UNIVERSITY OF CALIFORNIA
RIVERSIDE

Peak Efficiency Aware Request Scheduling for Data Center Server

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Electrical Engineering

by

Yukun Chen

June 2017

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, with the increasing growth of Internet service, especially with the fast growth of 4G network and mobile device, data center plays more and more critical role in the future. In addition, more and more devices and apps are based on cloud service instead of local machines, which will cause a huge burden for data centers. Due to massive service requests, especially in cloud computing, data center will consume huge power and generate huge cost. According to Data Center Knowledge, Googles newest data center at The Dalles in Oregon, a 164,000 square foot building that opened in 2016, brought its total investment in that site to \$1.2 billion [7]. In another example,the worlds top cloud providers made \$11.2 billion in revenue, selling raw compute power and storage as virtual services in 2016 [8]; this consumption will increase heavily in the future. According to Data Center Knowledge, it forecasts collectively a select group of the largest massive-scale Infrastructure-as-a-Service clouds will reach more than \$120 billion in revenue in 2020, growing exponentially within 5 years[8]. The cost of power and its associated delivery and cooling are becoming significant factors in the total expenditures of large-scale data centers[9]. Thus, it is necessary for us to save energy for data centers, because data centers will have a massive impact on power network, because of huge power consumption.

According to table 1.1, we can see that in the data center the CPU will consume the most part of power. Other parts like memory or disks will not have a huge impact on the power consumption. According to figure 1.1, we can see in data centers, the processor will consume the most of power. When we combine these results, we can see the server's CPU will play the major role in IT. Thus, in this paper, we will focus on data center's server level. We will design an new algorithm to reallocate the incoming job request. The goal for this algorithm is to lower the average power for each server.

Component	Peak Power (Watts)
CPU	80
Memory	36
Disks	12
Peripheral slots	50
Motherboard	25
Fan	10
PSU losses	38
Total	251

Table 1.1: Component Peak Power Consumption for a Typical Server [6]

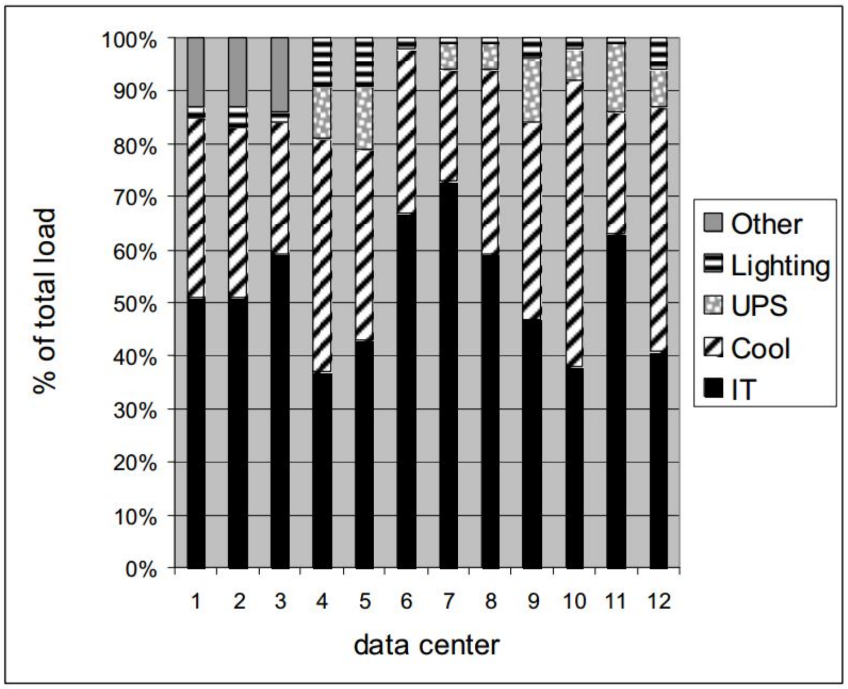


Figure 1.1: Data Center Benchmarking [1, 2]

1.1 Problem Statement

In this paper, we focus on server-level energy efficiency for data centers. Each server contains multiple CPUs to deal with incoming requests. In reality, peak efficiency for each CPU is not at 100% utilization [3]. Servers with different Energy Proportionality (EP) models have different peak efficiency utilization. Energy Proportionality is based on this equation:

$$EP = 1 - \frac{Area_{actual} - Area_{ideal}}{Area_{ideal}} \tag{1.1}$$

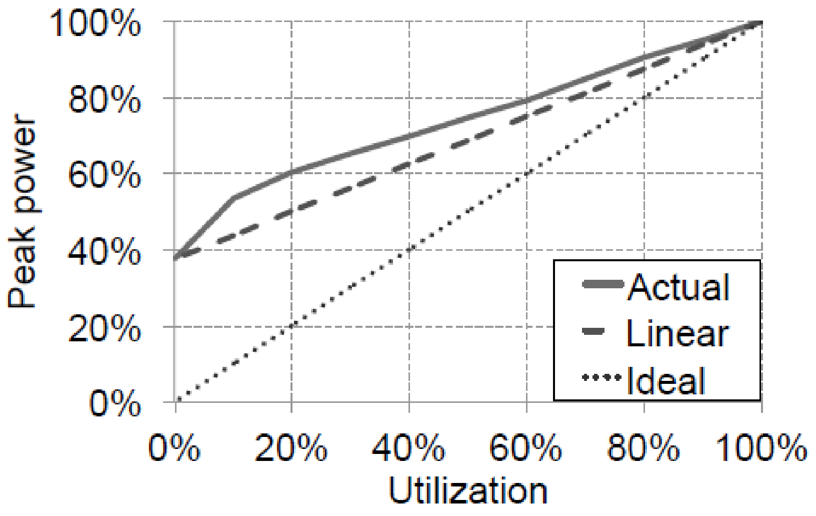


Figure 1.2: Energy proportionality curves used for calculating dynamic range (DR), linear deviation (LD), and energy proportionality (EP) metrics [3]

According to equation 1.1 and figure 1.2, we can calculate the energy proportionality. An ideal energy proportional server would have EP = 1, which means the relation between peak power and CPU utilization is linear and the start point is the origin. However, in real servers, when the server's status is idle, it will still consume power due to factors such as power supply inefficiencies, motherboard components, and leakage energy. Thus, we can get the linear line in figure 1.1. However, the actual power curve line is not the linear, but it is a non-linear curve. Thus, it has a peak value. When the utilization is at this value, we can get the peak power efficiency. In the simulation, we have a total of five energy proportionality models: Low, Mid, High, Super, Mixed. In this paper, we assume the value of low model is 0.24, the value of mid model is 0.73, the value of high model is 1, and the value of super model is 1.2. When we got the energy proportionality, we can know the relation about CPU's utilization and power. Thus, we can get the power efficiency utilization for these four models. In the simulator, different EP model will have different power distribution, which will cause a different peak efficiency utilization in the end. In this simulation, peak efficiency utilization of High model is about 60%. For Super Model, this value is about 50%. For Low and Mid, the peak efficiency value is at 100%.

1.2 Objectives

According to the table 1.1, CPU will play a critical role in data center's power consumption. In addition, different EP model will have it's own peak efficiency utilization. The goal of this work is to modify request scheduling by delaying incoming request in order to operate at peak efficiency utilization as much as possible in order to acquire high power efficiency. Thus, in this paper, we will design a new algorithm to queue up requests until we have enough requests to run at Peak efficiency utilization, which will modify the distribution of incoming service request. The average power cluster should decrease in the end.

Chapter 2

Background

In this chapter, we will introduce the background about data center and the BigHouse data center simulator. Our algorithm is focus on how servers allocate and schedule the incoming request.

2.1 Data center background

In figure 2.1, we can see the basic components of a data center. Data centers contains three major parts. The first part are servers, the second part are data storage components, and the third part are communication network components. The most common types of servers are of two types; mostly are high-end servers, and mid-range servers. Mid-range servers will have lower power consumption, and high-end server will have more powerful consumption. According to figure 2.2, we can see both two kinds of servers' power consumption will increase heavily in the future. The power usage of mid-range servers increased from about 0.6 kW in 2006 to 1.5 kW in 2020. The power usage of high-end servers will increase from 10 kW to 20 kW by 202. Both o them will consume more power in the future. In addition, with the fast development of model devices, more and more devices will be get access to the Internet, thus we need more data center to deal with huge data.

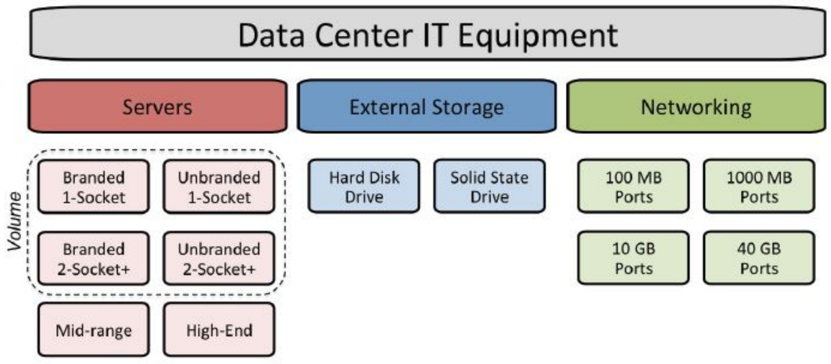


Figure 2.1: Equipment Types Modeled in Energy Estimation [4]

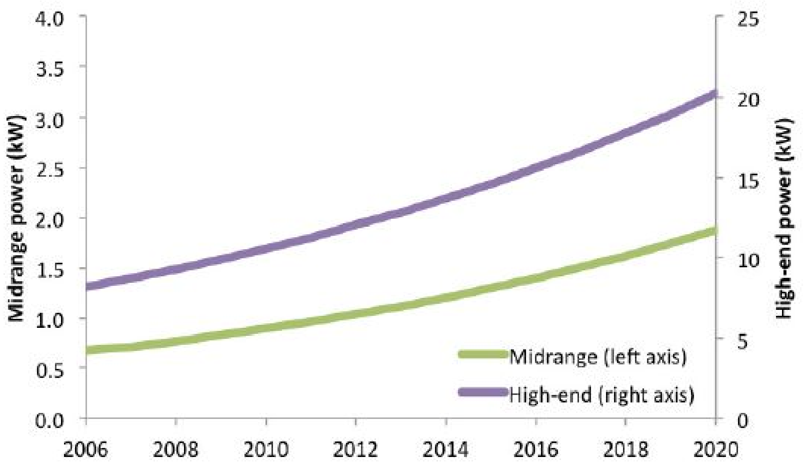


Figure 2.2: Average Power Draw Assumptions for Mid-Range and High-End Servers [4]

Data centers provide a range on information technology services like data storage and management, web services, email services, etc. Large data centers are industrial scale operations using as much electricity as a small town[10], thus data centers have huge power consumption demands. Data center contains major components such as servers, routers, and storage. In addition, data centers need cooling system to maintain the normal working condition. Due to this, many data centers are geographically located in advantageous locations.For example, the data center can be located close to a river, which can be used to easily to cool the server and it will reduce the cost of building cooling system. Besides cooling needs, data center also need stable power source, which requires Uninterruptible power supplies (UPS) and potentially connections to redundant power grids.

In this paper, we will focus on data center's power consumption. The power usage of a server cluster can be described by such equation:

$$P_{Cluster} = F(n) + V(u_t, n) + \epsilon \tag{2.1}$$

In this equation, F(n) is fixed power, which can be described by such equation:

$$F(n) = n \times (P_{idle} + (PUE - 1) \times P_{peak}) \tag{2.2}$$

This part is the fixed power consumption in data center. n means the number of servers in data center. P-idle is the power consumption when CPU is idle. PUE means that power usage effectiveness, which can be defined by:

$$PUE = \frac{\text{Total power usage}}{\text{Power usage of servers}} \tag{2.3}$$

The PUE is a measure of how much power consumed in the data center goes towards computation in the servers. Therefore, when PUE is close to 1, it means data center has excellent efficiency as the entire power consumption is at servers. If the PUE is greater than

1, it means that data center will consume other part of energy greatly for cooling and power infrastructures. For example, when PUE is quite high, data center may consume a lot of power on cooling system. Thus, temperature plays a critical role in PUE. The percentage of cooling varies but can be up to 50% (or even higher in poorly designed or operated data centers)[11, 1]. It recommends that the most critical data centers be maintained between 20 and 25℃, with an allowable range of 15 to 32℃[1]. The major part of server is the CPU. CPU will have a good working condition within this range of temperature. If the data center is outside this range, it should consume extra energy to maintain servers working within such range of temperature. Thus , we can see a lot of data center are located near the river.

The second part, $V(u_t, n)$ is variable power, which can be described by this equation:

$$V(u_t, n) = n \times (P_{peak} - P_{idle}) \times (2u_t - u_t^r) \tag{2.4}$$

n in this equation is the number of servers. P_{peak} means the power consumption when fully utilized. u_t is the average CPU utilization which is from 0 to 1. We can assume the u_t^r as a constant which is from 0 to 1. ϵ is a constant in this equation, which stands for correction constant. Thus, we can model the electric load in terms of n and u_t. When increasing n and u_t, both the consumption power and electric power consumption will increase. For a single server, we can just assume the n is equal to 1 in equation 2-4.

2.2 BigHouse

Big House is a simulation infrastructure for data center systems[12], which is based on the stochastic queuing simulation (SQS) methodology[5]. In general, we can simply regard this simulator as a three level simulator. Each data center contains multiple servers, and each servers contain a lot of CPUs. When the data center got incoming requests, data center will allocate these requests to each server based on different allocating models. The simulator supports two request scheduling models: Uniform and Peak. Uniform model will allocate incoming request equally to each server [13]. Peak model's allocation strategy aims to utilize the existing server at the peak energy efficiency point [3]. In Peak scheduling, requests are packed to a subset of servers enabling idle servers to sleep. If the existing server can not deal with the whole requests, we will wake up more servers to finish incoming jobs. Thus, some servers will be fully utilized and some servers' status will be idle.

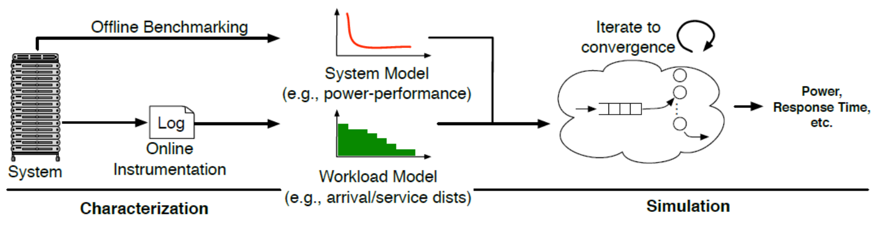

Offline Benchmarking
System
Log
Online Instrumentation
System Model (e.g., power-performance)
Workload Model (e.g., arrival/service dists)
Iterate to convergence
Power, Response Time, etc.
Characterization
Simulation

Figure 2.3: Overview of the BigHouse infrastructure flow. A system is (a) instrumented to derive workload inter-arrival and service time distributions and (b) characterized to create a model of system behavior (e.g., power-performance settings). From these inputs, simulations derive estimates for new system designs and/or configurations [5]

In Uniform model, each server will acquire incoming jobs equally, which means that each server will consume the same energy. Here, we assume that each server has the same number of CPU, and all CPUs have the same performance, which can ensure that each server will have the same energy performance in Uniform model. In Peak model, each server will have different power performance. Some servers will have high average power, because they deal with more jobs. Other servers will have less power because of less incoming jobs.

Figure 2.3 shows the overview of the BigHouse infrastructure flow. The simulator is driven by the workload's arrival time distribution and service time distribution. The simulator samples from the arrival time distribution to determine when to inject the next request into the simulator. Once a request reaches the CPU, the simulator samples from the service time distribution in order to determine the service time of that request. To determine the power consumption of the server, the server generates a model of system power profile. In this simulation, we have Low, Mid, High, Super and Mixed power profiles. Different load policy will cause the different system load behavior. The simulator samples the utilization of the server periodically and uses the power model to determine the power of the server.

When the simulation start, the simulator will consistently sample various statistics, such as the server power, server utilization, and request response time, until the statistics has convergence to a confidence interval of 95%. In figure 2.4, we can see that how the simulator sampling mechanism works. There are three major phases. The first step is calibration, the second step is measurement and the third step is convergence. In this simulator, it will finish the simulation when the convergence is at 95%.

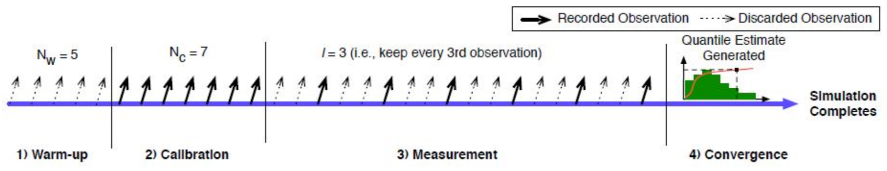


Figure 2.4: The Sequence of Phases in a BigHouse Simulation: At first, all observations are discarded during warm-up, avoiding coldstart bias. Next, during a brief calibration phase, a small sample is collected to determine the appropriate lag spacing and histogram configuration. The majority of the simulation is spent in the measurement phase, where observations are taken with sufficient spacing to ensure independence. Finally, when the desired statistical confidence is achieved, the simulation terminates, outputting quantile and mean estimates.[5]

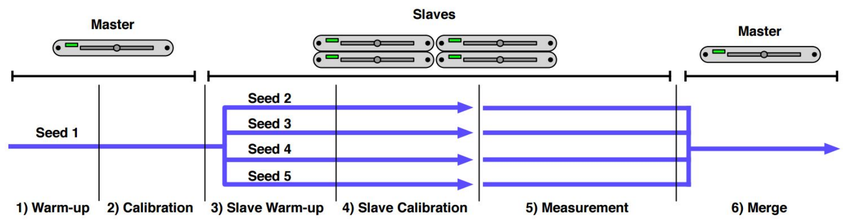


Figure 2.5: Parallel Execution on a Cluster [5]

This simulator can also run in a distributed system to speed up simulation, which is a master-slave model. The master machine will allocate task to slave machine, and slave machine will compute the incoming data and send data which has been done back to master machine to finish merging. Each slave machine will work independently and they can't communicate with each other or exchange data with each other. In this paper, we will just run this simulator in local machine, instead of disturbed system. In addition, in this simulator, there are a lot of work loads from a variety of sources. In this paper, we will choose two main workloads, DNS and WWW. In addition, we will assume the data center will contains one or five servers in most case. We will focus on the data center servers' average power consumption and it's latency. At the same time, we need to pay attention to the histogram of instant utilization.

2.3 Energy Proportionality

Energy proportionality is a measure of the relationship between power consumed in a computer system, and the rate at which useful work is done [14]. Energy proportionality (EP) is defined as:

$$EP = 1 - \frac{A_{actual} - A_{ideal}}{A_{ideal}} \tag{2.5}$$

where A_{actual} and A_{ideal} is the area under the server's actual and ideal energy proportionality curve, respectively as shown in figure 2.6. An ideal energy proportional server would have EP = 1. Values less than 1 indicate a server with poor energy proportionality that consumes significant power at low utilization. Values over 1 indicate a server with better energy efficiency than that of an ideal energy proportional server.

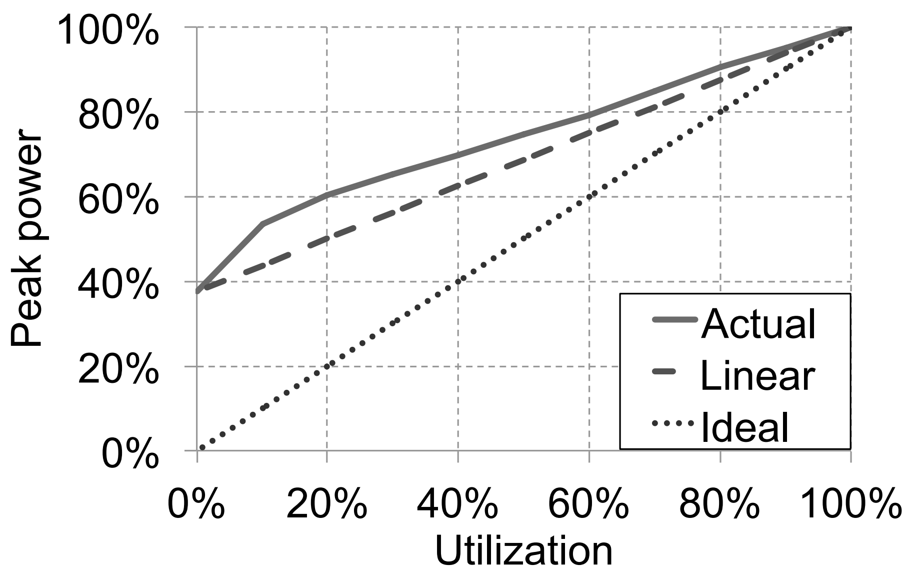


Figure 2.6: Energy proportionality curves. [3]

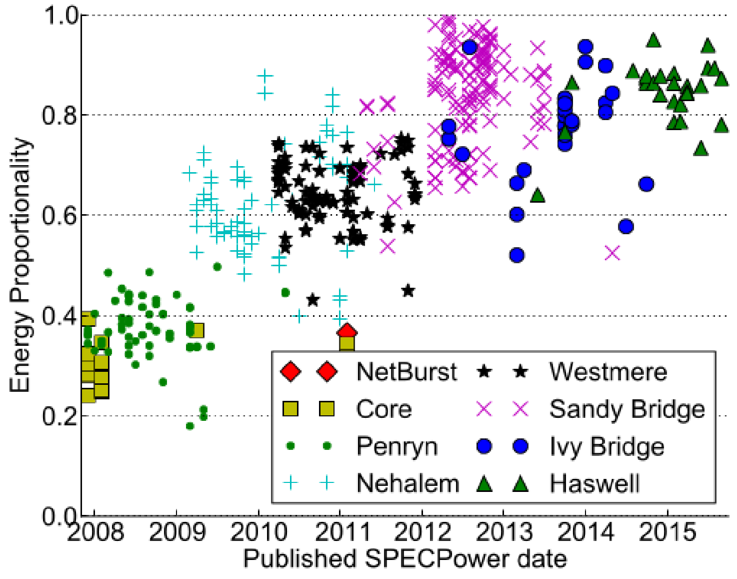


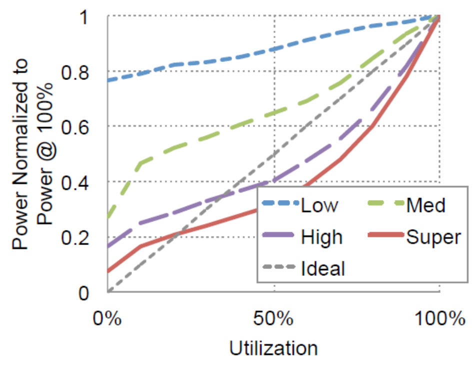
Figure 2.7: Energy proportionality experienced significant improvement between 2008 and 2012. Since then energy proportionality has leveled off with near-ideal servers becoming the norm. [3]

In the figure 2.7, we can see the improvement of energy proportionality from 2008 to 2017. As we can see, the energy proportionality has made a grate improvement. From 2008 to 2012, energy proportionality improved drastically from an average of around 0.3 to around 0.8. Since 2012, the observed best energy proportionality has topped out at 1.0, with the range and average energy proportionality of new servers approaching ideal [3]. A major contributor to the improvement of energy proportionality is processor power management such as DVFS [15, 16, 17, 18, 19], sleep states [20] and power gating [21, 22]. Dynamic voltage and frequency scaling (DVFS) is the adjustment of power and speed settings on a computing devices various processors, controller chips and peripheral devices to optimize resource allotment for tasks and maximize power saving when those resources are not needed[23]. When we implement the DVFS, we can allow devices, like data center, to perform more energy efficiently depend on different kinds of tasks. When the data center deal with the light task, data center can execute will low-power model which will lower the power consumption.

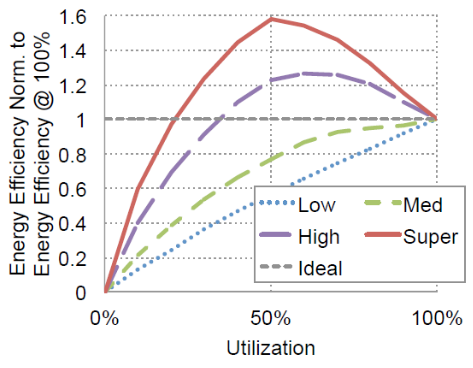
Chapter 3

Peak Efficiency Request Scheduling for Servers in Data Centers

In this chapter, we will introduce a new policy for request scheduling for server. According to figure 3.1, in real data center servers, we can't assume the relation between CPU utilization and energy efficiency as linear. In reality, the relation between energy efficiency and CPU utilization is a non-linear curve, which has a peak value at non-peak utilization. This is shown in figure 3.1. As can be seen from figure 3.1(a), as energy proportionality improved over the past decade, the power curve became increasingly non-linear. A side effect of this is that the energy efficiency curve also changed drastically. The energy efficiency of these servers are shown in figure 3.1(b). Historically, the peak energy efficiency occured at peak utilization (100% utilization). With high energy proportioanl servers, we now see that the peak energy efficiency point occurs at around 60% utlization. With super energy proportional servers, the peak efficiency point moves to around 50%



(a) Energy Proportionality Curves



(b) Energy Efficiency Curves

Figure 3.1: Energy efficiency properties of servers with different models [3]

utilization. In this work, we aim to exploit this property by trying to execute at this peak efficiency point to increase energy efficiency.

If more and more work load execute in peak efficiency utilization, we can increase the energy efficiency. CPU's utilization will depend on computer's usage of processing resources or the work load for the CPU, which is from 0 to 1. Different work load will have different utilization. Because the relation between CPU's utilization and energy efficiency is a curve, we need to design an algorithm to let more and more job request work in peak efficiency utilization. However, most data center workloads spend the majority of the time in low utilization regions. In order to increase the time operating in the peak utilization region, we propose to explore queuing up requests, and then executing in batches to get to the peak efficiency utilization point.

This algorithm shows how to let more and more job request work in the peak efficiency utilization. Firstly, we need to set up a queue with fixed length to hold incoming job request. Then, we need to check whether the queue is full or not, if the queue is full, we need to free some part of job which has been saved in the queue. We can not free all the job service because we can't make sure whether the instant utilization is close to the peak efficiency utilization. If it is not, we will let too many job request work on bad condition which will consume the extra energy. Thus, we just need to free a little part of job service saved in the queue. After checking whether the queue is full or not, we will check whether the instant utilization is equal to peak efficiency or not. If the instant utilization is equal to the peak efficiency utilization, there is no need to pull this job into queue, and we can execute it directly. If the instant utilization is not equal to peak efficiency utilization, we can add this job request into queue, and then check whether the queue is full or not. If the queue if full, we need to free a part of job request in the queue. In real simulation, we assume that the peak efficiency has a deviation which is about 5%. Within this deviation, we can regard this utilization is equal to peak efficiency utilization.

Algorithm 1 Peak efficiency scheduling

```
initialization;
Set up queue with fixed length;
if queue is full then
    while queue is full do
        free 10% job in the queue
    end
end
if instant utilization not equal to peak efficiency utilization then
    add job into queue;
    if queue is full then
        free 10% job in the queue
    end
else
    execute job service directly and free the data in queue
end
```

When implement this algorithm, we can let the most job request work in the peak efficiency utilization. However, we can't let all incoming job work in the peak efficiency utilization.The reason is that the length of queue is limited. When the queue is full, we have to free the queue before adding new job request. That the reason why there are still some job requirement not working in peak efficiency utilization. In addition, in this simulation, each server contains 72 cores, thus the total number of jobs each server can deal with is limited. When we check out the core in server are full, we also need to add the incoming job requests in the queue. If we still execute the incoming job request without checking whether the core in CPU is full or not, sometimes the simulation will not be executed successfully. Because when the instant utilization is at peak efficiency utilization, we will free all the data which has been stored in the queue, we don't need to worry about there are a lot of remaining data in the queue after finishing simulation.

Chapter 4

Evaluation

In this chapter, we will show the simulation result. Our simulation is based on BigHouse simulator. This simulation is run on a local machine. We will assume each server has the same CPU. In the same time, in this simulation, we just focus on server's energy consumption, and don't care about the other energy consumption in data center. In this simulation, we choose two cluster scheduling models, the first one is uniform, and the second one is peak. The uniform model will allocate incoming job equally to every server in data center. Because we assume each server's CPU is the same, thus each server should deal with the same amount job requests. The peak model is different from the uniform model, which will fully utilize one server firstly, ans then will allocate a new server. For this simulation, we will firstly assume the data center will have only one server, thus there is no difference between the peak model and uniform model. Difference kinds of job request will have different kind of peak efficiency utilization. In this simulation, we choose this work load which is named csedns, which is derived from a departmental DNS server.

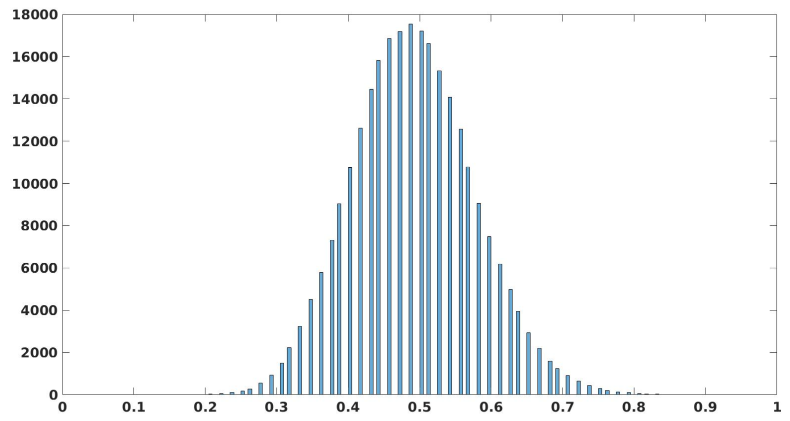


Figure 4.1: Original histogram for csedns with Super EP for 5 servers

Figure 4.1 shows the original histogram for csedns work load. In this figure , the value of ρ is 0.5. The super model's peak efficiency utilization is also 0.5. The shape of histogram is like the normal distribution. The peak point is at 0.5 which is equal to the value of ρ, which means when the utilization is reach at 0.5, it starts draining. Figure 4.2, we can see the value of ρ is 0.2. The workload is www, which has the same distribution. The difference is that when utilization is at 0.2, it will start draining. Figure 4.3 and 4.4 is the simulation result which implementing the algorithm. The result's histogram distribution are different. For figure 4.3, the value of ρ is 0.5 and the peak efficiency utilization is also 0.5. Thus, when the ρ is at 0.5, it will start draining. Thus, we can see that all the job request will work around 0.5. On the other hand, we can see that the shape of figure is different. In figure 4.3, the value of ρ is 0.2, which means that when the value of ρ at 0.2, it will start draining, and it will cost some extra time to reach the peak efficiency utilization point. That's the reason why the shape of simulation result is different. Thus, different value of ρ and the peak efficiency utilization, both of them will have a impact on the final

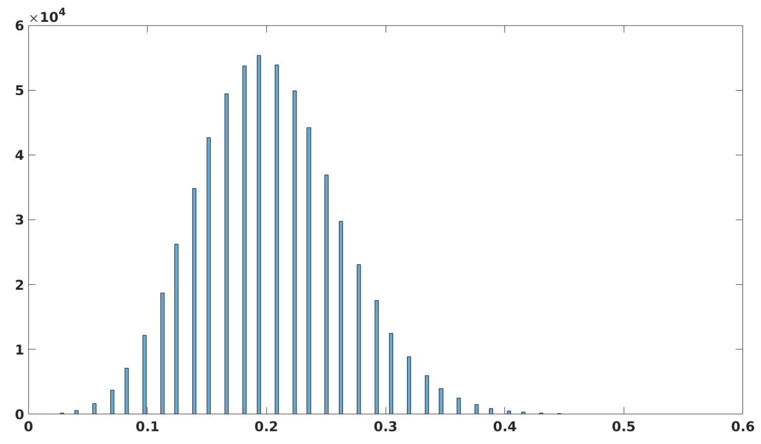


Figure 4.2: Original histogram for www with High EP for one servers

distribution of histogram. The value of ρ and the peak efficiency utilization will have a impact on the latency for the simulation result. We will post some results at the end of this chapter. Thus, there is no difference between uniform and peak cluster management

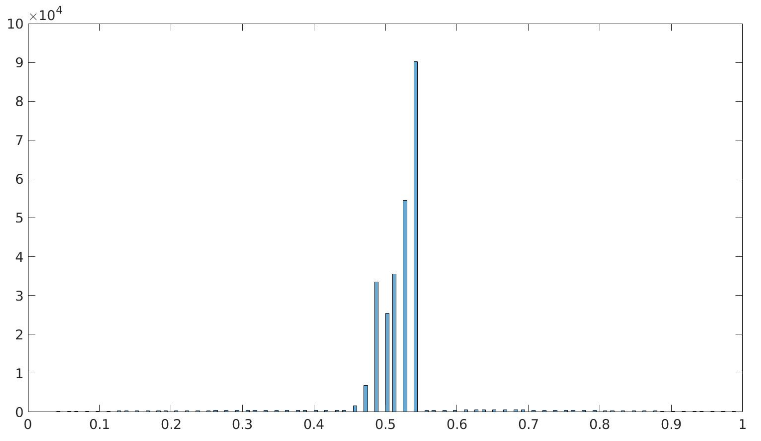


Figure 4.3: Histogram for csedns with Super EP after implementing Algorithm 1 for 5 servers

strategies. If we choose the uniform, every server will have the same histogram distribution

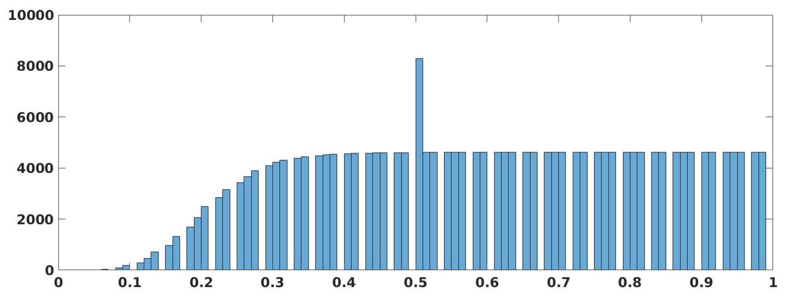


Figure 4.4: Histogram for www with Super EP after implementing Algorithm 1 for one server

and each server will take job request equally. However, when we choose the peak model, each may have different histogram distribution and each server may take different amount of job requests. Some servers may take huge amount of work load, and some of servers' status may be idle.

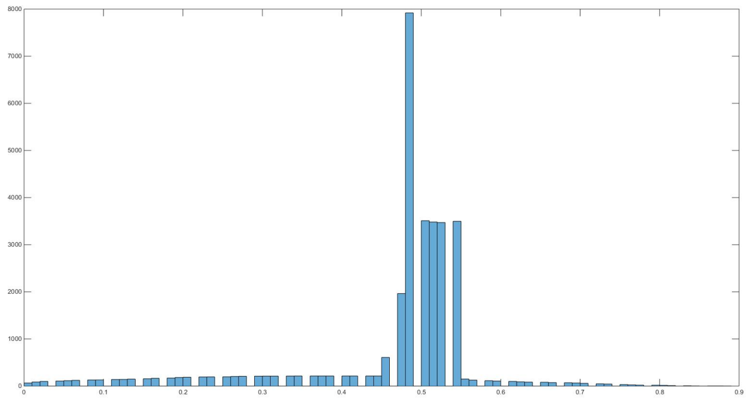


Figure 4.5: Histogram for csedns with Super EP with peak model within 10 servers its' 2ed server

Thus, next part we will show the simulation result for multiple servers in data center. We will shows the result of peak model. In this simulation, data center contains

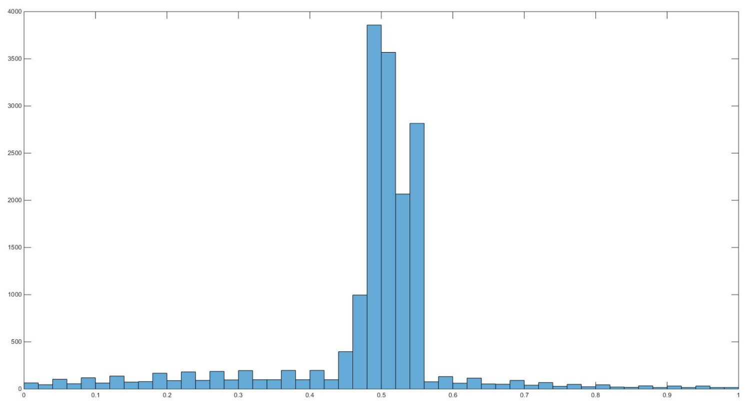


Figure 4.6: Histogram for csedns with Super EP with peak model within 10 servers its' 4th server

totally ten servers, and we will run the simulation at Super model, which the peak efficiency utilization is about 0.5 not 0.6. Figure 4.3 and 4.4 shows the one server's histogram utilization. Because of implementing the previous algorithm, the most job have been executed around the peak efficiency utilization, which is 0.5 in this simulation. It is quite clear that these two servers have different work load distribution and the number of job each server deal with is also different. As long as see the maximum number for these two histograms, we can see that figure 4.3, the maximum number of job services is close to 10000 and figure 4.4, the maximum number is just close to 4000. Thus, we can prove that in peak model, different servers will take different number of job requests. In addition, for uniform model, all the servers will have the same histogram distribution, which is almost the same as the the figure of one server. From these several figures, we can observe that more job services have run within peak efficiency utilization. However, it is not quite straightforward to prove that this algorithm will save much energy for data center. Because we need to see how much energy we will save by implementing this algorithm, or the increment about power efficiency.

we can describe the power efficiency by:

$$\text{Power Efficiency} = \frac{\rho}{\text{Average power}} \tag{4.1}$$

In this equation, ρ is the target Rho during the simulation. We can image ρ as a fixed number for each simulation. When the average power decrease, the power efficiency will increase.

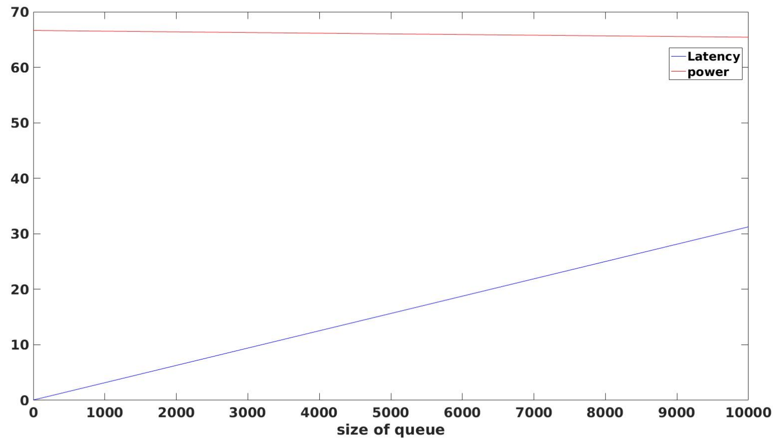


Figure 4.7: the relation between size of queue and latency or power

According to this figure, we can see when implementing this algorithm, the average power cluster will decrease. The simulation will also be completed when converging at 95%. When the average cluster power decrease , we can assume server will save some part of energy. In table 4.1, we can see the simulation result, which is based on dns benchmark. After implementing algorithm, the average power will decrease, which means that this server will execute the incoming job with low energy consumption. However, this method will also cause the longer latency, which is a negative part of this algorithm. Long latency means

ρ	Previous power	After	Power Decrement
0.1	49.938	48.8096	2.26%
0.2	59.1576	58.13737	1.72%
0.3	66.9764	65.7524	1.83%
0.4	74.987	73.574034	1.88%
0.5	84.691	83.0810	1.9%
0.6	97.800	95.8334	2.01%

Table 4.1: csedns average power (one server, Ep: High)

that the data center will spend more time dealing with the incoming request, which is not quite good for the job request which need low latency and quick response.

In table 4.3 and 4.4, we describe the latency by response mean time. As we can see, for both two work loads, the response mean time increased heavily. The major reason is that we set up a large size of queue to save the job request which is not at the peak efficiency utilization. Because of the size is very large, and the queue is very easy to be fulled up, which will cost long time to free the data which has been saved in the queue. If we reduce the size of queue, we can actually lower the latency but we can't reduce too much power, because a lot of job requests are far away from the peak efficiency utilization. If the size of queue is too small, there are still remaining a large part of job requests will work far away from the peak efficiency utilization, thus, we can not reduce much power. On the other hand, the value of ρ will also have an impact on the latency. Because we need

ρ	Previous power	After	Power Decrement
0.1	49.6248	42.979	13.4%
0.2	58.87754	50.2773	14.61%
0.3	66.629	56.523	15.2%
0.4	74.631	62.68	16.01%
0.5	84.3864	69.261	17.92%
0.6	97.16125	77.4306	20.31%

Table 4.2: www average power (one server, Ep: High)

to move job request from ρ to peak efficiency utilization. If the value of ρ is far away from the peak efficiency utilization, it will spend much longer time to move job request to the peak point. However, when the value of ρ is close the peak efficiency utilization, we can find that the latency will decrease. The reason is that when ρ is close to the peak efficiency utilization, it will spend less time reaching the peak efficiency utilization. According to the figure 4.7, we can clear see the relation between size of queue and power or latency. When we enlarge the size of queue, we can find that the latency increase heavily and the power decrease slowly. In table 4.4, we can check out the accurate value. When we enlarge the size of queue, we can see that

Thus, when implementing this algorithm, we can't sustain the latency at the same time. If we want to lower the size of queue, it will increase the power, which will consume more energy. In this simulation, this algorithm is just based on server level. If we want

ρ	Previous response mean time	After
0.1	0.187	93.618
0.2	0.18521	131.425
0.3	0.1852	87.6782
0.4	0.18517	65.80489
0.5	0.18513	52.681
0.6	0.184658	43.933

Table 4.3: csedns latency

to keep the low latency and decrease the power, we must optimize in CPU's level. In this simulation, each server contains 72 cores. Each core has same performance. There is no optimization in this level right now. If we can optimize in both two levels, we may reduce the power and keep latency in the low level.

ρ	Previous response mean time	After
0.1	0.06482	934.412
0.2	0.06478	467.247
0.3	0.065	311.526
0.4	0.065	233.66
0.5	0.0649	186.951
0.6	0.184658	155.8102

Table 4.4: www latency

size of queue	latency	average power
0	0.06	66.6294
100	0.3597	66.609287
1000	3.1674	66.50857
5000	15.6462	66.01972
10000	31.24943	65.4827

Table 4.5: the relation between size of queue and latency or power based on www workload and ρ is 0.3 and EP is High model

Chapter 5

Conclusions

In this paper, we introduced a new algorithm to let more and more job requests worked in the peak efficiency utilization, which is not 1.0 in reality. The basic idea is that when the instant efficiency utilization is not at peak efficiency utilization, we will set up a queue to save these work loads until the instant utilization. When more job requests work in the peak efficiency utilization, the average power will decrease. This result result proves that the job request work under peak efficiency utilization will some energy. That's the reason why the average cluster power will decrease. However, this algorithm has some weak points. When we increase the size of queue, the performance of data center will decrease. The latency will be much longer than before. The major reason is that when the size of queue is large, it will cost a lot of time to free the job saved in the queue, which will cause the huge latency. Thus, there is a trade off between saving power and preventing latency when implementing this algorithm. In addition, according to the previous figure and table, we can get this conclusion: when the size of queue increase, the average power will decrease a little, but the latency will increase heavily, which means that we must put up with the heavily longer latency.

Thus, let more job requests work in peak efficiency utilization can actually reduce the power consumption. However, it is not a perfect way because of the long latency. If we want to save more power, we can dive in CPU level. In this simulation, we just assume each server contains 72 cores and each core has the same performance. If we can let cores in CPU have different performance. Some cores will have strong performance for computing and some cores are energy-saving. We will allocate tasks dynamically based on the requirement of latency. If some job requests don't care about long latency, we can allocate these kinds of job requests to low-performance CPU, which will consume less energy and cost long time. In this way, we can reduce some latency because of the large size of queue. In the same time, we can also save some parts of energy.

Bibliography

[1] M. K. Patterson, "The effect of data center temperature on energy efficiency," in *Thermal and Thermomechanical Phenomena in Electronic Systems, 2008. ITHERM 2008. 11th Intersociety Conference on*, pp. 1167–1174, IEEE, 2008.

[2] "Lawrence berkeley national labs, benchmarking: Data centers, dec 2007, http://hightech.lbl.gov/benchmarkingdc-charts.html,"

[3] D. Wong, "Peak efficiency aware scheduling for highly energy proportional servers," in *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pp. 481–492, IEEE, 2016.

[4] A. Shehabi, S. Smith, N. Horner, I. Azevedo, R. Brown, J. Koomey, E. Masanet, D. Sartor, M. Herrlin, and W. Lintner, "United states data center energy usage report," *Lawrence Berkeley National Laboratory, Berkeley, California. LBNL-1005775 Page*, vol. 4, 2016.

[5] D. Meisner and T. F. Wenisch, "Stochastic queuing simulation for data center workloads," in *Exascale Evaluation and Research Techniques Workshop*, 2010.

[6] R. Brown *et al.*, "Report to congress on server and data center energy efficiency: Public law 109-431," *Lawrence Berkeley National Laboratory*, 2008.

[7] "http://www.datacenterknowledge.com/google-data-center-faq-part-2,"

[8] "http://www.datacenterknowledge.com/archives/2016/06/07/top-cloud-providers-made-11b-on-iaas-in-2015-but-its-only-the-beginning,"

[9] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 338–347, ACM, 2010.

[10] J. Glanz, "Power, pollution and the internet," *The New York Times*, vol. 22, 2012.

[11] G. G. DATA, "Center power efficiency metrics: Pue and dcie," 2008.

[12] D. Meisner, J. Wu, and T. F. Wenisch, "Bighouse: A simulation infrastructure for data center systems," in *Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on*, pp. 35–45, IEEE, 2012.

[13] D. Wong and M. Annavaram, "Implications of high energy proportional servers on cluster-wide energy proportionality," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, 2014.

[14] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, 2007.

[15] C.-H. Hsu and S. W. Poole, "Revisiting server energy proportionality," in *Parallel Processing (ICPP), 2013 42nd International Conference on*, pp. 834–840, IEEE, 2013.

[16] S. Kanev, K. Hazelwood, G.-Y. Wei, and D. Brooks, "Tradeoffs between power management and tail latency in warehouse-scale applications," in *Workload Characterization (IISWC), 2014 IEEE International Symposium on*, pp. 31–40, IEEE, 2014.

[17] M. Huang, J. Renau, S.-M. Yoo, and J. Torrellas, "A framework for dynamic energy efficiency and temperature management," in *Proceedings of the 33rd Annual ACM/IEEE International Symposium on Microarchitecture*, MICRO 33, (New York, NY, USA), pp. 202–213, ACM, 2000.

[18] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, (Washington, DC, USA), pp. 347–358, IEEE Computer Society, 2006.

[19] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & cap: Adaptive dvfs and thread packing under power caps," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, (New York, NY, USA), pp. 175–185, ACM, 2011.

[20] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: Eliminating server idle power," in *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, (New York, NY, USA), pp. 205–216, ACM, 2009.

[21] A. Lungu, P. Bose, A. Buyuktosunoglu, and D. J. Sorin, "Dynamic power gating with quality guarantees," in *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '09, (New York, NY, USA), pp. 377–382, ACM, 2009.

[22] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, ISLPED '04, (New York, NY, USA), pp. 32–37, ACM, 2004.

[23] "http://whatis.techtarget.com/definition/dynamic-voltage-and-frequency-scaling-dvfs,"