Utilizing Green Energy Prediction to Schedule Mixed Batch and Service Jobs in Data Centers

Baris Aksanli, Jagannathan Venkatesh, Liuyi Zhang, Tajana Rosing
Computer Science and Engineering Dept. (CSE)
University of California, San Diego
La Jolla, CA
E-mail: {baksanli, jvenkate, liz004, tajana}@ucsd.edu

ABSTRACT

As brown energy costs grow, renewable energy becomes more widely used. Previous work focused on using immediately available green energy to supplement the non-renewable, or brown energy at the cost of canceling and rescheduling jobs whenever the green energy availability is too low [16]. In this paper we design an adaptive data center job scheduler which utilizes short term prediction of solar and wind energy production. This enables us to scale the number of jobs to the expected energy availability, thus reducing the number of cancelled jobs by 4x and improving green energy usage efficiency by 3x over just utilizing the immediately available green energy.

Keywords

Data centers, green energy, prediction, MapReduce.

1. INTRODUCTION

Green energy sources promise to mitigate the issues surrounding non-renewable generation, but their output is very susceptible to environmental changes. This limits the use of green energy in time-sensitive applications. Prediction can reduce the uncertainty of the available resources, allowing end-users to scale demand with the predicted supply [17]. Data centers are a significant source of energy consumption with an estimated 2% global greenhouse gas emissions attributed to them [18]. However, the time-sensitive nature of their service-level workloads has precluded the use of green energy, as jobs might need to be stopped when the available green energy drops [16].

Data centers also have longer-running batch jobs (on the order of tens of minutes [9]) whose performance is measured in terms of throughput and job completion times instead of latency guarantees (e.g. web crawling, index update in search engines, web log analysis [07]). A number of computing frameworks have been developed to simplify the process of those jobs. Examples include MapReduce [25], Dryad [26], and Pregel [5]. The fault-tolerant nature of these frameworks mitigates source instability, allowing execution of a subset of the tasks in a job in order to scale with the available energy, as well as allowing re-execution of cancelled tasks that have been stopped due to a sudden lack of input energy.

Green energy prediction over short time intervals (tens of minutes) alleviates these issues by scaling the workload to the expected available green energy, resulting in better maintenance of forward progress and allowing more tasks/jobs to continue their execution even if instantaneous green energy supply drops below the necessary amount. The system offsets the remainder of the immediate need with brown energy with the assurance that over the prediction interval the average green energy will ultimately be available. This allows a more efficient use of the available energy; reducing the amount of wasted green energy and the number of tasks/jobs that must be re-executed; and ultimately, increasing the overall throughput of the data center.

The contribution of this paper is to develop a new data center job scheduling methodology that effectively leverages green energy prediction. We simulate a data center of 200 Intel Nehalem servers using measured data obtained on a small test bed of Nehalem servers that ran a mix of services (Rubis [6]) and batch jobs (MapReduce[8]). Our scheduler ensures that the required response time targets for services are met while maximizing the completion times and the number of MapReduce tasks run. We use green power data from a solar installation in San Diego [15], and wind power from National Renewable Energy Laboratory (NREL) [14] as our sources of green energy. Our results show maximum increase of 3x in green energy usage efficiency, a 1.6x increase in the amount of work performed by green energy over brown energy, and a 7.7x reduction in the number of jobs terminated due to the lack of instantaneously available green energy.

2. RELATED WORK

2.1 Energy Prediction

Solar energy prediction is typically obtained with estimated weighted moving average (EWMA) models, because of its relative consistency and periodic patterns [19]. As long as the weather conditions remain consistent within a period, the prediction is accurate, but becomes inaccurate, with mean error well over 20%, with frequent weather changes. Recent work utilizing small-scale solar generation uses a weather-conditioned moving average (WCMA), taking into account the mean value across days and a measured factor of solar conditions in the present day relative to previous days [20]. While this work provides only a single future interval of prediction, it specifically addresses inconsistent conditions, with a mean error of under 10%.

Wind energy prediction can be separated into two major areas: time-series analysis of power data; and wind speed prediction and conversion into power. Kusiak et al. [21] presents a comparison of several methodologies of time-series modeling of wind farms. The boosting-tree algorithm with both wind speed and power data performs well in their analysis, while the integrated model, a time-series analysis utilizing only wind speed measurements, performs poorly for calculating wind power, likely due to the cubic relationship between wind speed and power. Giebel et al. [22] focuses on the latter, describing a number of meteorological models including Numerical Weather Prediction (NWP), which forecasts atmospheric conditions over longer term. They use the resulting predictions to simulate the output of a wind farm providing accurate estimates for 3-6 hour time periods. However, this comes at the cost of needed a whole data center to calculate prediction. Sanchez et al. [24] suggest a statistical forecasting system that generates power curves (wind speed vs. wind power) for each turbine based on meteorological information and machine characteristics. They then utilize the power curves and available wind data for forecasting.

2.2 Green Energy in Data Centers

Green energy usage in a data center environment is a relatively new topic. Gmach et al. [2] augment a data center with PV and municipal solid waste based energy. However, since solid waste energy supply is constant over time, they do not address the problem of variability in renewable energy supply. Lee et al. [3] model an optimization problem which uses the market prices of brown and green energy to decide how much energy of each type should be bought in each interval. They do not make server level scheduling decisions based on the amount of green energy.

Stewart and Shen [4] analyze the energy requirement distributions of different requests and how to integrate green energy to the system. They state that the variable nature of green energy can be a problem, but do not propose solutions. Gmach et. al. [1] use wind and solar energy to cap the power usage of a data center environment. The paper addresses the problem of variability of green energy and overcomes this problem by adding extra energy storage. Krioukov et al. [16] use renewable energy for execution of MapReduce type jobs. They schedule MapReduce tasks with available green energy, but terminate them when the scheduler realizes that there is not enough green energy in subsequent intervals.

Our work, in contrast, uses prediction methods to estimate the amount of green energy in a given interval and utilizes that data to make decisions about scheduling policies of individual servers. We aim to increase the green energy usage efficiency by prediction as well as reduce the destructive impact of the variable nature of the green energy sources on batch job completion times. Additionally, unlike previous work, we include service jobs and batch jobs together in our model to obtain a more realistic system view, as data centers normally see both types of workloads.

3. SOLAR AND WIND ENERGY PREDICTION

The focus of current work on large-scale green energy prediction is on medium to long-range time horizons lasting from hours to days. As such, the techniques are highly complex, requiring intensive data acquisition and analysis from using SCADA units [19] for solar energy to entire data centers [22] for NWP wind prediction models. Our prediction interval needs to be only as long as the workloads we desire to schedule, which is on the order of tens of minutes (our predictor uses 30 min). We chose this interval based on run-time experiments on the scalable, fault-tolerant Hadoop framework [8], which we use as our batch workload. Furthermore, as the response time constraints of services that run in data centers can be quite short (tens of ms), our job scheduler and predictor need to be fast. As a result, we designed solar and wind energy prediction models of lower complexity and shorter time horizons.

3.1 Solar Prediction Methodology

We applied various time-series prediction algorithms described in the related work to the output data retrieved from a solar farm at the University of California, San Diego, [15]. While most solar prediction algorithms are accurate when weather conditions are stable, EWMA algorithms have 32.6% mean error in variable weather. The WCMA algorithm [20], when repurposed by us for larger solar installations (instead of the wireless sensor networks it was originally designed for), performed very well, with a mean error of 9.6% for 30 min prediction window even in artificially-created worst-case scenarios.

3.2 Wind Prediction Methodology

We develop a novel, low-overhead predictor that utilizes readily available data that has been shown to strongly correlate with wind energy prediction [21]: wind speed and wind direction. Our algorithm produces weighted nearest-neighbor (NN) tables to generate wind power curves using available wind speed and direction data at each 30-minute interval. Weighted tables allow the algorithm to adapt to seasonal changes by weighting recent results highly, while the power curves offer flexibility, allowing the algorithm to be used with different wind farms. The appropriate power curve table gets updated using the current interval's observed wind velocity, direction, and output power as follows:

$$P_{new}(v, d) = \alpha * P_{obs}(v, d, t) + (1-\alpha) * P_{old}(v, d) \qquad (1)$$

here $P_{new}(v,d)$ is the new power curve table entry for a given wind velocity v and direction d, $P_{old}(v,d)$ is the existing value for the same velocity and direction, and $P_{obs}(v,d,t)$ is the observed value at time t. While α can vary from 0 to 1, we found most consistent results with $\alpha=0.75$, which weights the model more heavily towards currently observed data. Future interval prediction uses a table lookup based on the predicted wind velocity and direction:

$$P_{pred}(v, d, t+k) = P(v(t+k), d(t+k)) \qquad (2)$$

The algorithm has been tested against a wind farm installation over a year's worth of power output data provided by the NREL, and the meteorological data provided by the National Climatic Data Center (NCDC). The results show a mean error of 17.2% for a 30-minute prediction interval, equaling or outperforming the time-series models described in [21], at much lower computational cost.

4. GREEN ENERGY SCHEDULING AND DATA CENTER MODELING

Our goal in this work is to evaluate the benefit of green energy prediction for increasing the data center job throughput while not sacrificing service jobs' response time constraints. To accomplish this we designed both predictive and instantaneous green energy based schedulers and compare them to the baseline of using only brown energy. The scheduler uses two separate job arrival queues as shown in

Figure 1. One queue is for web services that have response time requirements (e.g. 90th percentile should be less than 150ms), and the other for batch jobs which are more concerned about throughput and job completion time. When a web services client request arrives, the controller allocates a server that has the smallest number of batch jobs running on it in order to reduce the interference effects between these two types of workloads. Additionally, we put a limit to the number of clients a host can serve to distribute the web-requests evenly among servers. This limit is determined by using current number of clients and total number of host machines. For simplicity, we assume that each server has at minimum one web services request queue, and one or more batch jobs slots to execute. Web services start execution whenever there are available computing resources (CPU and memory) to ensure their response time requirements are met whenever possible. Therefore, we guarantee that the system provides enough brown energy to maintain these service requests. In this work we use Rubis as representative of web services [6]. Based on our measurements and [11] we model the interarrival time of Rubis requests generated by a client using a lognormal distribution.

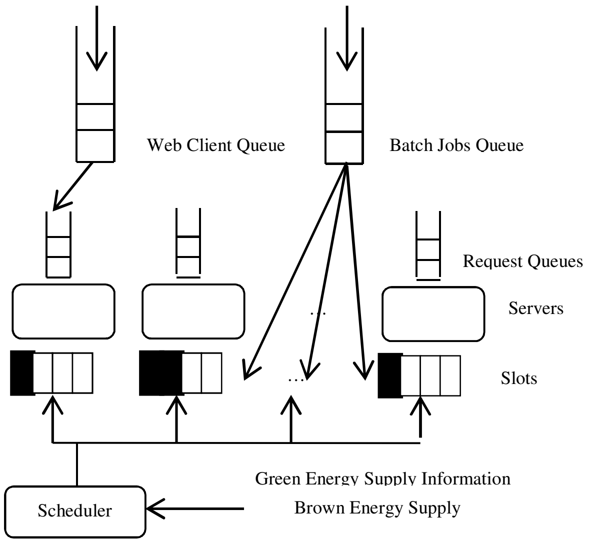

Figure 1: System architecture

We use open source version of MapReduce, Hadoop [8], to represent batch jobs. Input data of any given job is split and processed by many map/reduce tasks distributed across a fixed number of map/reduce slots in a cluster as shown in Figure 1. If there are more tasks than the available slots, the tasks without slots are queued. If any task fails, the MapReduce scheduler starts a fresh copy the task. The arrival process of this type of jobs is modeled by a lognormal distribution, as demonstrated in [9]. The total number of servers given to a job depends on the energy availability & green energy scheduling algorithms. At each time instance, power consumption of servers is estimated using a linear model based on CPU utilization as in [10]. The overall data center energy cost is calculated using aggregate server power scaled by the power utilization efficiency ratio (PUE) to account for the impact of other sources of inefficiencies (e.g. cooling costs). We use our data center test bed measured average PUE value of 1.26.

Predictive green energy scheduler: Our green energy predictor uses a 30-min prediction interval, a duration that is longer than that of our run-time tests of MapReduce jobs to ensure enough energy is available to finish the tasks. The predictor provides the scheduler with an estimate of the next period's average green energy availability at the beginning of each batch job allocation interval. It then computes the number of batch job slots that can be used for the given amount of energy in that interval. When computing the number of extra slots the scheduler uses the average power/slot information we got from our measurements (see next subsection). If this number is greater than the current number of available slots, the remaining extra slots are distributed to the active MapReduce cluster, so that they can run more tasks in parallel. However, if this number is smaller, then the scheduler deallocates some jobs. Jobs that run more concurrent tasks than their base requirement have their slots reduced first. The tasks running in deallocated slots are either immediately terminated or restarted with green energy later on (jobs using more than needed slots), or continue but use brown energy instead. This decision is made depending on the number of concurrent tasks in a job. The energy consumed to run the terminated jobs in the previous interval is wasted. In the results section, we quantify this cost of incorrect energy prediction by using the green energy usage efficiency metric. The main benefit of a predictor is that the number of deallocated slots for batch jobs can be dramatically reduced, and the number of available slots increased.

Instantaneous green energy scheduler: We compare the impact of green energy prediction to the instantaneous use of green energy presented in [16]. To simplify evaluation we use the same algorithm as predictive scheduler, but with a 1min scheduling interval which reflects the instantaneous case well.

4.1 Model validation using experimental testbed

We developed a discrete event-based simulation platform for scheduling a mix of service and batch jobs in a data center consisting of hundreds of servers. This enables us to evaluate the impact of using a combination of brown and green energy at scale. To ensure accuracy of our estimates, the parameters for our event-based simulator are obtained from measurements on Intel Nehalem [12] servers when running a mix of service (Rubis [6]) and batch workloads (MapReduce [7]) within Xen VMs. Rubis and MapReduce are run in separate VMs, with MapReduce run across 2 VMs, one utilizing 4 cores, and the other varying the number of cores occupied. Rubis is run with 9000 concurrent users.

Table 1: Measured interference of MapReduce and Rubis

# cores MapReduce	***1-4***	***5***	***6***	***7***	***8***
Rubis QoS	0.0.47	0.08	0.1	0.4	0.93
MapReduce Perf.	100%	94%	88%	83%	81%

Table 1 shows the measurements we obtained by scheduling an increasing number of MapReduce tasks on the same machine with service requests. We report a measure of normalized response time as Quality of Service (QoS) ratio, which is calculated using 90th percentile response time over the expected response time (for Rubis it is 150ms). We see that even in the worst case, where we allocate the maximum number of available cores to MapReduce jobs, normalized response time of Rubis, as measured by QoS ratio, is still less than 1. In addition, we see that the worst case performance impact on normalized MapReduce job completion times is maximum 20%. Mean measured service time of a single map or reduce task is around 10 minutes, though the maximum can be as high as 20 min, thus justifying our choice of 30min green energy prediction interval.

Table 2: Verification of simulation outputs

	Measured	Simulated	Error
Avg. Power Consumption	246 W	251 W	3%
Rubis QoS ratio	0.08	0.085	6%
Avg. MapReduce Comp. Time	112 min	121 min	8%

Given the measurements presented above, in our simulations we use 150ms as the target Rubis response time with 12ms service times for 1000 to 5000 clients representing different times in a day, 2min mean arrival time of MapReduce jobs [9] with average execution time of 10 min. To ensure that in our simulations we have at most 10% performance impact on MapReduce tasks, we use 5 slots per server. We compare simulation results using this setup to actual measurements on the Nehalem server. Table 2 shows that the average error is well below 10% for all quantities of interest, with power estimates having only 3% average error, while performance for services has only 6% and MapReduce completion times are within 8%.

5. RESULTS

We use our discrete event-based simulation platform to schedule a mix of service (Rubis) and batch jobs (MapReduce) in typical data center container consisting of 200 Intel Nehalem servers. The overall duration of simulation is 4.5 days. Simulations are repeated until we obtain a statistically stable average.

Each server has a single web service queue that servers multiple clients. Incoming client requests are distributed over the servers evenly. The client arrival distribution is assumed to be exponential as in [23], while client requests are generated using a lognormal distribution with mean 100 ms and 15 ms as mean service time. MapReduce jobs arrive to the system with a mean of 2 min and each task has 10 min execution time on average. We use 5 MapReduce slots per host. Services QoS ratio in all of our simulations remains between 0.09 and 0.2, thus ensuring that web request response time requirements are never violated. The ratio gets closer to 1 when the number of web services clients exceeds 10000. The average queue length for web requests is 0.8 for 1000 clients and 5.5 for 5000 clients.

We use a number of metrics reported in Table 3 to compare our predictive scheduler *(Pred.)* with the state of the art instantaneous green energy usage (Inst.) [16] when using only wind, only solar and combined two green energy sources. We define *GE Efficiency* as the ratio of the green energy doing useful work versus the total green energy available: GE_{useful_work} / GE_{tot}. Energy consumed by a task that is terminated before completion is not counted as a part of GE_{useful_work}. Green energy under-prediction is penalized by this metric. The percentage of jobs that are terminated as a result of the lack of green energy at the beginning of the scheduling interval, *% incomplete jobs*, is calculated relative to the overall number of jobs completed using green energy. This occurs when jobs launched with currently available green energy in a previous scheduling interval cannot be sustained due to the energy availability drop in the subsequent interval. Lastly, the efficiency of the system is in terms of green energy usage, *GE Job Ratio*, is defined as the total amount of work done with green energy, $Jobs_{GE}$, over the total work done in the system, $Jobs_{tot}$: $Jobs_{GE}/Jobs_{tot}$.

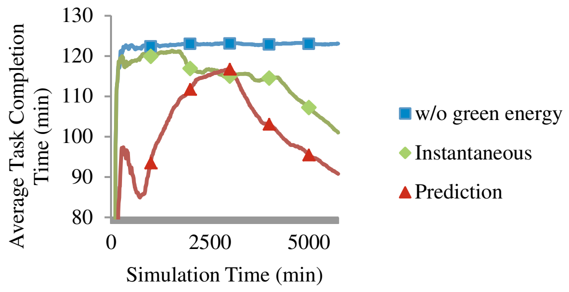

Figure 2: Average completion time of MapReduce tasks

Table 3 shows that prediction improves green energy efficiency up to 3x relative to instantaneous energy. The main reason for this result is that the system has good quality information on green energy availability for a longer interval and hence can make better scheduling decisions. Therefore, less green energy is wasted and 5x fewer MapReduce tasks need to be terminated. Finally, our predictive scheduler increases the number of MapReduce tasks executed with green energy by 2x relative to the instantaneous approach as a result of more accurate energy provisioning. Figure 2 shows how the average completion time of MapReduce jobs changes over time as a function of the way green energy is used. The baseline case uses only brown energy to run a mix of services with just enough MapReduce jobs so that services response time constraints and performance requirements of MapReduce jobs (maximum 10% hit to completion times) are met. In this scenario, we create the MapReduce jobs at the same rate to highlight the green energy effect more clearly. Our green energy prediction scheduler decreases MapReduce task completion times on average by 20%. In contrast, instantaneous usage of green energy results in 12% higher average batch task completion times compared to prediction.

An alternate way to compare using predicted vs. instantaneous green energy schedulers is to supplement with

Table 3: Comparison of instantaneous and predicted green energy with different alternative energy sources

	Wind Energy		Solar Energy		Combined	
	Inst.	*Pred.*	*Inst.*	*Pred.*	*Inst.*	*Pred.*
GE Efficiency	30% ± 2.5%	90% ± 2%	60% ± 5 %	93% ± 2 %	72% ± 5 %	93% ± 3 %
GE Job Ratio	35% ± 5 %	50% ± 5%	28% ± 4 %	45% ± 3%	40% ± 4 %	55% ± 5 %
% incomplete jobs	10% ± 3.3 %	1.3% ± 0.4 %	8.6% ± 2.5 %	2.4% ± 0.5 %	12% ± 2.5 %	3% ± 0.5 %

brown energy whenever there is not enough green energy to complete batch jobs. In this way we ensure that all service jobs meet their response time requirements and all batch jobs complete, so none are terminated. The first column of Table 4 shows the amount of brown energy needed to run all the tasks in the absence of green energy. When we use green energy instantaneously and do not terminate any tasks when there is not enough green energy available, we need extra 4.6 kWh of brown energy per data center container, but if we use our predictor, the extra brown energy needed is decreased by more than 7x to 0.64 kWh.

Table 4: Brown Energy for Inst. vs. Pred. Energy

Total BE w/o GE	Add. BE for Inst.	Add. BE for Pred.
240 kWh	4.6 kWh	0.64 kWh

6. CONCLUSIONS

As the cost of brown energy is becoming a critical bottleneck in data center environments, the need for alternative energy sources is growing. In this paper we present a novel green energy predictor, along with a data center scheduling policy which uses prediction information to obtain better performance for batch jobs without significantly affecting the performance of latency sensitive web requests. We use a simulation platform to compare our predictive policy with instantaneous use of green energy. Our simulation platform has been verified by measurements on real systems, with maximum 8% error across all relevant metrics. Our results show that prediction leads to 3x better green energy usage and reduces the number of terminated tasks up to 7.7x compared to instantaneous green energy usage. The response time requirements of web requests stay well below the 90th%ile during all the experiments.

7. ACKNOWLEDGEMENTS

This work was sponsored in part by the Multiscale Systems Center (MuSyC), National Science Foundation (NSF) Project GreenLight, NSF ERC CIAN, NSF Variability Expedition, NSF Flash Gordon, CNS, NSF IRNC TransLight/StarLight, Oracle, Microsoft and Google.

8. REFERENCES

[1] D. Gmach, J. Rolia, C. Bash, Y. Chen, T. Christian, A. Shah, R. Sharma and Z. Wang. “Capacity Planning and Power Management to Exploit Sustainable Energy”. *International Conference on Network and Service Management. CNSM’10.* 2010.

[2] D. Gmach, Y. Chen, A. Shah, J. Rolia, C. Bash, T. Christian, R. Sharma. “Profiling sustainability of data centers”. *Sustainable Systems and Technology (ISSST), 2010 IEEE International Symposium.* 2010

[3] K. Le, R. Bianchini, T. D. Nguyen, O. Bilgir, M. Martonosi. “Capping the brown energy consumption of Internet services at low cost”. In *Proceedings of the International Conference on Green Computing.* 2010.

[4] C. Stewart and K. Shen. ‘Some Joules Are More Precious Than Others: Managing Renewable Energy in the Datacenter”. *4th Workshop on Power-Aware Computing and Systems. HotPower’09.* 2009.

[5] G. Malewicz, M. H. Austern, A. J.C Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. 2010. Pregel: a system for large-scale graph processing. *In Proceedings of the 2010 international conference on Management of data (SIGMOD '10).*

[6] Rubis. http://rubis.ow2.org/

[7] C. Tang, S. Tara, R. Chang, C. Zhang. “Black-Box Performance Control For High-Volume Non-Interactive Systems”. *USENIX ’09.* 2009.

[8] Hadoop. http://hadoop.apache.org/

[9] S. Kavulya, J. Tan, R. Gandhi and P. Narasimhan. “An Analysis of Traces from a Production MapReduce Cluster”. *Carnegie Mellon University, Parallel Data Lab. Techinal Report.* DOI: CMU-PDL-09-107.

[10] D. Economou, S. Rivoire, C. Kozyrakis, P. Ranganathan. “Full system power analysis and modeling for server environments”. *In Workshop on Modeling Benchmarking and Simulation (MOBS), June 2006.*

[11] D. Ersoz, M. S. Yousif, and C. R. Das. “Characterizing Network Traffic in a Cluster-based, Multi-tier Data Center”. In *Proceedings of the 27th International Conference on Distributed Computing Systems* (ICDCS).

[12] Intel Microarchitecture Nehalem. http://www.intel.com/technology/architecture-silicon/next-gen

[13] Luiz André Barroso, Urs Hölzle. The data center as a computer: An Introduction to the Design of Warehouse-Scale Machines, 2009.

[14] National Renewable Energy Laboratory. http://www.nrel.gov/

[15] Energy Recommerce. http://www.mypvdata.com/

[16] A. Krioukov, C. Goebel, S. Asplaugh, Y. Chen, D. Culler, R. Katz. “Integrating Renewable Energy Using Data Analytics Systems: Challenges and Opportunities”. *IEEE Data Engineering Bulletin. March 2011.*

[17] J. Tanega. Towards Cooperative Grids: Sensor/Actuator Networks for Renewables Integration. *Sensys 2010 Doctoral Colloquium.* 2010.

[18] B. Watson, A. Shah, M. Marwah, C. Bash, R. Sharma, C. Hoover, T. Christian, C. Patel. Integrated Design and Management of a Sustainable Data Center. *Proceedings of IPACK2009.* 2009.

[19] C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 2004.

[20] J. Piorno, C. Bergonzini, D. Atienza, T. Rosing. Prediction and management in energy harvested wireless sensor nodes. University of California, San Diego, 2010.

[21] A. Kusaik, H. Zheng, Z. Song. “Short term prediction of wind farm power: A Data Mining approach”. IEEE TEC, Vol. 24, No. 1, pp. 125-136, March 2009.

[22] G. Giebel, R Brownsword, G Kariniotakis. “The State-of-the-Art in short-term prediction of wind power – A literature overview”. Project ANEMOS Deliverable Report. August 2003.

[23] D. Meisner, T. F. Wenisch. “Stochastic queuing simulation for data center workloads”. *Proc. of the Workshop on Exascale Evaluation and Research Techniques (EXERT),* Mar. 2010.

[24] I. Sanchez: “Short-term prediction of wind energy production”. *International Journal of Forecasting*, Vol. 22, pp 43-56. 2006.

[25] J. Dean and S. Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113.

[26] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. 2007. Dryad: distributed data-parallel programs from sequential building blocks. SIGOPS Oper. Syst. Rev. 41, 3 (March 2007), 59-72.