Data Center Energy Efficiency Improvements through Machine Learning

Jay W. Lapham
University of Minnesota, Morris
laph0011@morris.umn.edu

ABSTRACT

This paper gives an overview of data center design and technical considerations, scheduling, and machine learning. We discuss how machine learning may be used to improve data center energy efficiency. We present ways to maintain reliability measured by Service-Level Agreements while keeping energy costs as low as possible, both by directly reducing CPU power draw and indirectly by reducing waste heating (HVAC) costs are low. We also discuss power usage improvements through application of machine learning to server-scale task scheduling, including additional benefits of data centers using virtualization technology.

Categories and Subject Descriptors

C.C.4 [**Performance of Systems**]: Performance attributes

General Terms

Algorithms, Economics, Management, Performance

Keywords

Data Center, Energy Efficiency, Scheduling, Machine Learning, Power Consumption

1. INTRODUCTION

The amount of data collected and generated on a global scale is on the order of millions of terabytes per day. With such large volumes of information to store and process into meaningful output, data centers must be as efficient as possible. It is estimated that total data center energy consumption as a percentage of total US energy consumption will have doubled between 2007 and 2012 [3]. In this paper we will describe how data centers can use machine learning techniques to help lower energy costs while still maintaining high levels of service.

In the context of this paper, a *data center* is any facility dedicated to computer systems. A data center may be as small as a single dedicated server room within a larger facility, or as a large as an entire stand-alone complex. It is generally assumed that a data center is a finely tuned controlled environment, meaning it is kept within precise thresholds for temperature, humidity, and other factors. *Efficiency* is a very broad term that in this context means either *computational efficiency* or *energy efficiency*. Computational efficiency refers to how well a data center can process large quantities of data and how reliably it can do so. Energy efficiency refers to how well a data center and its supporting infrastructure can keep energy costs and CO_2 emissions low [3]. Both types of efficiency are very important for any organization to maintain, though we shall focus on how to achieve better energy efficiency without compromising computational efficiency.

Infrastructure costs for data centers are very high, on the order of tens to hundreds of millions of dollars. The current maximum rate in the U.S. for the year of 2011 was about \$0.16 per kilowatt hour. [2]. Total data center energy consumption for the United States alone was approximately 7 gigawatts in 2010 [7]. While \$0.16 per kw/hour may not seem substantial, total electrical costs for data centers in the U.S. alone was on the order of hundreds of millions of dollars in 2010. With such high costs, any savings are more than welcome for an organization of any size operating a data center, and large organizations stand to benefit the most from a reduction in energy consumption.

With machine learning techniques, it is possible to improve energy efficiency in data centers by improving how computational workload is distributed across multiple servers. This is done by improving *scheduling*, which is carried out by a supervising program (a *scheduler*) that determines how to distribute a large number of tasks to a large number of servers [1].

The following section will cover background information, giving a basic overview of both data centers and machine learning. Section 3 concerns itself with the details of scheduling and how it functions in regards to user satisfaction. Section 4 covers how to improve scheduling through the utilization of machine learning. Section 5 concludes the paper, giving a brief review of all material covered.

2. BACKGROUND

There are two important topics to cover before examining how to improve data center efficiency with machine learning: information related to data centers, and information related to machine learning.

UMM CSci Senior Seminar Conference, April 2012 Morris, MN.

2.1 Data Centers

Data centers are buildings or sections of buildings that are dedicated to housing large quantities of computing hardware for performing large-scale computational tasks. They are normally self-contained environments and are operational continuously. As a result, data centers require large amounts of power on a continual basis [9]. One key piece of technology used in data centers is the *Power Distribution Unit (PDU)*. PDUs distribute power as necessary to devices, and may also transform larger capacity power feeds into lower capacity power feeds if necessary [4]. An example of this would be a PDU connected to an external power line supplying power to many small servers. Data center *topologies* are how PDUs are structured to provide power to computational hardware. There are many different topologies available, a few of which are illustrated in Figure 1. It is likewise important to choose a topology that can easily accomodate growth if necessary, while still affording modularity and redundancy. Different topologies offer a broad spectrum of redundancy and complexity, where some topologies ensure greater reliability at the expense of higher setup and operating costs.

Redundancy is a key concern, so many data centers have redundant PDUs and pathways to connect servers to power and to other systems. This is done to prevent the center from failing if one or more of its components ceases to function properly. In the case of total external power failure, diesel generators, uninterruptible power supplies, or battery banks may be used to maintain system integrity [8]. These systems may be used to sustain power until a connection to an external power source can be restored, or to provide a buffer to save data and shut down the center gracefully. However, it is important to note that there must be a balance in ensuring reliability and keeping costs low: more infrastructure costs more upfront and requires more energy incurring further costs. In fact, many data centers are quite energy-inefficient, and their infrastructure is often overly complex and unnecessarily expensive [7]. Any data center must find a solution that works as a middle ground between reliability and cost according to their needs.

Many servers in a single location also produce a large amount of waste heat, which must be managed as well. This issue can be refered to as *Heating, Ventilation, and Air Conditioning (HVAC)* and is the second immediate concern regarding a data center. Any large amount of computational hardware creates a large amount of waste heat, which can be both a fire hazard and reduce efficiency. Forced air systems are commonplace, as they are energy efficient and more effective at distributing cooling effects [10].

Finally, it is important that a data center have a reasonable overall layout, taking both power and HVAC concerns into account. From an architectural standpoint, it is important that data centers be laid out in a way that promotes good airflow and infrastructure is easily maintained. Thus, by reducing the number of machines running at a given time, we may reduce energy costs both by reducing the direct power usage of computing hardware and by reducing the amount of energy required by support architecture such as cooling mechanisms.

A metric known as a *service-level agreement (SLA)* is used to gauge how effectively a data center operates. There are many different types of SLAs, each with different criteria. Typical SLAs are agreements on application resource consumption such as quotas on bandwidth, disk, and CPU usage [1]. They may also include performance guarantees on things such as response time, throughput, and time deadlines. SLAs are not necessarily fulfilled by better performance, but simply when performance is good enough for an end-user. A simple commonly-known metric that may be part of an SLA is *uptime*, which is the amount of time that a computer is functioning at an acceptible level. Uptime may be represented as a percentage for a fixed time period, or may be represented in days-since-last-downtime. Schedulers therefore must meet SLA requirements to ensure smooth operation of a data center.

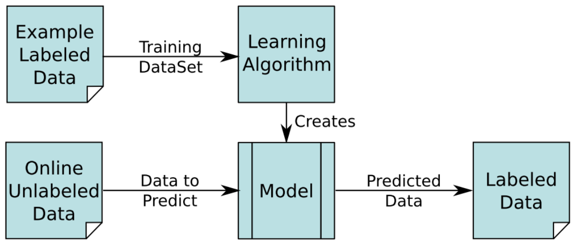


Figure 2: Supervised Machine Learning Schema [1].

2.2 Machine Learning

Machine Learning is a broad field of computer science that studies how to train computers to learn desirable behaviors. It is a branch of artificial intelligence concerned with machines learning to perform tasks, and with machines becoming better at performing said task with more experience [12].

Supervised machine learning is a technique may be used to create intelligent decision-making algorithms. It requires some prepared data, where both the input and desired output are known and well-defined, so that the program can be *trained* on said data and its performance level evaluated. A *model* is then built and its characteristics tweaked during an iterative improvement process. This process is usually repeated for many iterations, and the end result is a program that is very good at predicting the training output data from the training input data. This hopefully then translates to similarly good performance on unknown input data.

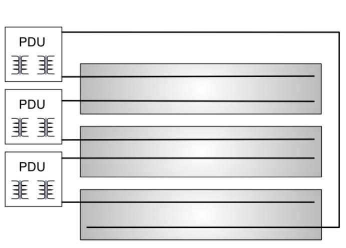
Figure 2 demonstrates this concept. As shown, a program is trained on known data that is referred to as the training dataset, and the resulting program creates a model that it uses to make conclusions from unknown data that has not been specially prepared, i.e. the same data used in training the algorithm. The model is normally tweaked and upgraded to keep results as accurate as possible over time, though the model is normally immutable once it is exposed to unknown data to prevent outliers from having an adverse effect on future results.

In the simulation to follow in section 4.1, the specific solution we shall use implements *decision trees* and these will be explained in that section.

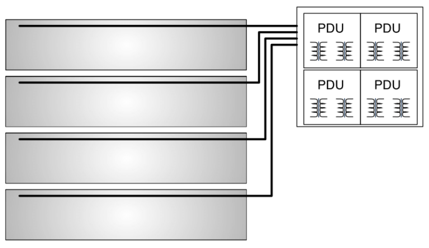
The end goal of machine learning is to build a model that can make educated guesses like a human being could, except in an automated fashion. The model itself is not intelligent, and once produced by the learning algorithm does not normally improve or change over time on its own.

3. SCHEDULING

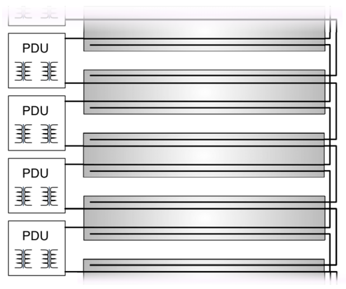
There are two well-known methods for conserving power



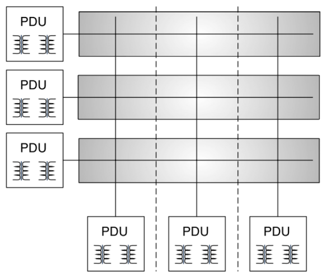
(a) Wrapped (conventional)



(b) Fully-connected



(c) Serpentine



(d) X-Y

Figure 1: Example of power distribution topologies [7]. Grey boxes are banks of servers; Power Distribution Units are labelled as PDUs, and the black lines are electrical lines. "Conventional" is an alternative name for "Wrapped."

and reducing emissions in a data center: *workload consolidation* and turning off servers when they are not required to perform computational tasks [1]. Workload consolidation is a complex task that involves shifting work to servers so that they are as close to 100% utilization as possible at any given time. This task is performed by schedulers, and a more efficient scheduling program directly translates to energy savings and lower heat output. Turning off spare servers seems a very simple task, but is inseparably tied to workload consolidation. Good consolidation means some servers may sit idle and be eligible for temporary off-lining. Poor consolidation means that many servers may be online and functioning at low levels of utilization. Both tasks of workload consolidation and off-lining idle servers are managed by a scheduling program.

A typical graph of Wattage consumed by a server in terms of CPU utilization is given in Figure 3, showing the difference in benefit between an idle server, one at maximum utilization, and one that is completely offline. As shown, it is highly desirable to offline CPUs instead of merely reducing their load to 0% so they are idling.

In a perfect setting, a scheduler has three things that allow it to function perfectly: as much time as it needs for its own computations, all possible information it may require to make decisions, and hardware that is capable of responding instantaneously. In a real-world setting, none of these three things are available. As a result, a technique for predicting outcomes and filling in gaps in missing information is highly desirable. Such a system would be able to function in less-than-optimal circumstances as are often encountered in reality as compared to theory.

For the purposes of this discussion, we will also assume that onlining and offlining host machines takes a negligible amount of time. In reality, this time frame could range from a few seconds to a few minutes, and should certainly be taken into consideration as well.

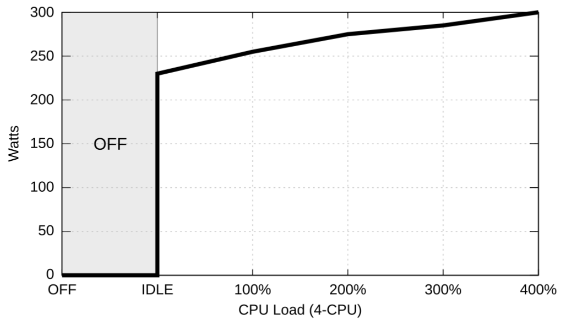


Figure 3: Example of wattage as a function of CPU load [1].

A scheduling program's most basic job description is to "move job j from its current host to host h" such that there will be greater workload consolidation and thus maximize expected benefit [1]. There are two important factors to keep track of when jobs are moved: how much performance impact there is in transitioning the job to a new host which we will represent as R, and the difference in energy consumption as a result of workload consolidation which we will represent as C. Normally, there is some significant performance loss when transitioning jobs between hosts due to hardware limitations. Generally, R and C for any given host cannot be known prior to performing the job transition, so it is necessary to have some method to predict what the estimated values, $\widehat{R}$ and $\widehat{C}$, will be.

The variable R_h indicates the *health status* of the jobs running on some host machine *h*. This variable ranges from 0, which indicates completely unacceptable performance, to 100, which indicates optimum performance from an end-user perspective. This does not mean that a value of 100 indicates optimum performance, it simply means that from a user's perspective, all jobs are finished well within a desired time frame. The health status of an individual job is denoted R_j. R_h, then, is the aggregate of values R_j for all allocated jobs on host *h*.

In formulaic terms, a *finished job j*, a task completed by a data center's computing hardware, is represented by the following tuple:

$$j = \langle \text{User}T_j, \text{SLAFactor}_j, \text{Start}T_j, \text{End}T_j \rangle$$

Here, UserT_j is the user estimation of the time to complete the task *j*, SLAFactor$_j$ is the threshold of failure above UserT_j that the user is willing to accept according to the Service-Level Agreement, and StartT_j and EndT_j are the start and end times of the task *j*.

The health status R_j can be calculated by means of the following formula:

$$R_j = f(\text{User}T_j, \text{SLAFactor}_j, \text{Start}T_j, \text{End}T_j)$$

The function f is something that is negotiated to suit a particular data center's needs. This function indicates the penalty for not satisfying the user's requirements, and gauges how well the job *j* was completed independently of which particular host the job was completed on. Two examples of such forumlae follow, where f_{hard} is a very strict function that has binary results, and f_{soft} which has a uniform distribution from 0 to 100 based upon performance:

$$f_{hard} = \begin{cases} 100 & \text{if } \text{End}T_j - \text{Start}T_j \le \text{User}T_j * \text{SLAF}_j \\ 0 & \text{otherwise} \end{cases}$$

$$f_{soft} = \max(100, \frac{\text{User}T_j}{\text{End}T_j - \text{Start}T_j} * \text{SLAF}_j * 100)$$

We will proceed under the assumption that we are using the softer version of *f*, as it is more lenient and will have more precise results as to what is acceptable and unacceptable performance. We will also assume we have a total of H hosts in our theoretical data center.

We focus on how a server performs overall rather than how well it performs individual tasks, hence R_h is a better metric to use instead of tracking individual tasks.

C_h indicates the power consumption for a given machine *h*, and can be measured directly during testing. It correlates strongly with CPU utilization, though not necessarily in a linear fashion. As previously stated, offline is a far cheaper state in terms of energy to be in than the idling state. The global function for a scheduler to optimize is therefore:

$$R = \sum_{j}^{Jobs_h} R_j/(NumJobs_h) \qquad \text{and} \qquad C = \sum_{h=1}^{H} C_h$$

where we wish to maximize the first function and minimize the second [1]. It is a reasonable decision to optimize for the maximization of R so long as we do not also increase C. This means we favor maintaining the given SLA as a priority over minimizing power consumption, as is a standard practice to date.

Our scheduler, then, being able to move jobs as they are in-progress from host to host, is a *Dynamic Backfilling (DB)* scheduler. The backfilling part of DB refers to the overall goal of this particular scheduling technique to fill as many hosts to as full as possible. This theoretically frees as many hosts as possible, allowing empty ones to be turned off to conserve power. However, Backfilling on its own does not allow for jobs to be moved if they are currently in-progress. Once a Backfilling scheduler assigns a job to a host, that job remains on said host until it is completed. This is what a Dynamic Backfilling solution improves upon: a Dynamic Backfilling scheduler can and does move jobs that are in-progress between different hosts, improving workload consolidation in real-time as jobs arrive and finish. To differentiate this from standard Dynamic Backfilling, we will refer to this approach as *Machine Learning Dynamic Backfilling (MLDB).*

4. MACHINE LEARNING IN SCHEDULING

4.1 Machine Learning Details

The algorithm that builds the model for predicting values of R and C as explained in section 3 will combined linear regression and an algorithm called *M5P*, which builds a decision tree where there is a possibility of linear regression at each node [1]. Linear regression may be used to predict values of R effectively, but as shown in Figure 3, the relationship between CPU usage of a host and that host's power consumption is nonlinear. This is where M5P is useful as it can model nonlinear relationships.

Linear regression is a statistical modelling tool that assumes a linear relationship between the inputs and outputs. Linear regression may be used for the prediction of unknown output values where all input values are known [11]. The M5P implementation uses decision trees to model for nonlinearity. *Decision trees* are tree-based data structures that classify data based on conditionals related to independent inputs. Checks are made at each node, flowing down the tree until a classification is made at a leaf. An example M5P decision tree can be seen in Figure 4. In this diagram, the first conditional check is whether a host's CPU usage is above or below 40%. This check divides the training data into two groups which need not be equal in size, using statistical tools with the goal of maximizing the linearity within each group. Then, we split these two groups again into four groups based upon whether the host machine has more or less than a certain number of jobs. This split does not have to be on the same number of jobs for each branch of the tree. This processing of classifying and splitting data continues until there is strong linearity within the individual groups of data at the tree's leaves. It is also possible for this model to be unbalanced, where one branch needs many more splits than another branch before it is reduced to strong linearity. Where a traditional decision tree has simple classifications at the leaves, M5P instead builds linear regression models at the leaves [5]. This allows us to use linear regression on data sets that have some nonlinearity, such as the relationship between CPU usage and power consumption.

4.2 Simulation Outline

Now that we have covered how scheduling and work distribution functions, we can incorporate elements of machine learning to build an intelligent program that can cope with non-optimal situations by comparing partial situations to

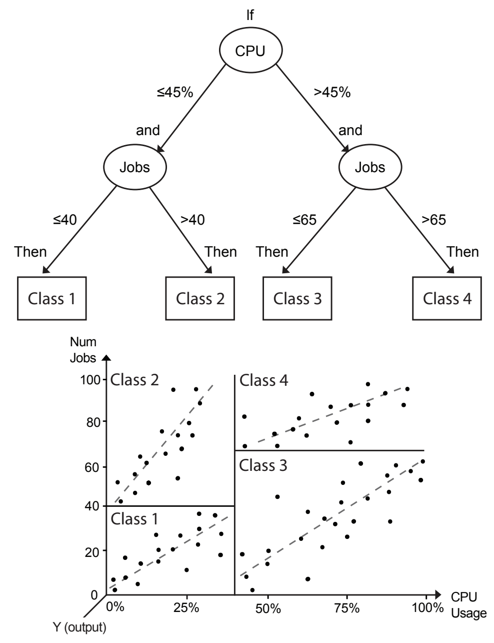


Figure 4: An example M5P tree with associated linear regression subsets.

previous experiences. To evaluate how well the machine learning approach works, we shall compare it to several other techniques. *Random* is as its name implies - it randomly allocates jobs to hosts, so long as a particular host can accommodate it. This is an unintelligent method of scheduling, and serves to give us a baseline by which to compare intelligent schedulers. *Round Robin* tasks to each available host, ensuring that each host is utilized, but usually to a low degree as tasks are not consolidated. *Backfilling* is a technique that attempts to fill each individual host as much as possible, meaning some machines may be eligible for offlining to conserve power. *Dynamic Backfilling* works as previously described at the end of section 3: it is essentially Backfilling but also gives the scheduler the ability to move tasks that are currently in-progress to other hosts, recalculating and reconsolidating as tasks enter and exit the system. Dynamic Backfilling has one failing, however: it does not function well when there is incomplete or imprecise information about the tasks in the system. This is where the *Machine Learning Dynamic Backfilling* (MLDB) technique we have previously discussed comes in, as it can make intelligent estimates about tasks if information is missing.

The simulation consisted of 400 hosts running on three different workloads. First, the Grid workload is one week's worth of data from October 2007 from the Grid5000 data center [1]. SLAs have been added to this workload by Berral et al, as it did not originally include them.

The second workload is an aggregate of services based on the load of *Ask.com*, corresponding to three different profiles. The first is a from 00:00 to 23:59 with a typical daytime usage increase and a low usage level during the night. The second workload displays the same behavior but has a larger increase in service usage in the afternoon. The third makes use of an entire week's workload to take the weekend activity decrease into account.

Algorithm 1 Algorithm for move selection

```
Poll hosts for information about their jobs and status;
OH := select "Emptiable Machines" [jobs < 4];
For each Machine (oh) in OH do:
    For each Job (j) in oh do:
        CH := select "Fillable Machines" [enough CPU and mem];
        For each Machine (ch) in CH do:
            -- predict effect of moving j from oh to ch;
            predict R(oh) and R(ch) after movement;
            predict C(oh) and C(ch) after movement;
            compute global R and C after movement;
        End For
        Get ch leading to highest R among those that decrease C;
        add movement (j,oh,ch) to List_of_movements;
    End For
    If (all jobs in oh can be reallocated) then:
        proceed with the List_of_movements;
    End If
End For
```

Figure 5: Pseudocode for move selection [1].

The final workload will consist of a heterogenous mix of both the Grid and Service workloads.

The key decision our scheduler must make is how to move jobs such that it may offline as many unnecessary hosts as possible. Pseudocode may be found for this in Figure 5. Our scheduler looks for hosts with few active tasks, and then checks to see what it estimates the performance impact and energy efficiency impact of moving those tasks would be. The "predict" functions in the pseudocode are direct uses of our machine learning solution. The input data is taken as input to our M5P decision tree, which can give us a meaningful answer about whether it would be beneficial to move jobs. If it finds a good match in keeping with the rules of decreasing C without decreasing R significantly, it moves the jobs to a different host. We wish to move jobs so that we offline as many machines as possible, thus decreasing total power consumption, so long as we do not negatively impact computational performance. This fulfills the task of workload consolidation.

4.3 Results

Totalled results can be found in Table 1. Overall, Dynamic Backfilling without a machine learning component still meets SLAs to a better degree than the other four techniques tested. Compared to each of the other scheduling techniques, it is clear that Machine Learning Dynamic Backfilling is more costly on the Grid workload. However, in the other two cases, MLDB maintained both a very high degree of SLA satisfaction and resulted in less kilowatts consumed by a significant degree. In the heterogenous workload simulation, no method achieved 100% SLA satisfaction, with MLDB coming in about 1% short. However, its overall power consumption was about 10% better than the next-best method, which is a significant improvement. It is clear from the results that the MLDB technique loses compared to standard Dynamic Backfilling for the grid workloads, and thus is most useful on service or heterogeneous workloads. Dynamic Backfilling is superior to MLDB on grid workloads possibly due to MLDB being more responsive to new tasks being added at uncertain intervals in service workloads [1].

5. CONCLUSIONS

As demonstrated, the use of machine learning can be very useful in improving the efficiency of a data center in terms

	Working Nodes (avg)	Running nodes (avg)	CPU usage (hours)	Power (kW)	SLA (%)
		Grid Workload			
Round Robin	16.11	41.37	5954.91	1696.66	85.99
Random	16.51	40.76	6017.85	1671.16	88.38
Backfilling	10.18	27.10	6022.34	1141.65	100.00
Dynamic Backfilling	9.91	26.46	6104.33	1118.86	100.00
Machine Learning DB	15.04	37.92	6022.27	1574.78	99.69
		Service Workload			
Round Robin	290.99	400.00	78419.97	19761.54	100.00
Random	218.46	400.00	75336.88	19784.38	100.00
Backfilling	108.79	352.88	59792.09	16257.26	100.00
Dynamic Backfilling	108.79	352.88	59748.10	16229.22	100.00
Machine Learning DB	99.61	270.50	61379.38	13673.71	100.00
		Heterogeneous Workload			
Round Robin	260.66	400.00	84432.96	19713.72	94.20
Random	224.08	400.00	82137.27	19763.63	88.53
Backfilling	110.85	330.19	65894.46	16304.38	99.50
Dynamic Backfilling	111.03	329.07	66020.58	16214.49	99.59
Machine Learning DB	124.20	307.89	68554.01	15110.33	98.63

Table 1: Simulation results, taken from [1].

of power usage. These improvements have implications for both improving data center capacity and reducing data center costs, both of which are highly desirable benefits. Energy efficiency can be greatly improved with a variety of techniques as demonstrated, while still maintaining data center service availability, reliability, and performance.

6. ACKNOWLEDGMENTS

I thank Nic McPhee and Elena Machkasova of the University of Minnesota, Morris computer science department for their phenominal assistance in constructing this paper. I thank Kevin Arhelger for his domain knowledge and feeback as well. I thank Katlyn Jaeger for her assistance in designing figures. I also thank my numerous other peers who reviewed and gave feedback on this paper during its creation.

7. REFERENCES

[1] J. L. Berral, I. n. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, and J. Torres. Towards energy-aware scheduling in data centers using machine learning. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 215–224, New York, NY, USA, 2010. ACM.

[2] C. P. U. Comission. Rates and chart tables - electricity (2000 thru 2011). `http://www.cpuc.ca.gov/PUC/energy/Electric+Rates/ENGRD/ratesNCharts_elect.htm`, 2011. [Online; accessed 8-April-2012].

[3] I. Dumitru, I. Fagarasan, S. Iliescu, Y. H. Said, and S. Ploix. Increasing energy efficiency in data centers using energy management. In *Proceedings of the 2011 IEEE/ACM International Conference on Green Computing and Communications*, GREENCOM '11, pages 159–165, Washington, DC, USA, 2011. IEEE Computer Society.

[4] R. Koller, A. Verma, and A. Neogi. Wattapp: an application aware power meter for shared data centers. In *Proceedings of the 7th international conference on Autonomic computing*, ICAC '10, pages 31–40, New York, NY, USA, 2010. ACM.

[5] S. Kramer. M5P - OpenTox Documentation. `http://www.opentox.org/dev/documentation/components/m5p`, 2011. [Online; accessed 17-April-2012].

[6] B. Pariseau. A leg-up on raised floors? `http://itknowledgeexchange.techtarget.com/data-center/a-leg-up-on-raised-floors/`, 2011. [Online; accessed 8-March-2012].

[7] S. Pelley, D. Meisner, P. Zandevakili, T. F. Wenisch, and J. Underwood. Power routing: dynamic power provisioning in the data center. In *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems*, ASPLOS '10, pages 231–242, New York, NY, USA, 2010. ACM.

[8] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No "power" struggles: coordinated multi-level power management for the data center. In *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, ASPLOS XIII, pages 48–59, New York, NY, USA, 2008. ACM.

[9] Wikipedia. Data center — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Data_center&oldid=480607304`, 2012. [Online; accessed 7-March-2012].

[10] Wikipedia. HVAC — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=HVAC&oldid=480291658`, 2012. [Online; accessed 7-March-2012].

[11] Wikipedia. Linear regression — wikipedia, the free encyclopedia, 2012. [Online; accessed 24-April-2012].

[12] Wikipedia. Machine learning — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=478057288`, 2012. [Online; accessed 7-March-2012].