

| Ordering Policy | Total Cost of the Overall Schedule (in $; rounded off up to two places) |
|---|---|
| Least Slack Time First | 4738.11 |
| First Come First Server | 4160.93 |
| Least Running Time First | 4071.16 |
| Least Nodes First | 3635.98 |

Table 16: Total cost of the overall schedule as per each policy for workload 2

From the information in the tables, we can see that the maximum amount of green energy was utilized by least nodes first policy. We can also observe that there is not much disparity amongst the other policies as well when it comes to usage of green energy. However, the minimum amount of green energy was used by LSTF. From Table 15, we can observe that maximum amount of brown energy was used by LSTF and minimum was used by FCFS policy. However, the difference between the brown energy usage between LRTF, FCFS and Least nodes first is not much. From Table 16, we can observe that the minimum amount of cost incurred was with least nodes first policy and the most expensive scheduling was done by LSTF.

One important thing to notice in the above information trend is that even though the brown energy usage by FCFS was lesser than LRTF and least nodes first policy, then also the cost of scheduling the jobs as per FCFS is higher than LRTF and least nodes first. This is because, in LRTF and least nodes first policy, many jobs were scheduled in the later half of the day that is when the cost of electricity is cheaper. Whereas in FCFS, these jobs were scheduled in the first half of the day which comes under on peak electricity timings. This can be clearly observed in the figures as well.

In FCFS and LSTF, the schedule prepared was up till 46-time slots, that is, all the jobs were scheduled and executed in these time slots. Where as in LRTF and least nodes first, schedule extends to 47-time slots.

From the above observations, it can be concluded that if the priority of the user is to maximize the usage of green energy, then the user should go for least nodes first as per the results in workload 2. We could also see that, the cost incurred by the schedule prepared by the least nodes first policy is the cheapest. However, if the user’s priority is to use the minimum amount of brown energy, then FCFS should be selected. Another advantage of FCFS being that in both the cases, that is workload 1 and workload 2 it used the minimum number of slots to schedule all the jobs. Hence, it is time efficient as well.

# 7. Conclusion

In this thesis, we presented a scheduling algorithm, which maximizes the amount of green energy to schedule the jobs in a workload. The schedule which it prepares uses the minimum amount of brown energy. It schedules the jobs while preserving their deadlines. In cases when enough green energy is not present to satisfy the demand of a job and the deadline of the job is not violated, to reduce or minimize the overall cost incurred to execute this job, it tries to schedule it in the later half of the day because of cheaper brown energy sourced electricity prices. We tried this algorithm with four different scheduling policies to understand the behavior of each scheduling policy with respect to the algorithm and hence figured out the best circumstances in which each of the policies should be used.

In cases of lighter workload, we found out that LRTF policy was the most efficient from green energy and cost perspective but for someone, who has time as their priority, FCFS would be preferred. However, with our real-time workload experiment, we found that Least Nodes first gave the best results with respect to green energy usage and execution cost. But FCFS could be still preferred for time-saving scenarios.

There are some limitations to the algorithm. There might be a case, in which algorithm rejects or does not schedule some jobs, which can be brought to light if the higher number of jobs is fed into the system. We did not see this case with our experiments but we do not ignore the possibility of it and this is why it is part of our future works.

Our other future works include increasing the size of the scheduling window to more than 24 hours. In the workload 2 experiments which consisted of real-world jobs, the deadlines of all the jobs were one day long. In our future works, we plan to make these deadlines stricter.

The study that we have presented in this thesis helps us to take a step towards a sustainable system in which cloud computing can harmoniously coexist with the environment.

## References:

1. BURRINGTON, I., *The Environmental Toll of a Netflix Binge*, in *The Atlantic Daily*. 2015.
2. Brown, R., *Report to Congress on Server and Data Center Energy Efficiency: Public Law 109-431*. 2008.
3. Matteson, S., *How cloud computing will impact the on-premise data center*, in *TechRepublic*. 2013.
4. *An Inefficient Truth Executive Summary*. 2007.
5. Luomi, M., *Abu Dhabi's Alternative-Energy Initiatives: Seizing Climate-Change Opportunities.* Middle East Policy Council.
6. Mathews, J.A., *Green growth strategies—Korean initiatives.* Futures, 2012. **44**(8): p. 761-769.
7. Goiri, I., et al. *Energy-aware scheduling in virtualized datacenters*. in *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*. 2010. IEEE.
8. *Green Web Hosting Business. http://www.ecobusinesslinks.com*
9. Adnan, M.A., R. Sugihara, and R.K. Gupta. *Energy efficient geographical load balancing via dynamic deferral of workload*. in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. 2012. IEEE.
10. Adnan, M.A., et al., *Dynamic deferral of workload for capacity provisioning in data centers.* arXiv preprint arXiv:1109.3839, 2011.
11. Zhang, Y., Y. Wang, and X. Wang. *Greenware: Greening cloud-scale data centers to maximize the use of renewable energy*. in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. 2011. Springer.
12. Stewart, C. and K. 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*. 2009. IEEE.
13. Brown, M. and J. Renau, *Rerack: Power simulation for data centers with renewable energy generation.* ACM SIGMETRICS Performance Evaluation Review, 2011. **39**(3): p. 77-81.
14. Le, K., et al., *Cost-and energy-aware load distribution across data centers.* Proceedings of HotPower, 2009: p. 1-5.
15. Li, C., A. Qouneh, and T. Li, *Characterizing and analyzing renewable energy driven data centers.* ACM SIGMETRICS Performance Evaluation Review, 2011. **39**(1): p. 323-324.
16. Sharma, N., et al. *Blink: managing server clusters on intermittent power*. in *ACM SIGPLAN Notices*. 2011. ACM.
17. Yoo, A.B., M.A. Jette, and M. Grondona. *Slurm: Simple linux utility for resource management*. in *Workshop on Job Scheduling Strategies for Parallel Processing*. 2003. Springer.
18. Toosi, A.N. and R. Buyya. *A Fuzzy Logic-based Controller for Cost and Energy Efficient Load Balancing in Geo-Distributed Data Centers*. in *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*. 2015. IEEE.
19. York, K., *The Internet is the single biggest thing we're going to build as a species*. 2016: Twitter.
20. Drake, N., *Cloud computing beckons scientists.* Nature, 2014.
21. Drake, N., *How to catch a cloud.* Nature, 2015. **522**(7554).
22. *What is Energy?* ; Availa*ble from: http://www.conserve-energy-future.com.*
23. Panwar, N., S. Kaushik, and S. Kothari, *Role of renewable energy sources in environmental protection: a review.* Renewable and Sustainable Energy Reviews, 2011. **15**(3): p. 1513-1524.
24. contributors, W., *Renewable energy*, in *Wikipedia, The Free Encyclopedia*. 2017, Wikipedia, The Free Encyclopedia.
25. Rozenblat, L., *YOUR GUIDE TO RENEWABLE ENERGY* 2017.
26. Ltd., G.R.E., *Global Map of Incentives for Renewable Energy*. 2013.
27. Lewis, J.I., *The evolving role of carbon finance in promoting renewable energy development in China.* Energy Policy, 2010. **38**(6): p. 2875-2886.
28. Frondel, M., et al., *Economic impacts from the promotion of renewable energy technologies: The German experience.* Energy Policy, 2010. **38**(8): p. 4048-4056.
29. Shaikh, P.H., et al., *Building energy for sustainable development in Malaysia: A review.* Renewable and Sustainable Energy Reviews, 2017. **75**: p. 1392-1403.
30. *DSIRE: Database of State Incentives for Renewables & Efficiency*.
31. *Green Web Hosting Companies. https://www.sustainablebusinesstoolkit.com/best-green-web-hosting/*
32. *Power & Energy Technology Report. 2011. https://technology.ihs.com/Research-by-Market/450473/power-energy-technology*
33. *A UK's government initiative - Carbon Reduction Commitment.*
34. Goiri, Í., et al. *Greenslot: scheduling energy consumption in green datacenters*. in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 2011. ACM.
35. Toosi, A.N., et al., *Renewable-aware geographical load balancing of web applications for sustainable data centers.* Journal of Network and Computer Applications, 2017. **83**: p. 155-168.
36. *What is flexible pricing? https://www.agl.com.au/residential/help-and-support/flexible-pricing*
37. *Energy & Gas Prices. https://www.energyaustralia.com.au/home/electricity-and-gas/plans*

38. *Minitab. http://www.minitab.com/en-us/*