Utility-Aware Deferred Load Balancing in the Cloud Driven by Dynamic Pricing of Electricity

Muhammad Abdullah Adnan and Rajesh Gupta
University of California San Diego

***Abstract*—Distributed computing resources in a cloud computing environment provides an opportunity to reduce energy and its cost by shifting loads in response to dynamically varying availability of energy. This variation in electrical power availability is represented in its dynamically changing price that can be used to drive workload deferral against performance requirements. But such deferral may cause user dissatisfaction. In this paper, we quantify the impact of deferral on user satisfaction and utilize flexibility from the service level agreements (SLAs) for deferral to adapt with dynamic price variation. We differentiate among the jobs based on their requirements for responsiveness and schedule them for energy saving while meeting deadlines and user satisfaction. Representing utility as decaying functions along with workload deferral, we make a balance between loss of user satisfaction and energy efficiency. We model delay as decaying functions and guarantee that no job violates the maximum deadline, and we minimize the overall energy cost. Our simulation on MapReduce traces show that energy consumption can be reduced by ∼15%, with such utility-aware deferred load balancing. We also found that considering utility as a decaying function gives better cost reduction than load balancing with a fixed deadline.**

I. Introduction

The recent increase in energy prices along with the rise of cloud computing brings up the issue of making clouds energy efficient; as according to an EPA report data centers consume 61 million MWh per year in US, costing about 4.5 billion dollars [1]. There has been a lot of exploration under the quest for green data centers, searching for opportunities to reduce energy consumption in the context of cloud computing resources. While there are a number of hardware and software techniques for energy savings considering different aspects, one non-conventional perspective is to utilize the predetermined service level agreements (SLAs) for energy efficiency. Specifically, latency is an important performance metric for any web-based services and is of great interest for service providers who are responsible for services on the cloud.

Naturally, energy efficiency in the cloud has been pursued in various ways including the use of renewable energy [2], [3] and improved scheduling algorithms [4], [5], etc. Among these, improved scheduling algorithm is a promising approach for its broad applicability regardless of hardware configurations. Typically, a SLA specification provides a measure of flexibility in scheduling that can be exploited to improve performance and efficiency. Recent work explores the opportunities from the SLAs to shift and balance computational loads in a cloud computing environment and delay computation for improving the performance of the applications (e.g. see [6], [7], [8]). But delayed scheduling may cause customer dissatisfaction and sometimes increases the overall cost of execution if future variation of electricity pricing is not considered. In this paper, we use deferral with dynamic pricing of electricity for energy efficiency while using utility functions to capture the loss in the value of execution by deferral.

Many applications in real world have latency constraints or delay bounds, especially in Cloud Computing. When combining with energy conservation, latency is usually a critical adjusting tool between performance loss, energy consumption and customer satisfaction. Lee et al. [9] conducted a survey to measure the impact of delay on user satisfaction and represented delay as utility functions which are often decaying with time. Jensen et al. [10] used time utility functions (TUFs) for scheduling in real time systems with latency constraints and energy bound. Our work uses similar utility functions and extends their usage with dynamic deferral by making a trade-off between energy efficiency and loss of utility. Most prior work either uses delay as a constraint or minimizes the average delay of overall computation. Unlike previous work, we represent delay as inversely proportional to utility as increase in delay diminishes the utility for executing a job. Then we determine the schedule that minimizes the net cost for execution which is the product of delay and energy price. We apply our technique for both homogeneous and heterogeneous workloads and show significant cost reductions with respect to the greedy heuristic that does not take into account the utility of execution.

This paper makes two contributions. First, we present online algorithms for utility-aware load balancing in data centers with dynamic deferral. When all the jobs have same utility function (homogeneous), we present a formulation that balances load uniformly. But when the jobs have different utility functions (heterogeneous), our formulation uses individual utility functions for each job to determine schedule. Second, we validate our algorithms using MapReduce traces (representative workload for data centers) and evaluate cost savings using synthetic utility functions generated from the shapes extracted from surveys in the literature [11]. In practice, utility functions are provided by the users during the job submissions.

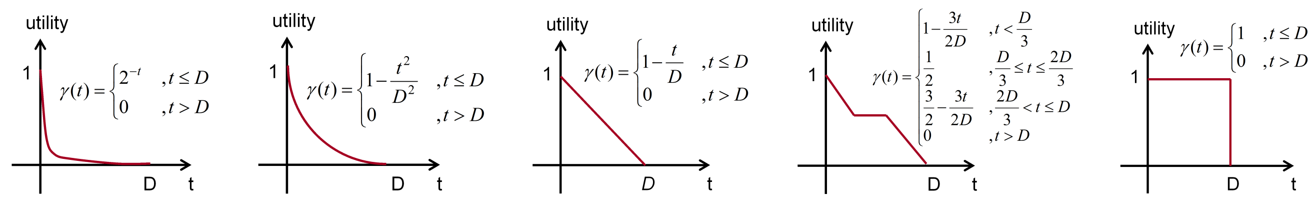
The rest of the paper is organized as follows. Section II presents the model we use to formulate the optimization. In Section III, we present the algorithm for determining optimal assignment for both homogeneous and heterogeneous workload. Section IV evaluates our algorithm on real electricity prices and workload data and Section V concludes the paper.

II. Model Formulation

We consider a large computing facility ("cloud"), consisting of n computation units. The computation units can represent machines in a datacenter or geographically distributed datacenters or servers waiting for requests. We divide time into slots for the ease of load balancing decisions. The cost for executing the jobs may vary with time, yet remain fixed within a time slot of length τ. The time interval we are interested in is $t \in \{0, 1, \ldots, T\}$ where T can be arbitrarily large. In practice, T can be a year and the length of a time slot τ could be as small as milliseconds for service requests (e.g. HTTP) or as large as several minutes for batch-like jobs (e.g. MapReduce). At each time slot t, the total workload L_t arrive at the central dispatcher from which load balancing decisions are made.

In our model, the jobs can be delayed but the utility for execution of a job can diminish with time, which we represent by a *utility function*, $\gamma(t) : \mathbf{N} \rightarrow [0, 1]$. Figure 1 illustrates different types of utility functions where the maximum value

978-3-9815370-0-0/DATE13/2013 EDAA



(a) Exponential (b) Polynomial (c) Linear (d) Decay (e) Step

Fig. 1. Different utility functions representing user satisfaction. Utility for the execution of a job is maximum when the job is released ($t = 0$).

of utility for a job is 1 when the job is released and the utility decays with time. Since all the jobs do not loose value at the same rate, the utility function can take any decaying form [9]. Interactive jobs usually have steeper utility functions (Figure 1(a), 1(b)) while batch jobs have flatter utility (Figure 1(d), 1(e)). For interactive jobs utility becomes zero after a specified period of time while for batch jobs utility may diminish with time but never become zero. For the sake of formulation we assume that the utility function has a maximum deadline D, after which the utility becomes zero. This helps us guarantee that all the jobs will be finished after a certain period of time. First we consider the uniform case with a single utility function for all the workload. Later we extend it for the case where different jobs may have different utility functions.

At the beginning of each time slot t, the released jobs are assigned to machines/servers with deferral (delay) of $0 - D$ slots. The total computation capacity M_i in server i is fixed and given for $1 \leq i \leq n$. We normalize the workload L_t (released at time t) by the processing capability of the servers i.e. L_t denotes the computation capacity required to execute the workload. Hence we assume that for all t, $L_t \leq \sum_{i=1}^{n} M_i$. Note that this condition is not necessary since we can defer the workload to later slots. But we do not consider that case in this paper. We assume that the cost for execution may vary with time. Hence it might be beneficial to defer the execution of some tasks for cost saving. Let $x_{i,d,t}$ denote the portion of the workload L_t, assigned to be executed at time $t + d$ at server i for $0 \leq d \leq D$. Then $\sum_{i=1}^{n}\sum_{d=0}^{D} x_{i,d,t} = L_t$. If $t < 0$, $L_t = 0$ and $x_{i,d,t} = 0$ for all i, d.

The goal of this paper is to minimize the total cost (price) for executing the workload making a trade-off between energy price and utility loss by deferred execution. The energy cost $C_{i,t}(y_{i,t})$ is the cost for executing the workload $y_{i,t}$ at server i at time slot t. We assume that $C_{i,t}(y_{i,t})$ is a nonnegative, (weakly) convex increasing function. Note that the function itself can change with time, which allows for time variation in energy prices. The simplest example for the cost function for a time slot is an affine function which is the common model for the energy cost for typical servers:

$$C_{i,t}(y_{i,t}) = \alpha_i + \beta_{i,t} y_{i,t}$$

where α_i and $\beta_{i,t}$ are constants for server i and time slot t (e.g. see [2]) and $y_{i,t}$ is the executed workload to server i at time t. Since we assume that the future price of energy is not known, we use predicted parameters for future costs, $\tilde{C}_{i,t}(y_{i,t}) = \alpha_i + \tilde{\beta}_{i,t} y_{i,t}$ at time slots $t' > t$.

III. Algorithm

In this section, we present an online algorithm for the utility-aware deferred load balancing problem. Given the model, the goal of the dispatcher is to determine the dispatching rule $x_{i,d,t}$ to minimize the total cost during $[1, T]$. Since the workload is not known in advance, we formulate an online optimization which is applied at each time t, over $[t, t + D]$ slots to determine the assignment for the released workload L_t. In the formulation, we represent delay as inversely proportional to utility as increase in delay diminishes the utility for executing a job. Then our objective is to minimize the net cost for execution which is the product of delay (with respect to the release time) and energy price. As jobs are delayed, the net cost for execution increases. So it is better to execute the jobs early. But the energy price may be higher in the earlier time slots. Hence we make the trade-off by the product of energy price and delay for the execution of a job. Thus for each job j, the objective is to minimize: $\frac{1}{\text{utility}_j} \times$ execution cost_j Therefore each job is executed with a certain utility at the time and place that minimizes the total net value of the execution of the jobs. By taking the sum of the product of delay and energy cost for all the jobs, we put more priority to the jobs with steeper utility functions (interactive jobs). We first present a formulation for the homogeneous workload where all the jobs have the same utility function. Later, we extend it for the heterogeneous workload.

A. Homogeneous Workload

The dispatcher makes decision on the assignment of the workload to the servers. It has access to the past and current prices, but only statistical knowledge about future prices, which it can use to make optimal decision for future time slots. The following optimization is applied at the dispatcher at each time t to determine the assignment of workload $x_{i,d,t}$ to servers for $0 \leq d \leq D$ and $1 \leq i \leq n$.

$$\min_{x_{i,d,t}} \quad \sum_{i=1}^{n}\sum_{d=0}^{D} \frac{1}{\gamma(d)} \cdot C_{i,t}(x_{i,d,t-d}) + \sum_{i=1}^{n}\sum_{k=1}^{D}\sum_{d=k}^{D} \frac{1}{\gamma(d)} \cdot \tilde{C}_{i,t+k}(x_{i,d,t+k-d}) \tag{1a}$$

$$\text{subj. to} \sum_{i=1}^{n}\sum_{d=0}^{D} x_{i,d,t} = L_t \tag{1b}$$

$$0 \leq \sum_{d=s-t}^{D} x_{i,d,s-d} \leq M_i \quad \forall i, t \leq s \leq t + D. \tag{1c}$$

where $\tilde{C}_{i,t'}()$ is the predicted cost function at time $t' > t$ for data center i and $x_{i,d,t''}$ is the unexecuted workload at data center i that was assigned at time $t'' < t$ to be executed at time $t'' + d$ where $t - t'' \leq d \leq D$. Initially, $x_{i,d,t} = 0$ when $t < 0$. Constraint (1b) ensures that total new assignment is equal to the total released workload and constraint (1c) ensures that total assignment to a server does not exceed the capacity of the server.

Since the energy cost function $C_{i,t}(\cdot)$ is an affine function and $\gamma(d)$ is constant for the time slot d, the objective function is linear as well as the constraints. Hence it is clear that the optimization in (1) is a linear program. Note that the workload $x_{i,d,t}$ in the formulation is not considered to be integer. This is acceptable because the number of requests at each time slot is in the range of thousands and we can round the resulting assignment with minimal increase in cost.

B. Heterogeneous Workload

We now consider the case where different jobs have different utility functions and hence have different deadlines. We distinguish between the jobs based on their utility functions. Suppose there are J types of utility functions. Then the workload can be decomposed into J classes, with workload in each class $L_{j,t}$ having the same utility function $\gamma_j(\cdot)$ where $1 \le j \le J$. To represent different classes of workload, we include another dimension by adding an index j with the notations for workload assignment $x_{i,d,j,t}$. Then we can reformulate the optimization at time t, for $0 \le d \le D$, $1 \le i \le n$ and $1 \le j \le J$,

$$\min_{x_{i,d,j,t}} \sum_{i=1}^{n}\sum_{d=0}^{D}\sum_{j=1}^{J} \frac{1}{\gamma_j(d)} \cdot C_{i,t}(x_{i,d,j,t-d})$$

$$+ \sum_{i=1}^{n}\sum_{k=1}^{D}\sum_{d=k}^{D}\sum_{j=1}^{J} \frac{1}{\gamma_j(d)} \cdot \tilde{C}_{i,t+k}(x_{i,d,j,t+k-d}) \quad (2a)$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\sum_{d=0}^{D} x_{i,d,j,t} = L_{j,t} \qquad \forall j \quad (2b)$$

$$0 \le \sum_{d=s-t}^{D}\sum_{j=1}^{J} x_{i,d,j,s-d} \le M_i \quad \forall i, t \le s \le t+D \quad (2c)$$

IV. Simulation

In this section, we evaluate the cost incurred by the utility-aware load balancing algorithm relative to the greedy solution in the context of workload generated from realistic data.

Electricity Price: We assume that the servers are distributed geographically at distant locations. We run our simulations for four sites and choose distant locations near those power grids whose real time electricity prices are publicly available. We used the publicly available data from electricity markets from Independent System Operator New England (ISO-NE) [12], New York Independent System Operator (NYISO) [13], Electric Reliability Council of Texas (ERCOT) [14] and Electricity Market of New Zealand (NZ) [15]. We took the locational based marginal prices (LBMP) from the 5 minute spot markets for three days (15th, 14th and 8th February, 2012) and ran our experiments on the prices of 15th February using the prices for 14th and 8th for prediction of future prices. We use the four locations to have both temporal and geographical variation of electricity prices e.g. the time zones of New York, New England, Texas and New Zealand are GMT-5, GMT-7, GMT-6 and GMT+13 respectively. The variation of electricity prices for different locations are plotted in Figure 2 with Eastern Standard Time (EST). These graphs indicate significant spatio-temporal variation in electricity prices.

Workload Description: We use two publicly available MapReduce traces as examples of dynamic workload. The MapReduce traces were released by Chen et al. [16] which are produced from real Facebook traces for one day (24 hours) from a cluster of 600 machines. We count the number of different types of job submissions over a time slot length of 5 minutes and use that as a dynamic workload (Figure 3) for simulation. The two samples we use represent strong diurnal properties and have variation from typical workload (Workload A) to bursty workload (Workload B). For the experiments we used time slot length of 5 minutes because the electricity prices vary with an interval of 5 minutes. In practice, load balancing decisions can be made more frequently with slot length size in the range of seconds. We then assign deadline for each job in terms of the number of slots the job can be delayed. For the case of homogeneous workload, we set D to be 12 slots and use the utility functions from Figure 1 for the simulation. This is realistic because MapReduce workloads have deadlines in the range of minutes as deadlines of 8-30 minutes from SLAs for these workloads have been used in the literature [7], [8]. For the heterogeneous case, we use k-means clustering to classify the MapReduce workload into five job groups (JGA-JGE) based on the map, shuffle and reduce bytes. The characteristics of each group are depicted in Table I where smaller jobs dominate the workload mix. This kind of clustering has been used by Chen et al. [16] for classifying the workload. For each class of jobs we assign a utility function from Figure 1(a)-(e) such that for larger class (small/interactive jobs) utility decays quickly and for smaller class (large/batch jobs) utility decays slowly.

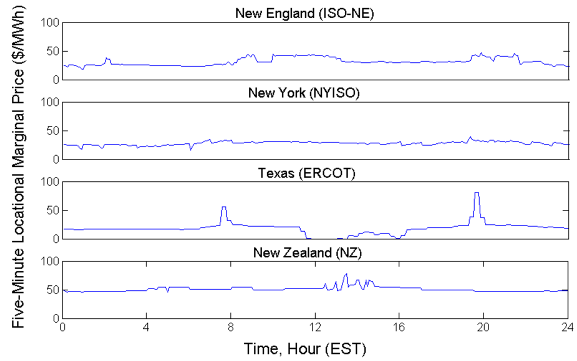
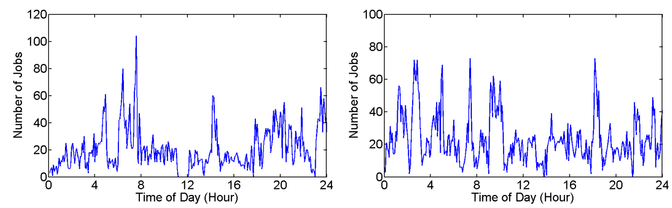


Fig. 2. Illustration of five minute locational marginal electricity prices in real time market on 15th February, 2012 for four different regions (a) New England (ISO-NE), (b) New York (NYISO), (c) Texas (ERCOT), (d) New Zealand (NZ).



(a) Workload A (b) Workload B

Fig. 3. Illustration of traces for dynamic workload used in the experiments.

TABLE I
CLUSTER SIZES AND UTILITY FUNCTIONS FOR WORKLOAD CLASSIFICATION BY K-MEANS CLUSTERING FOR HETEROGENEOUS WORKLOAD

Cluster	Workload A		Workload B		Utility
	#Jobs	GB	#Jobs	GB	Figure 1
JGA	5700	1.49	6272	1.18	(a)
JGB	145	86.76	263	49.73	(b)
JGC	33	253.95	56	141.71	(c)
JGD	14	651.63	38	441.24	(d)
JGE	2	8702.16	9	1296.13	(e)

Utility Functions: We generated utility functions synthetically for each job using a family of curves drawn from those in Figure 1. We understand that often utility functions are not explicit, as previous work showed users can describe their job utility [9] but often schedulers do not specifically request that information. In fact, users do suffer a loss in value with job deferral and this loss is not uniform as some jobs loose value faster than others or by different functions. Our goal is to capture these phenomena by the utility functions and their heterogeneity.

Cost Parameters: The cost function parameter $\beta_{i,t}$ is determined using current electricity price and the future prices, $\tilde{\beta}_{i,t'}$ for $t' > t$, are determined using a electricity price prediction model. For our simulations, we use load independent parameter $\alpha_i = 0$, for all i. Depending on the nature of the workload we set the total capacity of each site because the algorithms keep on assigning the workload to the site with lowest cost until the site is overloaded. For both the workload, we use capacity $M_i = 50$ for all i.

The future electricity prices $\tilde{\beta}_{i,t}$ for the next D time slots are randomly generated from Gaussian distributions because of their high unpredictability and the volume (D) of generation. We used the same mean but different variances for the generation in each time slot. We use the optimal daily coefficients for the price prediction filter from [17]. Let $\tilde{\mu}_i^t[\chi]$ and $\tilde{\sigma}_i^t[\chi]$ be the predicted means and standard deviations for each time slot t on day χ for geographical location i. Then the mean of the prediction model for Gaussian distribution is obtained as follows:

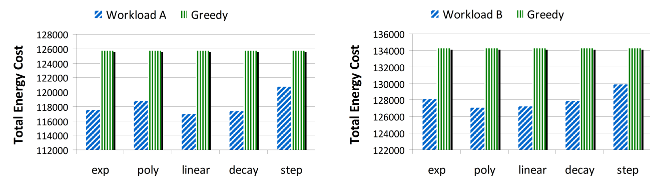
$$\tilde{\mu}_i^t = \varepsilon_0 + \sum_{j=0}^{D} \varepsilon_{t-j}\beta_{i,t-j}.$$

Here, ε_j are the coefficients for the moving average method which can be estimated by training the model over the previous day prices. The variance parameter $\tilde{\sigma}_i^t[\chi]$ is estimated from the history using the following equation:

$$\tilde{\sigma}_i^t[\chi] = k_1\sigma_i^t[\chi-1] + k_2\sigma_i^t[\chi-2] + k_7\sigma_i^t[\chi-7].$$

Here, $\sigma_i^t[\chi-1]$, $\sigma_i^t[\chi-2]$ and $\sigma_i^t[\chi-7]$ denote the previous standard deviation values σ_i^t on yesterday, the day before yesterday and the same day last week, respectively. Since we used the electricity prices for Wednesday (15th February, 2012), we chose the $k_1 = 0.837$, $k_2 = 0$ and $k_7 = 0.142$ from [17]. For the previous standard deviation values ($\sigma_i^t[\chi-1]$, $\sigma_i^t[\chi-7]$) values we use the past standard deviation of electricity prices for D slots on those days such that $\sigma_i^t[\chi-1] := std(\beta_{i,t}[\chi-1], \beta_{i,t-1}[\chi-1], \ldots, \beta_{i,t-D}[\chi-1])$ and $\sigma_i^t[\chi-7] := std(\beta_{i,t}[\chi-7], \beta_{i,t-1}[\chi-7], \ldots, \beta_{i,t-D}[\chi-7])$; where $std(.)$ denotes the standard deviation. The mean $\tilde{\mu}_i^t[\chi]$ is computed from the moving average of the prices for D previous slots on the current day χ. By using two different methods for mean and variance, we exploit both the temporal and historical correlation of electricity prices.

Evaluation: Currently load balancing in the cloud typically does not use deferral of workload [2]. Often the load balancing decisions are made dynamically using the greedy method [4]. In this method, jobs are assigned to the data centers with the lowest current electricity prices without considering utility or deferral. Using our algorithms, we can make a trade-off between energy efficiency and utility loss by deferring some of the workload to later time slots. We compare the total energy cost from our algorithms with the greedy strategy without deferral and evaluate the cost reduction. Figure 4 illustrates the cost reduction with each of the utility functions for the homogeneous case. For both the workload, step function performs poorly due to error in future price prediction. This shows the benefit of representing utility (delay) as a decaying function over time, giving less weight for the later slots at the time of load balancing decisions. Hence considering utility



(a) Workload A (b) Workload B

Fig. 4. Comparison of total cost from utility-aware load balancing and greedy method for homogeneous workload.

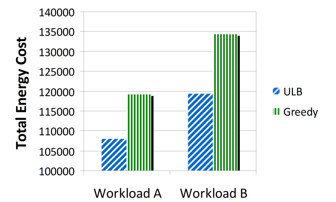


Fig. 5. Comparison of total cost from utility-aware load balancing (ULB) and greedy method for heterogeneous workload.

as a decaying function is better than representing utility by a deadline. Figure 5 depicts the cost reduction for the workload mix from Table I. It shows that distinguishing the workload and scheduling them according to the shape (steepness) of their utility functions results in cost savings of ∼15%.

V. Conclusion

We have proposed online algorithms for utility-aware load balancing with dynamic deferral. The algorithms optimize the dynamic assignment of the workload to the servers by making a trade-off between energy consumption and utility loss due to deferral with future electricity price prediction. Our simulations highlight that significant cost and energy savings can be achieved using the algorithms. Our future work is to consider other parameters such as heterogeneity of servers, capacity provisioning etc. during the load balancing decisions.

References

[1] *Server and Data Center Energy Efficiency,* Final Report to Congress, U.S. Environmental Protection Agency, 2007.

[2] Z. Liu, M. Lin, A. Wierman, S. Low, and L. H. Andrew, *Greening Geographical Load Balancing,* In Proc. ACM SIGMETRICS, 2011.

[3] B. Aksanli, T. Simunic Rosing, and I. Monga, *Benefits of Green Energy and Proportionality in High Speed Wide Area Networks Connecting Data Centers,* In Proc. of DATE, 2012.

[4] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, B. Maggs, *Cutting the electric bill for internet-scale systems,* In Proc. ACM SIGCOMM, 2009.

[5] M. Lin, A. Wierman, L. H. Andrew, E. Thereska, *Dynamic right-sizing for power-proportional data centers*, In Proc. IEEE INFOCOM, 2011.

[6] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleey, S. Shenker, and I. Stoica, *Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling*, in Proc. of ACM EuroSys, 2010.

[7] A. Verma, L. Cherkasova, R. Campbell, *Resource Provisioning Framework for MapReduce Jobs with Performance Goals,* In Proc. of Middleware, 2011.

[8] S. K. Garg, R. Buyya. *SLA-based Resource Provisioning for Heterogeneous Workloads in a Virtualized Cloud Datacenter*, Proc. ICA3PP, 2011.

[9] C. B. Lee, A. E. Snavely, *Precise and realistic utility functions for user-centric performance analysis of schedulers*, In Proc. HPDC, 2007.

[10] E. D. Jensen, C. D. Locke, and H. Tokuda, *A time-driven scheduling model for real-time systems*, In Proc. of IEEE RTSS, 1985.

[11] C. B. Lee, and A. E. Snavely, *On the User-Scheduler Dialogue: Studies of User-Provided Runtime Estimates and Utility Functions*, Int'l Journal of High Performance Computing Applications, 20(4):495-506, 2006.

[12] http://www.iso-ne.com.

[13] http://www.nyiso.com.

[14] http://www.ercot.com.

[15] http://www.electricityinfo.co.nz.

[16] Y. Chen, A. Ganapathi, R.Griffith, R. Katz, *The Case for Evaluating MapReduce Performance Using Workload Suites,* in Proc. of IEEE MASCOTS, 2011.

[17] A. H. Mohsenian-Rad and A. Leon-Garcia, *Optimal residential load control with price prediction in real-time electricity pricing environments,* IEEE Trans. Smart Grid, 1(2), pp. 120-133, Sep. 2010.