Expressive Power-Based Resource Allocation for Data Centers

Benjamin Lubin
Harvard University*

Jeffrey O. Kephart
IBM Thomas J. Watson Research Center

Rajarshi Das
IBM Thomas J. Watson Research Center

David C. Parkes
Harvard University

Abstract

As data-center energy consumption continues to rise, efficient power management is becoming increasingly important. In this work, we examine the use of a novel market mechanism for finding the right balance between power and performance. The market enables a separation between a 'buyer side' that strives to maximize performance and a 'seller side' that strives to minimize power and other costs. A concise and scalable description language is defined for agent preferences that admits a mixed-integer program for computing optimal allocations. Experimental results demonstrate the robustness, flexibility, practicality and scalability of the architecture.

Keywords: Autonomic Computing; Auctions and Market-Based Systems

1 Introduction

In 2006, US data centers used about 61 billion kWh; that is, 1.5% of the 4 trillion kWh consumed in total. This is the amount of energy used by 5.8 million average US households (5% of all households). Producing this power resulted in 37 million metric tons of CO_2, or .6% of the 5.9 billion metric tons released from all sources. That is roughly 16% of that produced by the burning of jet fuel and more than that used to power TVs. This electricity cost US $4.5 billion and required a peak load capacity of about 7GW, more than double the level of consumption in 2000. Peak load capacity is expected to double again by 2011 to 12GW, requiring the construction of 10 additional 500MW power plants.[1] Given the rapid, unabated rise in electrical power consumption and the associated financial and environmental costs, data center operators realize that the established practice of running large numbers of significantly under-utilized servers is no longer acceptable, and are eager for energy-saving solutions.

Market paradigms have often been proposed as a useful paradigm for allocating limited computational resources and satisfying multi-criteria objectives. The earliest work on such markets was for time sharing on the PDP-11 in the 1960s by Sutherland [1968]. In the intervening years there have been proposals to use such methods in high performance computing, grid computing, as well as in data centers.

However, existing proposals have deficiencies that can render them impractical for modern data centers. We propose a general method for overcoming these concerns, and illustrate its applicability to one specific environment. We offer a:

Realistic model of resources: We support a finer granularity of computational entity (e.g. core vs server which is especially important as multi-core machines become the norm) and finer control over power state of machines (e.g. Dynamic Voltage and Frequency Scaling (DVFS), not just on/off). We also handle heterogeneous applications running on heterogeneous classes of servers.

Realistic representation of goals: We use a less restricted form of utility function that supports distributional (and percentile) considerations, not just means. These functions are derived from standard long-term Service Level Agreements (SLAs) that are programatically interpreted as short-term utility functions in a dynamic environment.

Principled optimization: We use Mixed Integer Programming, and, unlike previous systems, we do not rely on heuristic solvers but instead present a carefully formulated MIP model that can scale to large problem sizes.[2]

We show that a market-based approach provides a natural, feasible, and advantageous framework for representing the milieu of physical and computational resources, and the applications that consume these resources, in modern-day data centers. Experimental results indicate that our system can robustly and scalably improve net profits of our data center prototype by up to 137%. For large instances of 1000 machines and 10 applications (each associated with a customer), each with a demand of 2700 transactions a second, when limiting MIP solve time to 10 minutes we achieve an average solve time of 5.16 minutes, with a negligible approximation imposed on the instances that timeout. Thus a single machine is capable of optimizing the usage on 1000 others, giving us a very acceptable .1% overhead factor.

Related Work: Chase *et al.* [2005] present a compelling market-based system for data center resource allocation, and are able to experimentally demonstrate significant energy savings over static allocations. However, their greedy clearing mechanism imposes restrictions on the form of utility that can be modeled, SLAs are not directly represented, and demand/utility computations occur with respect to the mean, not distributional information. Their model does not handle

*Work completed while an intern at IBM Research

[1] *Carbon Dioxide Emissions from the Generation of Electric Power in the United States*, US DOE and EPA Report, July 2000; *Report to Congress on Server and Data Center Energy Efficiency*, US EPA Report, August 2007; *Emissions of Greenhouse Gases Report: Carbon Dioxide Emissions*, US EIA Report, December 2008.

[2] We leverage recent advances in MIP solving that enable complex combinatorial markets to be solved quickly in practice, even though they address NP-hard problems [Sandholm, 2007].

the heterogeneity of data-center machines or modern power-throttling architectures (instead simply turning machines on and off), and their allocation is at the level of servers and not cores. The non-linear costing model that we use is related to the one provided by Chen *et al.* [2005]. But rather than identify total-value maximizing allocations with respect to SLAs, they treat SLAs as constraints and attempt to find the cheapest allocation subject to meeting implied quality constraints.

Recent work on resource allocation in data centers has focused on *Utility Computing*, which seeks to provide access to the data center in a way analogous to that of a public utility (e.g., gas, water, power) [Low and Byde, 2006; Byde, 2006]. In general, Utility Computing views computational resources as more fungible than in the present work, where we assume that only particular machines are suitably configured for, and capable of running, certain applications. Rather then argue for a more radical shift in how computation is bought, sold, and deployed, in this work we propose a more gradual evolutionary step by creating a market that handles this heterogeneity in present data centers and which encompasses and generalizes the present contract format (SLAs). There is an extensive literature on using market-based methods in related contexts, including *Computational Grids* and in *High Performance Computing*. Yeo and Buyya [2006] and Broberg *et al.* [2007] are good surveys of this work, which can be informative here as well.

A market in our setting is combinatorial in nature; participants must be able to bid (indirectly, via SLAs) on 'packages' of items, where the value for a package may not be a linear combination of the value for its constituent items. There is a long literature on combinatorial auctions; an excellent overview is in the book by Cramton *et al.* [2006].

2 A Market Model of Data Center Allocation

Typical data centers have hundreds to thousands of servers, many of which will share the same hardware and software configuration. We call such equivalence classes *'machine groups'* and assume that this partitioning is performed by a separate offline process. The owner of a data center typically contracts (either internally or externally) to provide these resources to a set of applications (each associated with a customer), each with time-varying load and utility and a range of resource requirements and importance.

In present use, each application is associated with an SLA that is negotiated between customer and data-center provider. Such an agreement specifies a price, a performance objective (e.g. a cap on the 95th percentile of response time), and a penalty for failing to meet the objective. The SLA is useful for assigning a relative importance to the applications, but despite its quantitative feel it is generally used at present in only a qualitative way, as a guideline for personnel when manually configuring data-center operations. Yet, SLAs suggest a direction towards application utility functions that are highly relevant to obtaining reasonable performance in a power-constrained environment [Kephart and Das, 2007; Steinder *et al.*, 2008]. In this work, we introduce a system that adopts SLAs for the purpose of utility-based optimization of resource configurations.

When allocating resources in the data center we seek to optimize the operator's business value for the data center: i.e., the revenue net of costs. This means assigning (portions of) the machines from discrete machine groups to the various applications as well as specifying the power for each machine, and thus restraining overall consumption. For this, we use a sophisticated model of the power saving modes available to modern servers and assume access to monitors of both power consumption and application demand.

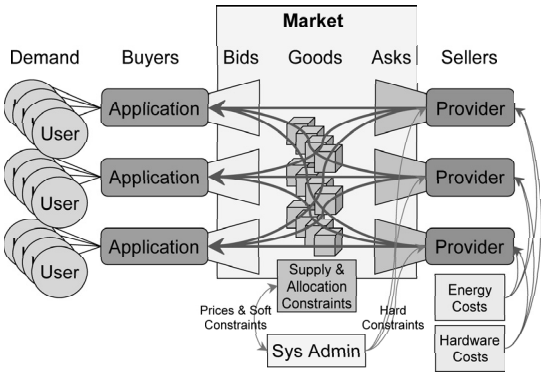


Figure 1: The Data Center Market Model

Our market allocates goods (cores of machines from the various machine groups) to applications, as illustrated in Figure 1. The market is repeatedly cleared over brief periods, by using predictions about future supply and demand to translate applications' long-term SLAs into short-term bids.

The winner-determination problem for this market requires optimization, and is potentially solved in many different ways. We choose to formulate it as a Mixed Integer Program and solve it via ILog CPLEX 11.1. The form of the buyer (or customer's) value model and seller cost model have been chosen to ease formulation of the problem as a MIP, as sketched below.[3]

For each period, we use a myopic net revenue maximization objective:

$$\max \quad \sum_{a \in A} V_a - \kappa E^{\text{TOTAL}} - H^{\text{TOTAL}}$$

where V_a is the value of the chosen allocation of machines to application (associated with a particular buyer) $a \in A$, κ is the dollar cost of a kW-hour of energy, E^{TOTAL} is the total energy used to establish and maintain the chosen allocation for the current period, and H^{TOTAL} is the dollar cost for the hardware. The objective is thus quite straightforward–the complexity comes from the constraints. We begin by defining the buyer value, V_a, i.e. the value associated with application a of some buyer.

2.1 Buyer Valuation Model

Today, the contracts signed for data center provisioning are typically in terms of SLAs. We model a per-application SLA contract as a piecewise linear function for the value of receiving a given response time at a given demand percentile. Figure 2 shows an example of an SLA value curve of this

[3]The size of the formulation will grow linearly with the number of applications and machine groups. However, it will grow with the square of the number of power modes (since it encodes a transition matrix from one mode to the next). Fortunately, polynomial growth in the size of the MIP need not imply exponential growth in practical computation time, and we examine scalability in Section 3.1.

form for two different applications A and B. A bidding proxy represents each application within the market, and takes such an SLA and combines it with supply and demand prediction and an application performance model, to represent the SLA as a short-term bid; i.e. a bid that is appropriate for the period of time for which the market is cleared.

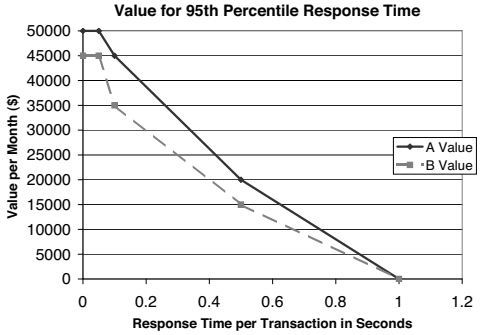


Figure 2: SLAs as Provided by two Different Applications

The bidding proxy needs a model of how a given supply of machines (and thus transaction capacity) and application demand for the next planning episode will translate to the long-term response time distribution (and in turn to its, e.g., 95th percentile), and thus to the value curve associated with an SLA. Here, as a very simple example, we consider that transaction processing is described as an M/M/1 queue (exponential inter-arrival and service times). In this case, the response time distribution is exponential with mean response time $1/(\mu - \lambda)$, where μ is the supply and λ is the demand, both in transactions per unit time. The fraction of response times above a percentile P is given by the exponential quartile function: $-\frac{\ln(1-P)}{(\mu-\lambda)}$. The proxy composes the customer's SLA (Figure 2) with this response time model, resulting in a value function over both supply and demand at, e.g., the 95th percentile (Figure 3).[4]

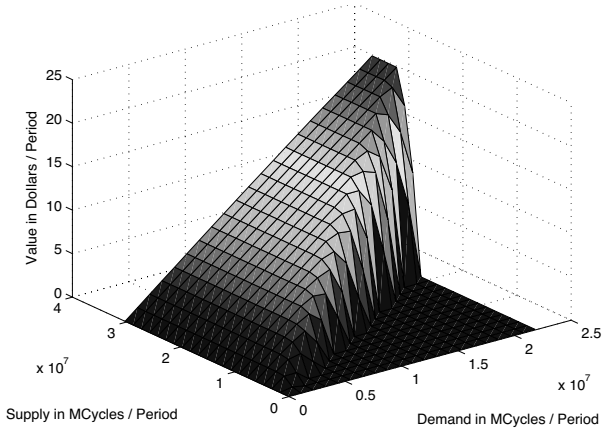


Figure 3: Application Value by Supply and Demand

Next the bidding proxy needs a predictive model of application demand over the next period. We have found it sufficient to simply use statistics gathered over a small window of previous periods to provide a Gaussian model of the distribution of possible demand in the next period via a Maximum Likelihood Estimation (MLE) prediction of mean and standard deviation. The bidding proxy draws equal weighted samples from this Gaussian demand prediction model and takes a slice from the value model (Figure 3) for each. Then, these slices are averaged to produce a single supply-value curve, under our demand model. By taking a piecewise linear approximation of this curve (obtained by chaining the control points of the originally supplied response-time/value curve through these transformations) we arrive at the utility curve provided to the market in a given period as a bid, an example of which is shown in Figure 4.

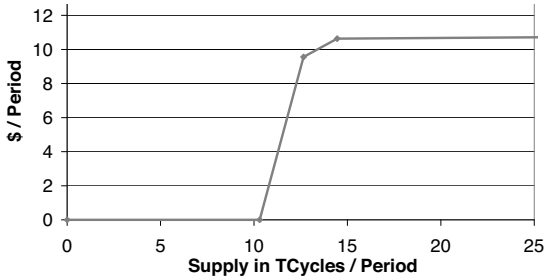


Figure 4: Expected Short-term Value for a Single Application

If we apply this function to the cycles provided by a potential allocation, then we have specified a utility function as needed by the winner determination algorithm, and with a significant dimensionality reduction:[5]

$$V_a = F_a(Q_a),$$

for application a, where F_a is this piecewise linear function and Q_a is the quantity of cycles provided to application a by the chosen allocation. To formulate this function, any standard MIP representation for a piecewise linear function can be used, which will induce auxiliary constraints and variables in order account for the various segments of the function.

In turn, the total quantity of cycles provided to application a can in a period be defined as:

$$Q_a = \sum_{g \in G_a} \sum_{f \in M_g} \sum_{t \in M_g} \gamma_{g,t}(\tau - \delta_{g,f,t}) C^{\text{SOLD}}_{g,f,t,a} \; \forall \; a \in A$$

where G_a is the set of machine groups that can support application a, M_g is the set of power modes available to machines in group g, $\gamma_{g,t}(\Delta)$ is the number of cycles provided by a machine from group g in mode t over a period of time Δ, τ is the amount of time in the current period, $\delta_{g,f,t}$ is the amount of time it takes to transition from mode f to mode t and each $C^{\text{SOLD}}_{g,f,t,a}$ variable defines a quantity of cores (i.e. goods) allocated from group g that were in mode f and are now in mode t (described in more detail below).

2.2 Defining The Goods in the Market

Within each machine group, we track only the number of cores in each power state. An allocation of some quantity of such cores is ultimately mapped into an assignment of cores on physical machines in post-processing.[6] This avoids the

[4]The value is also scaled by the demand relative to the mean, aligning the one-period bid with the long-term (SLA) statistics.

[5]Our queuing model permits a reduction from the $|G_a \times M_g \times M_g|$ variables to the single variable Q_a. More complex models may require additional dimensions, though in general a significant diminution should be possible.

[6]We currently use a fast but potentially only approximate greedy assignment; however, more sophisticated methods could be used if the identities of machines in a group has importance.

creation of immaterial distinctions that would only complicate winner determination. However, to properly encode the data-center cost model, described in the next section, we need a representation that captures power-state transitions enabling us to account for resultant changes in energy usage, cycle loss and increases in failure rate. Consequently, the $C^{\text{SOLD}}_{g,f,t,a}$ variables capture the number of cores in a given machine group starting in mode f in the last period, transitioning to (the possibly identical) mode t in the current period for a given application a.

Constraints are defined to ensure that an allocation of these goods will be physically implementable; e.g., on present day platforms it is required that all cores on the same physical machine be at the same power level:

$$|\text{CORES}_g| \sum_{f \in M_g} M^{\text{SOLD}}_{g,f,t} = \sum_{f \in M_g} \sum_{a \in A} C^{\text{SOLD}}_{g,f,t,a} + C^{\text{PARTUNSOLD}}_{g,t}$$

$$|\text{CORES}_g| \sum_{f \in M_g} M^{\text{UNSOLD}}_{g,f,t} = \sum_{f \in M_g} C^{\text{UNSOLD}}_{g,f,t} - C^{\text{PARTUNSOLD}}_{g,t}$$

$$\forall\, t \in M_g \;\forall\, g \in G$$

where $|\text{CORES}_g|$ is the number of cores per machine in group g, $C^{\text{UNSOLD}}_{g,f,t}$ are variables counting the unassigned cores, $M^{\text{SOLD}}_{g,f,t}$ and $M^{\text{UNSOLD}}_{g,f,t}$ count sold and unsold machines respectively and $C^{\text{PARTUNSOLD}}_{g,t}$ count the unsold cores on partially sold machines. Additionally, we need to restrain available supply, through the machine counts:

$$|\text{MACHINES}_g| = \sum_{f \in M_g} \sum_{t \in M_g} M^{\text{SOLD}}_{g,f,t} + M^{\text{UNSOLD}}_{g,f,t} \;\forall\, g \in G$$

where $|\text{MACHINES}_g|$ is the number of machines in group g.

2.3 Seller Cost Model

On the supply side of the market, we explicitly model both the hardware and energy costs of running the data center's machines in their various power states. Our model captures the power consumed and performance attained by each machine as a function of the number of active and inactive cores, as measured empirically on an IBM BladeCenter HS21 Server (Figures 5a and 5b). Modern *Dynamic Voltage and Frequency Scaling* (DVFS) enabled machines can have their most efficient state at less than full power: e.g. a maximum of 64 vs. 50 MCycles/Watt with 4 cores active (taking the ratio of the curves in each figure).

We define the energy requirements (i.e. power over the time period) of the active cores as follows (omitting that for idle hardware in the interest of space):

$$E^{\text{SOLD}} = \sum_{g \in G} \sum_{f \in M_g} \sum_{t \in M_g} \text{E}^{\text{MULT}} (\text{E}^{\text{TRANS}}_{g,f,t} + \text{E}^{\text{BASE,ACTIVE}}_{g,\tau,t}) M^{\text{SOLD}}_{g,f,t}$$
$$+ \sum_{g \in G} \sum_{f \in M_g} \sum_{t \in M_g} \sum_{a \in A} \text{E}^{\text{MULT}} \text{E}^{\text{CORE,ACTIVE}}_{g,\tau,t} C^{\text{SOLD}}_{g,f,t,a}$$

where $\text{E}^{\text{TRANS}}_{g,f,t}$ is the energy required to go from power-state f to t, $\text{E}^{\text{BASE,ACTIVE}}_{g,\tau,t}$ is the base power for an active machine, and $\text{E}^{\text{CORE,ACTIVE}}_{g,\tau,t}$ is the incremental energy needed to run a fully loaded core in this power-state. Here E^{MULT} accounts for the typically two- to three-fold increase in energy needed to run power supply units, uninterruptible power supplies, network

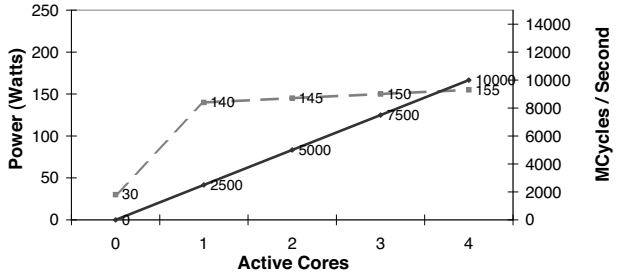


Figure 5a: Power and Speed Under Low Power

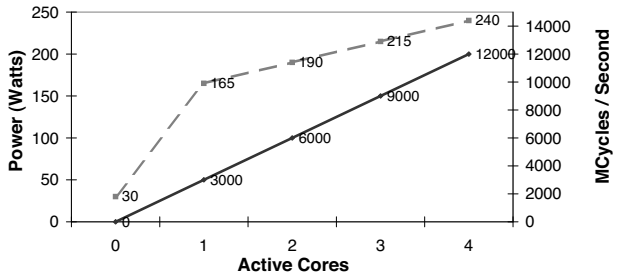


Figure 5b: Power and Speed Under High Power

switches and storage, and most importantly, cooling equipment.

We stipulate the hardware costs for active cores (again omitting the similar expression for idle hardware) as follows:

$$H^{\text{SOLD}} = \sum_{g \in G} \sum_{f \in M_g} \sum_{t \in M_g} (\text{H}^{\text{BASE}}_{g,\tau,g} + \text{H}^{\text{TRANSITION}}_{g,f,t}) M^{\text{SOLD}}_{g,f,t}$$

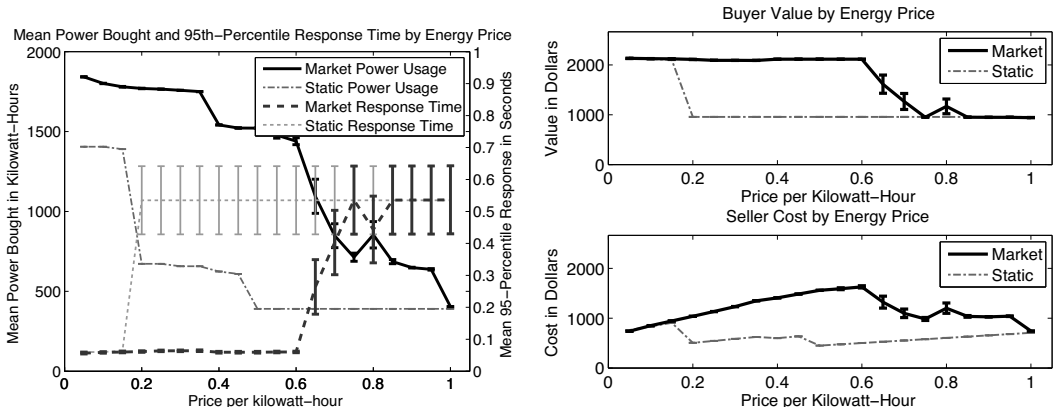
where $\text{H}^{\text{BASE}}_{g,\tau,g}$ is the pro-rated cost for each core, and includes not only the amortized server cost, but also supporting equipment, buildings and personnel; $\text{H}^{\text{TRANSITION}}_{g,f,t}$ accounts for the cost associated with in an increased failure rate upon a state transition due to e.g. spinning up/down hard drives. We expect each of these numbers to be easily obtainable through a principled evaluation of existing business practices and capital investments.

Episodic formulations have a common problem in that they may not bear large transition costs when they create a temporary loss, despite a long-term gain. Consequently, we also find it useful to include a predictor on the sell side that tracks the allocation over previous periods (similarly to the buyer demand prediction) and tweaks the optimizers' view of the power state prior to the new period to better match the predicted demand than the actual power state as selected in the previous period. A full explanation of this component is postponed for a longer version of this paper.

A system administrator might, in addition, wish to specify additional restrictions on the allocation to ensure implementability. Considerable flexibility is possible; some examples include: min/max cores/machines for a given application, min/max energy used in a given machine group or for a given application, and max cores in a given machine group that can be allocated to a given application if a certain number are already allocated to specific alternative application (anti-colocation).

3 Experimental Results

We have evaluated our market-based system in a set of simulation experiments to show both computational tractability and to show effective allocative behavior over a wider range



(a) Energy Used and Response Time **(b)** Buyer Value and Seller Cost

Figure 5: Varying Energy Cost Under Market and Heuristic (Static) Algorithms

of environments. Each experiment has been performed on a 3.2GHz dual-processor dual-core workstation with 8GB of memory and CPLEX 11.1. Each data point is the mean of 10 randomly generated time-dependent demand traces.

Our realistic but synthetic traces are the sum of two sinusoid curves (e.g. 1 day period with 9000 peak transactions/minute plus 1 week period with 36000 peak transactions/minute) and a noise term drawn from a Gaussian with a s.d. equal to 25% of the signal. These match well with real customer traces, where request density is time-dependent and oscillates over both days and weeks.[7] Each transaction is assumed to use 300 MCycles, which is representative of the processing needed to e.g. produce a custom report. Lastly, each allocation period is 10 minutes, which is fast enough to react to dynamic changes in the load but without thrashing.

Because no allocator in the literature has comparable capabilities, we adopt as a benchmark a sophisticated *greedy allocator*, which operates as follows:

1. Put all the machines in their highest efficiency state.
2. Determine the target supply for each application by calculating what is required to produce its ideal response time at its 95th percentile of demand.
3. Allocate cores (from feasible machine groups) to the applications, weighted by the marginal value of supply to each application. If an application's supply of high efficiency cores is exhausted, then instead bump one of the machines supporting it into a higher power state. Stop when either all the targets have been met or all the cores/states have been allocated.
4. Consider each application in turn and trim the allocation until the expected value at the 95th percentile of demand is greater than or equal to the expected cost.
5. Place remaining machines into their lowest power state.

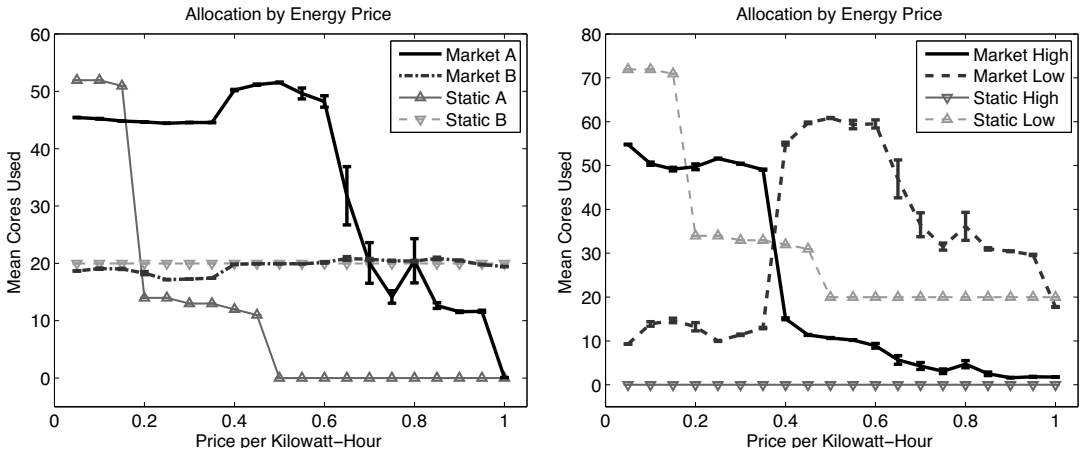
For exposition purposes we consider a simple scenario with two applications (i.e. two customers) and three machine groups (each capable of supporting the first, second and both applications respectively), for a simulated week of time-varying demand. Figure 5a shows the effect of varying the price of energy under both the market and the static allocation algorithm. We can see that, as expected, under both algorithms the energy used falls and consequently the mean response time rises as the price of energy is increased. However, bidding proxies in the market find it profitable to purchase enough energy to maintain a near-optimal response-time until the price finally reaches such a point that such high energy usage can no longer be sustained, and more energy-frugal allocations are chosen. In Figure 5b, we see the impact of the choice of these allocations on buyer (i.e. customer) and seller value, as judged by SLAs and revenue net of cost respectively. The greedy allocation is cheaper to provide because of the static power levels, but also results in significantly lower buyer value over a wide range of prices. The maximum revenue net cost improvement is 137% higher in the market model, though margins become slim when energy is expensive.

It is also important to consider distributive effects to customers in the data-center setting. In this scenario, the 'A' application induces a larger load then 'B', but with a smaller marginal value for cycles. Consequently, as energy prices rise, the static allocator quickly devotes the limited resources that can be afforded to the 'B' allocation, thereby starving the 'A' application, as seen in Figure 6a. The market allocation maintains the allocation for the 'B' application, but also recognizes that some resources can profitably be given to 'A'. This is possible by switching machines between their most efficient modes to conserve energy, and their high-power modes to track spikes in demand. Figure 6b shows that in this setting the static allocator has placed all of the machines in the high efficiency 'Low Power' mode, whereas the market has made use of both modes. When the price for power is low, the most efficient allocation is to maintain quite a few machines in the high power state. However, as the price crosses 40 cents a kWh, there is a phase change and it becomes much more efficient to run mostly in the low-power mode. Beyond about 60 cents per kWh, it becomes impossible to afford the energy needed to maintain a supply sufficient to keep a low response time, and the optimal allocation shrinks.

3.1 Scalability and Complexity

To evaluate the scalability of our MIP formulation we evaluated 10 instances of a scenario with 200 quad-core machines in each of five groups for a total of 1000 machines. We con-

[7] Unlike captured data, robustness to load-structure can be tested.



(a) By Application **(b)** By Power Mode

Figure 6: Allocation By Energy Cost Under Market and Static Algorithms

figured 10 applications, each with a demand for some 2700 transactions a second, to draw upon these resources with each group supporting three of the applications in a ring topology. We restricted the optimizer to no more then 10 minutes of computation per instance, taking advantage of the anytime nature of modern MIP solvers. Including the four instances thus capped, the average solve time was 5.16 minutes, well within the time of a single period. Further, the approximation resulted in only a 0.08% revenue loss when compared to the optimal solution, which would have taken an additional 29 minutes on average for these difficult cases. Thus a single machine is capable of optimizing the usage on 1000 others, giving us a very acceptable .1% overhead factor. For a data center with many more machines, one could then decompose into multiple machine pools, each of size around 1000.

We have also investigated the effect on run-time of the structure of the bipartite graph that defines which application can be supplied by each machine group. For this, we use a scenario with five applications and five machine groups and where supply is set so as to be just sufficient to meet demand. The complexity of the winner-determination problem rises as a sigmoid as we vary the number of edges in the bipartite graph. A graph with 30% of the edges (already highly connected for current data centers) takes only 3.8% of the time needed to clear the market with a complete graph. With 50% connectivity the computation time has risen to 58.8%, and by 60% connectivity the timing has already risen to 86.6%. Further, the increasing complexity is matched by a corresponding decrease in application response time. With 60% of the edges, we are only 8% above the response time of the complete graph.

4 Conclusions

We have established that suitably designed combinatorial markets can find practical application to power-managed resource allocation in corporate data centers. Further, it is possible to inject revenue-based utility functions directly into the present data center business/allocation model without the large changes associated with Utility Computing, a requirement for rapid industry adoption. Such markets facilitate agent information isolation, quantifying the trade-offs of multi-objective optimization, and facilitate the use of combinatorial optimization in a scalable way, provided carefully-designed models are used. Future work should address the economic and game-theoretic considerations that follow from the ability to enable dynamic competition between customers via such a market-based allocation paradigm, and also consider the possibility of richer SLA models and hierarchical allocation paradigms.

References

[Broberg *et al.*, 2007] J. Broberg, S. Venugopal, and R. Buyya. Market-oriented grids and utility computing: The state-of-the-art and future directions. *Grid Computing*, 2007.

[Byde, 2006] A. Byde. A comparison between mechanisms for sequential compute resource auctions. *Proc. AAMAS*, 8(12):1199–1201, 2006.

[Chase *et al.*, 2001] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, and R.P. Doyle. Managing energy and server resources in hosting centers. In *Proc. SOSP*, pages 103–116. ACM New York, NY, USA, 2001.

[Chen *et al.*, 2005] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. In *Proc. SIGMETRICS*, pages 303–314. ACM New York, NY, USA, 2005.

[Cramton *et al.*, 2006] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, Jan 2006.

[Kephart and Das, 2007] J.O. Kephart and R. Das. Achieving Self-Management via Utility Functions. *IEEE Internet Computing*, pages 40–48, 2007.

[Low and Byde, 2006] C. Low and A. Byde. Market-based approaches to utility computing. Tech. Rep. 23, HP, Feb 2006.

[Sandholm, 2007] T. Sandholm. Expressive Commerce and Its Application to Sourcing: How We Conducted $35 Billion of Generalized Combinatorial Auctions. *AI Magazine*, 28(3):45, 2007.

[Steinder *et al.*, 2008] M. Steinder, I. Whalley, J. E. Hanson, and J. O. Kephart. Coordinated management of power usage and runtime performance. In *NOMS*, pages 387–394, 2008.

[Sutherland, 1968] I. E. Sutherland. A futures market in computer time. *Commun. ACM*, 11(6):449–451, 1968.

[Yeo and Buyya, 2006] C.S. Yeo and R. Buyya. A taxonomy of market-based resource management systems for utility-driven cluster computing. *Software, Practice & Experience*, 36(13):1381–1419, 2006.