Enhancing Data Center Sustainability Through Energy Adaptive Computing

Krishna Kant, George Mason University
Muthukumar Murugan, University of Minnesota
David H.C. Du, University of Minnesota

The sustainability concerns of Information Technology (IT) go well beyond energy efficient computing and require techniques for minimizing environmental impact of IT infrastructure over its entire life-cycle. Traditionally, IT infrastructure is overdesigned at all levels from chips to entire data centers and ecosystem; the paradigm explored in this paper is to replace overdesign with rightsizing coupled with smarter control, henceforth referred to as *energy adaptive computing* or EAC. The paper lays out the challenges of EAC in various environments in terms of the adaptation of the workload and the infrastructure to cope with energy and cooling deficiencies. The paper then focuses on implementing EAC in a data center environment, and addresses the problem of simultaneous energy demand and energy supply regulation at multiple levels from servers to the entire data center. The proposed control scheme adapts the assignments of tasks to servers in a way that can cope with the varying energy limitations. The paper also presents some experimental results to show how the scheme can continue to meet *quality of service* (QoS) requirements of tasks under energy limitations.

General Terms: Design, Algorithms, Energy Adaptation

Additional Key Words and Phrases: Energy Efficiency, Energy and Thermal Adaptive Computing, Workload Adaptation

1. INTRODUCTION

Traditionally, computing has emphasized primarily on the performance of both hardware and software. Lately power/thermal issues of computing equipment have forced a consideration of these aspects on par with performance. Power/thermal issues arise at all levels from transistors up to entire ecosystems that involve data centers, clients and the intervening network. At the architectural level, the increasing speeds, smaller feature sizes, and exploding wire widths (and hence resistance) all conspire to make power/thermal issues the main architectural hurdle in sustaining Moore's law. At higher levels, the smaller form factors and more difficult cooling aggravate the problem further. Although power/thermal management is an active area of research, power/thermal issues are still largely approached in the form of opportunistic methods to reduce power consumption or stay within the thermal profile while minimizing any performance impact [B.Heller et al. 2010; Gurumurthi et al. 2003]. Much of this research is focused on reducing the direct energy usage of the data center, whereas from an environment impact perspective one needs to consider the entire life–cycle of energy consumption - that is, the energy consumption in the manufacture, distribution, installation, operation and disposal of the entire data center infrastructure including IT assets, power distribution equipment, and cooling infrastructure [Chang et al. 2010].

With the widespread and deepening power/thermal issues, the systems are literally becoming power and thermal limited, and a new perspective on computing is required. Looking at energy consumption from this larger perspective entails not only low power consumption during operation but also leaner designs and operation using renewable energy as far as possible. Thus, the fundamental paradigm that we consider is to replace the traditional overdesign at all levels with rightsizing coupled with smart control in order to address the inevitable lack of capacity that may arise occasionally. In general, such lack of capacity may apply to any resource, however, we only consider its manifestation in terms of energy/power constraints. Note that power constraints

could relate to both real constraints in the power availability as well as the inability to consume full power due to cooling/thermal limitations. Power consumption limitation indirectly relates to capacity limitation of other resources as well, particularly the dominant ones such as CPU, memory, and secondary storage devices. We call this as *energy adaptive computing* or EAC. [1] The main point of EAC is to consider energy related constraints at all levels and dynamically adapt the computation to it as far as possible. A direct use of locally produced renewable energy could reduce the distribution infrastructure, but must cope with its often variable nature. Thus, better adaptation mechanisms allow for more direct use of renewable energy.

In general, we need adaptation to deal with both supply side and demand side variations. The supply side variations result both from the actual variations in energy supply and the variations as a result of varying the partitioning of available energy among various components. The demand side variations (which themselves drive variability in partitioning) result from variations in workload intensity and characteristics. It has been noted that as the computing moves towards more real–time data mining driven answers to user queries [Chang et al. 2008], the demand side variations could become significantly more severe, thereby further increasing the need for adaptation to available energy.

2. RELATED WORK

Power control techniques [Nedevschi et al. 2008; Isci et al. 2006; Chase et al. 2001] typically focus on harvesting the idle periods or periods of low activity in the workloads and either put the devices in low power modes or reduce the operation bandwidth of the components. Research works have explored the use of power control techniques in various components in the data centers, including CPU [Yang and Orailoglu 2006; Parikh et al. 2004], memory [Kant 2011; Venkatachalam and Franz 2005], network links [Kant 2009; Gupta and Singh 2007] and disks [Gurumurthi et al. 2003; Colarelli and Grunwald 2002].

Heller et al. [B.Heller et al. 2010] propose a dynamic change in the number of active components with changing workload patterns. The goal is to use only a required subset of network components and power down unnecessary components. Moore et al. [Moore et al. 2005] incorporate temperature profiles in data centers to make workload placement decisions. Wang et al. [Wang et al. 2009] propose an algorithm based on optimal control theory to meet with the energy and thermal constraints in chip multi-processors. Their algorithm exploits the availability of per-core DVFS in current day processors and formulates a MIMO model for multi-core processors. Anderson et al. [Andersen et al. 2009] propose a cluster architecture with low-power, cheap processors and flash storage. This architecture performs best in data intensive application scenarios with small sized random accesses.

Virtualization is increasingly being used in high performance server clusters due to its application isolation capabilities and ease of management. Nagarajan et al. [Nagarajan et al. 2007] propose a scheme that migrates virtual machines hosting MPI applications from a fault–prone node to a healthy node proactively. Verma et al. [Verma et al. 2008] investigate the power management in virtualized server clusters hosting HPC applications. They use their experimental results to build a framework for power - aware application placement in virtualized clusters. Their placement strategy takes both CPU usage and working set size into account.

Nathuji and Schwan [Nathuji and Schwan 2007] propose a coordinated power management scheme in distributed environments with virtual machines. They leverage the guest level power management techniques and realize them in real time on the host

[1] Here "energy adaptation" implicitly includes power and thermal adaptation as well.

without violating the guaranteed resource isolation between multiple guests. X. Wang and Y. Wang [Wang and Wang 2010] propose a cluster level coordinated control architecture that aims at providing per-VM performance guarantees and a cluster level power control. Sharma et al. [Sharma et al. 2011] propose a scheme to handle intermittent energy constraints by applying duty cycles to servers. The servers are turned off and on periodically to reduce their energy consumption. Their technique is a purely power driven management scheme and is independent of workload demands.

K. Kant proposes *Energy Adaptive Computing* (EAC) [K.Kant 2009] as a solution to the problem of handling variations in energy availability. Our current work explains the principles and challenges behind Energy Adaptive Computing in a comprehensive manner and also serves as an extension of our previous work [Kant et al. 2011]. In this paper we present a complete design and analysis of a control scheme called (*Willow*) to achieve QoS guarantees with EAC. *Willow* considers power and thermal constraints simultaneously. This paper builds a detailed task model based on QoS requirements of tasks and evaluates *Willow* via more detailed simulations to include the response time as a QoS measure.

3. SUSTAINABILITY AND ENERGY ADAPTIVE COMPUTING

It is well recognized by now that much of the power consumed by a data center is either wasted or used for purposes other than computing. In particular, when not managed properly, up to 50% of the data center power may be used for purposes such as chilling plant operation, compressors, air movement (fans), electrical conversion and distribution, and lighting [Greenberg et al. 2006]. Furthermore, the operational energy is not the only energy involved here. Many of these functions are quite materials and infrastructure heavy and a substantial amount of energy goes into the construction and maintenance of the cooling and power conversion/distribution infrastructures. In fact, even the "raw" ingredients such as water, industrial metals, and construction materials involve considerable hidden energy footprint in the form of making those ingredients available as usable energy.

It follows that from a sustainability perspective, it is not enough to simply minimize operational energy usage or wastage; we need to minimize the energy that goes into the infrastructure as well. Towards this end, it is important to consider data centers that can be operated directly via locally produced renewable energy (wind, solar, geothermal, etc.) with minimal dependence on the power grid or large energy storage systems. Such an approach reduces carbon footprint not only via the use of renewable energy but also by reducing the size and capacity of power storage and power-grid related infrastructure. For example, a lower power draw from the grid would require less heavy–duty power conversion infrastructure and reduce its cost and energy footprint. The down–side of the approach is more variable energy supply and more frequent episodes of inadequate available energy to which the data center needs to adapt dynamically. Although this issue can be addressed via large energy storage capacity, energy storage is currently very expensive and would increase the energy footprint of the infrastructure.

In large data centers, the cooling system not only consumes a substantial percentage of total power (up to 25%) but also requires significant infrastructure in form of chiller plants, compressors, fans, plumbing, etc. Furthermore, chiller plants use a lot of water, much of which simply evaporates. Much of this resource consumption and infrastructure can be done away with by using ambient (or "free") cooling, perhaps supplanted with undersized cooling plants that kick in only when ambient temperature becomes too high. Such an approach requires the energy consumption (and hence the computation) to adapt dynamically to the available cooling ability. The energy available from a renewable source (e.g., solar) may be correlated with the temperature (e.g., more solar

energy on hotter days), and such interactions need to be considered in the adaptation mechanisms.

Yet another sustainability issue is the overdesign and over provisioning that is commonly observed at all levels of computer systems. In particular, the power and cooling infrastructure in servers, chassis, racks, and the entire data center is designed for worst-case scenarios which are either rare or do not even occur in realistic environments. Although data centers are beginning to derate specified power and cooling requirements to address this lack of realism, derating alone is inadequate for two reasons: (a) it still must provide significant safety margin, and (b) derating is used only for sizing up the data center power distribution and cooling capacities, not in server design itself. Instead we argue for much leaner design of all components having to do with power/thermal issues: heat sinks, power supplies, fans, voltage regulators, power supply capacitors, power distribution network, Uninterrupted Power Supply (UPS), air conditioning equipment, etc. This leanness of the infrastructure could be either static (e.g., lower capacity power supplies and heat sinks, smaller disks, DRAM, etc.), or dynamic (e.g., phase shedding power supplies, hardware resources dynamically shared via virtualization). In either case, it is necessary to adapt computations to the limits imposed by power and thermal considerations. We assume that in all cases the design is such that limits are exceeded only occasionally, not routinely.

4. DISTRIBUTED ENERGY ADAPTIVE COMPUTING

It is clear from the above discussion that many advanced techniques for improving energy efficiency of IT infrastructure and making it more sustainable involve the need to dynamically adapt computation to the suitable energy profile. In some cases, this energy profile may be dictated by energy (or power) availability, in other cases the limitation may be a result of thermal/cooling constraints. In many cases, the performance and/or QoS requirements are malleable and can be exploited for energy adaptation. For example, under energy challenged situations, a user may be willing to accept longer response times, lower audio/video quality, less up to date information, and even less accurate results. These aspects have been explored extensively in specific contexts, such as adaptation of mobile clients to intelligently manage battery lifetime [Flinn and Satyanarayanan 2004]. However, complex distributed computing environments provide a variety of opportunities for coordinated adaptation among multiple nodes and at multiple levels. In general, there are three types of distributed energy adaptation scenarios: (a) Cluster computing (or server to server), (b) Client-server, and (c) Peer to Peer (or client to client). These are shown pictorially in Fig. 1 using dashed ovals for the included components and are discussed briefly in the following. Notice that in all cases, the network and the storage infrastructure (not shown) are also important components that we need to consider in the adaptation.

Although we discuss these three scenarios separately, they generally need to be addressed together because of multiple applications and interactions between them. For example, a data center would typically support both client-server and cluster applications simultaneously. Similarly, a client may be simultaneously involved in both peer-to-peer and client-server applications. In this paper however we explore only cluster EAC in detail.

4.1. Cluster EAC

Cluster EAC refers to computational models where the request submitted by a client requires significant computation involving multiple servers before the response can be returned. That is, client involvement in the service is rather minimal, and the energy adaptation primarily concerns the data center infrastructure. In particular, a signifi-

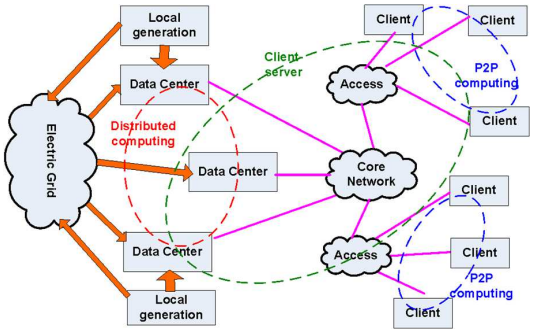


Fig. 1. Illustration of energy adaptation loops

cant portion of the power consumed may go into the storage and data center network and they must be considered in adaptation in addition to the servers themselves.

In cluster EAC, the energy adaptation must happen at multiple levels. For example, the power capping algorithms may allocate a certain power share to each server in a chassis or rack, and the computation must adapt to this limit. In addition, there may be a higher level limit as well – for example, the limit imposed by the power circuits coming into the rack. At the highest level, energy adaptation is required to conform to the power generation (or supply) profile of the energy infrastructure. The limits placed at the lower level generally need to be more dynamic than at higher levels. Translating higher level limits into lower level limits is a challenging problem and requires a dynamic multi–level coordination [Raghavendra et al. 2008]. A related issue is that of energy limitation along the software hierarchy (e.g., service, application, software modules, etc.) and corresponding multi–level adaptation.

In addition to the energy availability, the thermal constraints play a significant role in workload adaptation. Traditionally, CPUs are the only devices that have significant thermal issues to provide both thermal sensors and thermal throttling mechanisms to ensure that the temperature stays within appropriate limits. For example, the T states provided by contemporary CPUs allows introduction of dead cycles periodically in order to let the cores cool. DIMMs are also beginning to be fitted with thermal sensors along with mechanisms to reduce the heat load. With tight enclosures such as blade servers and laptop PCs, ambient cooling, and increasing power consumption, other components (e.g. switching fabrics, interconnects, shared cache, etc.) are also likely to experience thermal issues. In challenging thermal environments, a coordinated thermal management is crucial because the consequences of violating a thermal limit could be quite severe. Also, an over throttling of power to provide a conservative temperature control could have severe performance implications.

Thermal control at the system level is driven by cooling characteristics. For example, it is often observed that all servers in a rack do not receive the same degree of cooling, instead, depending on the location of cooling vents and air movement patterns, certain servers may receive better cooling than others. Most data centers are unlikely to have finer grain mechanisms (e.g., air direction flaps) to even out the cooling effectiveness. Instead, it is much easier to do their thermal management to conform to the cooling profile. So, the simplest scheme is for each server to manage its own thermals based on the prevailing conditions (e.g., on-board temperature measurements). However, such independent controls can lead to unstable or suboptimal control. A coordinated approach such as the one considered in this paper could be used to ensure satisfactory operation while staying within the temperature limits or rather within the power limits dictated by the temperature limit and heat dissipation characteristics.

4.2. Client-Server EAC

Client-server EAC needs to deal with both client-end and server-end adaptation to energy constraints in such a way so that client's QoS expectations are satisfied. A coordinated client-server energy adaptation could even deliver benefits beyond adaptation per se. As the clients become more mobile and demand richer capabilities, the limited battery capacity gets in the way. The client-server EAC can provide better user satisfaction and service by seamlessly compensating for lack of client resources such as remaining battery, remaining disk space, operating in a hot environment, etc. In fact, such techniques can even help slow down client obsolescence and thus enhance the goal of sustainability.

Client-server EAC can be supported by defining client energy states and the QoS that the client is willing to tolerate in different states and during state switching. This information could be communicated to the server side in order to effect appropriate adaptation actions as the client energy state changes. The situation here is similar to but more complex than the contract based adaptation considered in [Petrucci et al. 2009]. The major challenge is to decide optimal strategy to ensure the desired end-to-end QoS without significant overhead or increase in complexity. In a client-server computing, the client adaptation could also be occasionally forced by the server-side energy adaptation. Since server-side adaptation (such as putting the server in deep sleep state and migrating the application to another server) can affect many clients, the server side adaptation decisions become quite complex when interacting with a large number of geographically distributed and heterogeneous clients. Involving the network also in the adaptation further complicates the problem and requires appropriate protocol support.

4.3. Peer to Peer EAC

In a peer to peer setting involving increasingly mobile clients, energy consumption is becoming an important topic. Several recent papers have attempted to characterize energy consumption of P2P content sharing and techniques to improve their energy efficiency [Gurun et al. 2006; Kelényi and Nurminen 2009; Kelenyi and Nurminen 2008]. Energy adaptation in P2P environment is quite different from that in a client-server setting. For simple file-exchange between a pair of peers, it is easy to consider the energy state of the requesting and serving peers and that of their network connections; however, a collective adaptation of a large number of peers can be quite complex. Furthermore, it is important to consider the fundamental P2P issue of get-give in this adaptation. In particular, if a peer is in a power constrained mode, it may be allowed to be more selfish temporarily (i.e., allowed to receive the appropriate low-resolution content that it needs without necessarily supplying content to others). In a more general situation such as Bit Torrent where portions of file may come from different clients, deciding and coordinating content properties and assembling the file becomes more challenging. In particular, it might be desirable to offload some of these functions to another client (that is not in energy constrained mode). In general, addressing these issues requires defining appropriate energy related metrics relative to the content requester, all potential suppliers (or "servers"), transit nodes and the intervening network. A framework that allows minimization of global energy usage while satisfying other local performance and energy requirements can be quite challenging.

5. CHALLENGES IN DISTRIBUTED EAC

5.1. Energy Constraints in Data Centers

In the above, we made little distinction between "energy" and "power"; however, there are subtle differences with respect to both their minimization and adaptation to limits

on them. Energy (or equivalently average power over long periods) can be minimized by deliberately increasing power consumption over short periods. There are many situations where simultaneous energy and power constraints may be required. For example, in a data center, the capacity of power circuits (e.g., chassis or rack circuit capacity, or capacity of individual asset power supply) must be respected while at the same time abiding by energy constraints (or constraint in terms of average power over longer periods)

The real energy or power limitation usually applies only at a rather high level – at lower levels, this limitation must be progressively broken down and applied to subsystems in order to simplify the overall problem. For example, in case of a data center operating in an energy constrained environment, the real limitation may apply only at the level of the entire data center. However, this limitation must be broken down into allocations for the physical hierarchy (e.g., racks, servers, and server components) and also along logical hierarchy (e.g., service, application, and tasks). While such a recursive break-down allows independent management of energy consumption at a finer-grain level, an accurate and stable allocation is essential for proper operation. We address this issue in detail in section 6.

5.2. Estimation and Allocation of Energy

Good energy allocation or partitioning requires an accurate estimation of energy requirements at various layers. Although a direct measurement of energy consumption is straightforward and adequate for making allocations, it only allows reactive (or after the fact) estimation. For example, if additional workload is to be placed on a server, it is necessary to know how much power it will consume *before* the placement decision is made. This is often quite difficult since the energy consumption not only depends on workload and hardware configuration but also on complex interactions between various hardware and software components and power management actions. For example, energy consumed by the CPU depends on the misses in the cache hierarchy, type of instructions executed, and many other micro-architectural details and how they relate to the workload being executed.

A fairly standard method for estimating power is to compute power based on a variety of low–level performance monitoring counters that are available on–chip and becoming increasingly sophisticated. The trick usually is to find a small set of counters that can provide a good estimate of power [Krishnan et al. 2011]. While quite accurate, such a scheme does not have much predictive power since the relationship between high level workload characteristics and performance counters is nontrivial. If multiple tasks are run on the same machine, they can interact in complex ways (e.g., cache working set of one task affected by presence of another task). Consequently, neither the performance nor the power consumption adds up linearly, e.g., the active power for two VMs running together on a server does not equal the sum of active powers of individual VMs on the same server. It may be possible to come up with an estimation method that can account for such interference.

5.3. Mechanisms to Cope with Energy Limitations

When energy availability is restricted, certain applications – particularly those involved in background activities – don't need to run. Others may run less frequently, with fewer resources, or even change their outputs, and still provide acceptable results. For applications that are driven by client requests and must run, the treatment depends on a variety of factors such as Service Level Agreement (SLA) requirements, level of variability in the workload characteristics, latency tolerance, etc. For example, if the workload can tolerate significant latencies and has rather stable characteristics, the optimal mechanism at the server level is to migrate the entire workload to a

smaller set of servers so they can operate without power limitations and shut-down the rest. In this case, a tradeoff is necessary with respect to additional energy savings, SLA requirements, and migration overheads. As the workload becomes more latency sensitive, the latency impact of reconfiguration and power management actions must be taken into account. In particular, if firing up a shut–down server would violate latency and response time related SLA, it is no longer possible to completely shut–down the servers and instead one of the lower latency sleep modes must be used..

Energy (or equivalently average power over long periods) can be minimized by deliberately increasing power consumption over short periods. For example, it may be possible to minimize energy for a given set of tasks to be executed by running these tasks at the highest speed and then putting the machine in a low–power mode. This race-to-halt policy has traditionally been suboptimal because of the possibility of significant voltage reductions at low speeds in the traditional DVFS (dynamic voltage and frequency scaling) techniques by running them at lower frequencies and voltages so as raise the device utilization. However, as voltages approach minimum threshold for reliable operation, the voltage differences between various available speeds are becoming rather small. At the same time, the idle power consumption continues to go up due to increased leakage current. In such a situation, race-to-halt becomes an increasingly attractive scheme.

It is important to note that in case of EAC, often the problem is not inadequate work, but rather inadequate energy to process the incoming work. Obviously, in order to reduce the average power consumption, we need to slow down processing, except that this slowdown is not triggered by idling. Unlike the situation where the goal is to minimize wasted energy, an energy constrained environment requires careful simultaneous management of multiple subsystems in order to make the best use of the available energy. For example, it is necessary to simultaneously power manage CPU, memory and IO adapters of a server in order to ensure that the energy can be delivered where most required. It is apparent that there is scope for considerable further work on how and when to apply various kinds of adaptation mechanisms (e.g., lower resolution, higher latency, control over staleness and/or accuracy, etc.) under various kinds of power/thermal limitation scenarios.

6. ENERGY ADAPTATION IN DATA CENTERS

In this section we explore cluster EAC in detail. In particular, we discuss the design of a control system called *Willow* [Kant et al. 2011] to provide energy and thermal adaptation within a data center. *Willow* provides a hierarchical energy adaptation within data centers in response to both demand and supply side variations. Although a comprehensive energy adaptation scheme could include many facets, the current design of *Willow* is geared towards load consolidation and migration. In Section 6.4 we present the design of a QoS aware scheduling scheme and demonstrate how *Willow* can be used in tandem with the scheduler to achieve the guaranteed QoS for different applications.

6.1. Hierarchical Power Control

Power/energy management is often required at multiple levels including individual devices (CPU cores, memory DIMMs, NICs, etc.), subsystems (e.g., CPU - cache subsystem), systems (e.g., entire servers), and groups of systems (e.g., chassis or racks). In a power limited situation, each level will be expected to have its own power budget, which gets divided up into power budgets for the components at the next level. This brings in extra complexity since one must consider both the demand and supply sides in a coordinated fashion at various levels. In this paper we use such a multilevel power control architecture. One simple such power management model is shown in Figure 2. The data center level power management unit (PMU) is at the level 3. The rack level

PMU is at level 2 and server/switch level PMUs are at level 1. With such a multilevel

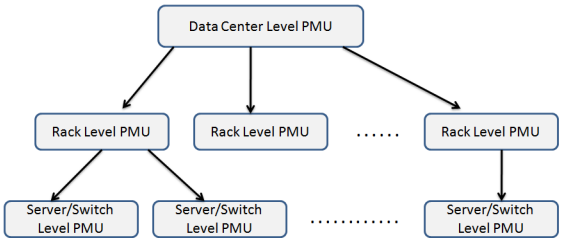


Fig. 2. A simple example of multi-level power control in a datacenter

power management architecture our control scheme attempts to provide the scalability required for handling energy and thermal adaptation in large data centers with minimum impact on the underlying networks.

In the hierarchical power control model that we have assumed, the power budget in every level gets distributed to its children nodes in proportion to their demands. All the leaf nodes are in level 0. The component in each level $l+1$ has configuration information about the children nodes in level l. For example the rack level power manager has to have knowledge of the power and thermal characteristics of the individual components in the rack. The components at level l continuously monitor the demands and utilization levels and report them to level $l+1$. This helps level $l+1$ to continuously adjust the power budgets. Level $l+1$ then directs the components in level l as to what control action needs to be taken. The granularities at which the monitoring of power usage and the allocation adjustments are done are different and are discussed later in Section 6.3.1.

6.2. Energy-Temperature relationship

In the design of our control scheme we limit the power consumption of a device based on its thermal limits as follows.

Let t denote time, $T(t)$ the temperature of the component as a function of time, $P(t)$ power consumption as a function of time, and c_1, c_2 be the appropriate thermal constants. Also, let T_a denote the ambient temperature, i.e., temperature of the medium right outside the component. The component will eventually achieve this temperature if no power is supplied to it. Then the rate of change of temperature is given by

$$dT(t) = [c_1 P(t) + c_2(T(t) - T_a)]dt \tag{1}$$

Being a first-order linear differential equation, this equation has an explicit solution. Let $T(0)$ denote the temperature at time $t = 0$. Then,

$$T(t) = [T_a + [T(0) - T_a]e^{-c_2 t}] + c_1 e^{-c_2 t} \int_0^t P(\tau) e^{c_2 \tau}\, d\tau \tag{2}$$

where the first term relates to cooling and tends to the ambient temperature T_a and the second term relates to heating. Let T_{limit} denote the limit on the temperature and P_{limit} is the limit on power consumption so that the temperature does not exceed T_{limit} during the next adjustment window of Δ_s seconds. It is easy to see that,

$$T(\tau) = T_a + P_{limit} c_1 / c_2 [1 - e^{-c_2 \Delta_s}] + [T(0) - T_a] e^{-c_2 \Delta_s} \tag{3}$$

It can be observed that Equation 2 can be used to predict the value of temperature of the device at the end of the next adjustment window and hence can help in making the migration decisions. We use this relationship to estimate the maximum power consumption that can be allowed on a node so that it does not exceed its thermal limits.

6.3. Supply And Demand Side Coordination

Willow implements a unidirectional hierarchical power control scheme. Migrations of power demands are initiated by the power and thermal constraints introduced as a result of increase in demand at a particular node or decrease in power budget to the node. Simultaneous supply and demand side adaptations are done to match the demands and power budgets of the components.

6.3.1. Time Granularity. The utilization of server resources in a data center varies widely over a large scale. If the nature of the workload fluctuates significantly, it is likely that different resources (e.g., CPU cores, DRAM, memory bus, platform links, CPU core interconnects, I/O adapters, etc.) become bottlenecks at different times; however, for a workload with stable characteristics (but possibly varying intensity) and a well-apportioned server, there is one resource (typically CPU and sometimes network adapter) that becomes the first bottleneck and its utilization can be referred to as server utilization. We assume this is the case for the modeling presented in this paper, since it is extremely difficult to deal with arbitrarily configured servers running workloads that vary not only in intensity but their nature as well. Under these assumptions, the power consumption can be assumed to be a linear monotonic function of the utilization.

Because of varying intensity of the workload, it is important to deal with average utilizations of the server at a suitable time granularity. For convenience the demand side adaptations are discretized with a time granularity of Δ_{Dl}. It is assumed that this time granularity is sufficiently coarse to accommodate accurate power measurement and its presentation, which can be quite slow. Typically, appropriate time granularity at the level of individual servers are of the order of tens of milliseconds or more. Coarser granularities may be required at higher levels (such as rack level).

Even with a suitable choice of Δ_{Dl}, it may be necessary to do further smoothing in order to determine trend in power consumption. Let $CP_{l,i}$ be the power demand of node i at level l. For exponential smoothing with parameter $0 < \alpha < 1$, the smoothed power demand CP' is given by:

$$CP'_{l,i} = \alpha CP_{l,i} + (1 - \alpha) CP'^{old}_{l,i} \tag{4}$$

Note that the considerations in setting up the value of Δ_{Dl} come from the demand side. In contrast, the supply side time constants are typically much larger. Because of the presence of battery backed UPS and other energy storage devices, any temporary deficit in power supply in a data center is integrated out. Hence the supply side time constants are assumed to be $\Delta_{Sl} = \eta_1 \Delta_{Dl}$, where η_1 is an integer > 1. *Willow* also performs workload consolidation when the demand in a server is very low so that some servers can be put in a deep sleep state such as S3 (suspend to memory) or even S4 (suspend to disk). Since the activation/deactivation latency for these sleep modes can be quite high, we use another time constant Δ_{Al} for making consolidation related decisions. We assume $\Delta_{Al} = \eta_2 \Delta_{Dl}$, for some integer η_2 such that $\eta_2 > \eta_1$.

6.3.2. Supply Side Adaptation. As mentioned earlier we ignore the case where the data center operates in a perpetually energy deficient regime. The available power budget of any level $l+1$ is allocated among the nodes in level l proportional to their demands. As

mentioned in Section 6.3.1 the supply side adaptations are done at a time granularity of Δ_{Sl}. Hence the power budget changes are reflected at the end of every Δ_{Sl} time period. Let TP_{l+1}^{old} be the overall power budget at level $l+1$ during the last period. TP_{l+1} is the overall power budget at the end of current period. Δ_{TP}= TP_{l+1}- TP_{l+1}^{old} is the change in overall power budget. If Δ_{TP} is small we can update the values of $\text{TP}_{l,i}$'s rather trivially. However if Δ_{TP} is large we need to reallocate the power budgets of nodes in level l. In doing so we consider both *hard* constraints due to power limitations of devices and *soft* constraints due to available power budgets.

The power and thermal constraints thus necessitate the migration of demand in level l from power deficient nodes to nodes with surplus power budget. Any increase in the overall power budget happens at a higher level and is then reflected in its constituent lower levels. This situation can lead to three subsequent actions.

(1) If there are any under provisioned nodes they are allocated just enough power budget to satisfy their demand.
(2) The available surplus can be harnessed by bringing in additional workload.
(3) If surplus is still available at a node then the surplus budget is allocated to its children nodes proportional to their demand.

6.3.3. Demand Side Adaptation. The demand side adaptation to thermal and energy profiles is done systematically via migrations of the demands. We assume that the fine grained power control in individual nodes is already being done so that any available idle power savings can be harvested. Our focus in this paper is on workload migration strategies to adapt to the energy deficient situations. For specificity we consider only those type of applications in which the demand is driven by user queries and there is minimum or no interaction between servers, (e.g.,) transactional workloads. The applications are hosted by one or more virtual machines (VMs) and the demand is migrated between nodes by migrating these virtual machines. Hence the power consumption is controlled by simply directing the user queries to the appropriate servers hosting them.

We carefully avoid pitfalls like oscillations in decisions by allowing sufficient margins both at the source and the destination to accommodate fluctuations after the migrations are done. The migrations are initiated in a bottom up manner. If the power budget $\text{TP}_{l,i}$ of any component i is too small then some of the workload is migrated to one of its sibling nodes. We call this as local migration. Only when local migrations to sibling nodes is not possible non–local migrations are done.

The migration decisions are made in a distributed manner at each level in the hierarchy starting from the lowermost level. The local demands are first satisfied with the local surpluses and then those demands that are not satisfied locally are passed up the hierarchy to be satisfied non-locally. Now we define a few terms related to the migration decisions.

Power Deficit and Surplus: The power deficit and surplus of a component i at level l are defined as follows.

$$P_{def}(l,i) = [CP'_{l,i} - TP_{l,i}]^+ \tag{5}$$

$$P_{sur}(l,i) = [TP_{l,i} - CP'_{l,i}]^+ \tag{6}$$

where $[]^+$ means if the difference is negative it is considered zero.

If there is no surplus that can satisfy the deficit in a node, the excess demand is simply dropped. In practice this means that some of the applications that are hosted in the node are either shut down completely or run in a degraded operational mode to stay within the power budget.

Power Margin (P_{min}): The minimum amount of surplus that has to be present after a migration in both the source and target nodes of the migration. This helps in mitigating the effects of fluctuations in the demands.

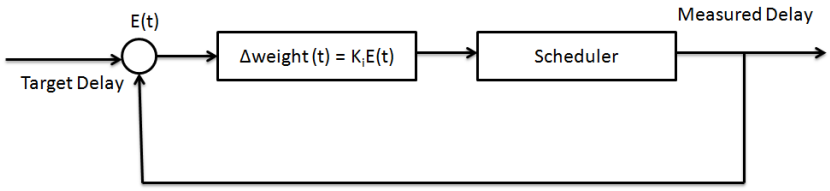
Migration Cost: The migration cost is a measure of the amount of work done in the source and target nodes of the migrations as well as in the switches involved in the migrations. This cost is added as a temporary power demand to the nodes involved.

A migration is done if and only if the source and target nodes can have a surplus of at least P_{min}. Also migrations are done at the application level and hence the demand is not split between multiple nodes. Finally *Willow* also does resource consolidation to save power whenever possible. When the utilization in a node is really small the demand from that node is migrated away from it and the node is deactivated.

The matching of power deficits to surpluses is done by a variable sized bin packing algorithm called FFDLR [Friesen and Langston 1986] solves a bin packing problem of size n in time $O(n \log n)$. The optimality bound guaranteed for the solution is (3/2) OPT +1 where OPT is the solution given by an optimal bin packing strategy.

6.4. QoS Aware Scheduler

Traditional scheduling algorithms like round robin and priority based scheduling are not QoS aware and do not have any feedback mechanism to guarantee the QoS requirements of jobs. A few algorithms that attempt to guarantee some level of QoS do so by mechanisms like priority treatment and admission control. Our objective in this work is to allow for energy adaptation while respecting QoS needs of various applications to the maximum extent possible. In this regard we implemented a QoS aware scheduling algorithm in the nodes as shown in Figure 3. The scheduler uses an integral controller to adjust the weights of the applications at regular intervals based on the delay violations of the applications. The applications are allocated CPU shares proportional to their weights. Applications with higher delay violations get more CPU share. It is well known from classic queuing theory [Kant 1992] that in an asymptotic sense, as $U \to 1$ the wait time of jobs is directly proportional to $1/(1-U)$ where U is the overall queue utilization. Hence the integral gain for the controller is calculated as $(1-U)$. Using the overall utilization as the integral gain also avoids oscillations and keeps the system stable.



E(t) = Error observed over a period of Δt
K_i = Integral gain (1 – U)
U = Utilization
Δweight (t) = weight (t) – weight (t – 1)
weight (t) = weight at time t proportional to which CPU time is allocated

Fig. 3. QoS Aware Scheduler

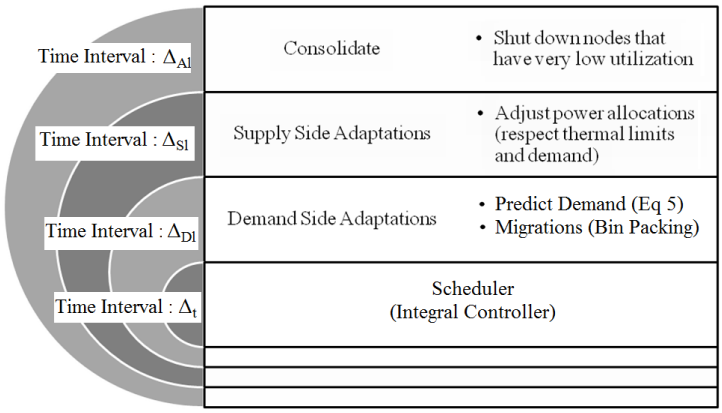


Fig. 4. Various adaptations done in *Willow* and the different time granularities

The error in delay $E(t)$ for each application is the difference between the delay bound and the measured delay. At the end of each sample interval t (of size Δt), the new weights are calculated as follows.

$$weight(t) = weight(t-1) + K_i * E(t) \tag{7}$$

Then the CPU shares are allocated to the applications proportional to their weights.

Figure 4 shows an overall picture of the different adaptation mechanisms that are done in *Willow* at different time granularities. The scheduler works in the individual nodes at the smallest time granularity. At the next higher time granularity the demand side adaptations are done that include migration of deficits to surplus nodes. At the next higher granularity the supply side adaptations such as allocating power budgets at different levels takes place. At the largest time granularity consolidation related decisions are made that include shutting down of nodes with very low utilization.

7. EXPERIMENTAL EVALUATION

7.1. Assumptions and QoS model

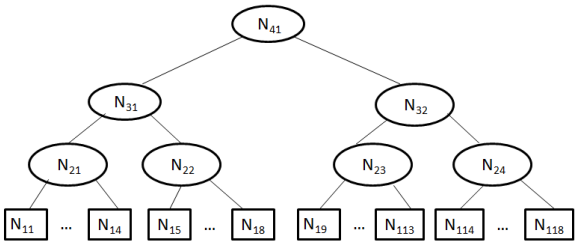


Fig. 5. Configuration used in simulation

We built a simulator in Java for evaluating the ability of our control scheme to cater to the QoS requirements of tasks when there are energy variations. The Java simulator can be configured to simulate any number of nodes and levels in the hierarchy. For our evaluations we used the configuration shown in Figure 5. There are 18 nodes and 3 types of applications. The application types and their SLA requirements are shown

Table I. Utilization vs power consumption

Application Type	SLA Requirement	Mean Runtime
Type I	Average Delay $\leq 120ms$, cannot be migrated	$10\,ms$
Type II	Average Delay $\leq 180ms$, can be migrated	$15\,ms$
Type III	Average Delay $\leq 200ms$, can be migrated	$20\,ms$

in Table I. For simplicity we assume that each application is hosted in a VM and can be run on any of the 18 nodes. To begin with, each node is assigned a random mix of applications. The static power consumption of nodes is assumed to be 20% of the maximum power limit $(450W)$. There is a fixed power cost associated with migrations. In the configuration that we use for our experiments, we assume a single SAN storage that can be accessed by all nodes in the data center. Storage migration is done when the VM disk files have to be migrated across shared storage arrays due to shortage of storage capacity and is usually dealt with separately (eg. Storage VMotion [VMWare vSphere]) so as to reduce the delays involved. Since we deal with compute intensive applications in our experiments, we assume a single shared storage domain is large enough to support the applications and we do not account for delays involving data migrations.

Since our experimental platform consists of multiple VMs with independent traffic patterns, initially we ran our experiments for the case where the traffic to each individual VM was a Poisson process. In order to test our proposed scheme with real world traces, we used the Soccer World Cup 98 [M. Arlitt and T. Jin] traces for our evaluation. In this paper, we present only the results with the World Cup 98 trace. The trace dataset consists of all the requests made to the 1998 Soccer World Cup Web site during April to July, 1998. Since the trace data was collected from multiple physical servers, we had to process the traces before we could use them in our virtualized environment. We used traces collected on different days for different VMs. We scaled the arrival rates of queries to achieve the target utilization levels. We assume that the queries in the traces belonged to the three classes of applications as shown in Table I. The service times for queries for each application class is different and the energy consumed by a query is assumed to be proportional to its runtime. The time constant multipliers for discrete time control η_1 and η_2 in Section 6.3.1 are assumed to be 4 and 7 respectively. Unless specified otherwise the ambient temperature of the nodes was assumed to be $25°$C. Also the thermal limit of the servers and switches is assumed to be 70°C. The thermal constants in Equation 1 were determined to be $c_1 = 0.2$ $c_2 = -0.008$ from the experiments as described in [Kant et al. 2011]. We use these values for our simulations as well.

We measure the utilization of a VM based on CPU time spent by the VM in servicing the requests. However the actual utilization may be higher. For instance the actual utilization of a task may be increased due to CPU stalls that are caused by memory contention from other tasks or context switches between multiple VMs. In our simulations we include a factor called the interference penalty for the utilization and power/energy calculations. Basically the idea is that when n tasks are running on a server, the utilization of each task is increased due to the interference from the other $(n-1)$ tasks. Hence the actual utilization of an application is calculated as follows.

$U_i = U_i + \alpha \sum_j U_j$, $\forall j \in \{1, 2, ...n\} - \{i\}$

$U_{node} = \min [1.0, \sum_{i=1}^{n} U_i]$

where n is the total number of applications in the node

U_i is the utilization of i^{th} application , $i \in \{1, 2, ...n\}$.

U_{node} is the actual utilization of the node.

We then calculate the average power consumed in the node based on the actual average utilization of the node. We conducted a few experiments on a Dell machine running VMWare ESX server to determine the value of α. We varied the number of VMs and their utilization levels and compared the sum of their utilizations with the actual utilization reported by the ESX server. We then calculated the the value of α using simple first order linear regression. The value of α was found to be 0.01 Note that the workload that we used in our experiments was totally CPU bound. For other workloads that involve memory or network contention, the interference penalty might be higher.

7.2. Simulation Results

In order to evaluate the performance of the integral controller we placed all 3 types of applications on a single node and the incoming traffic to each of the applications was assumed to be a Poisson process. Figure 6 shows the delays of queries normalized to their delay bounds at a utilization level of 60%. Any value of delay greater than 1 implies an SLA violation. A QoS ignorant scheduler simply serves queries as they arrive. This leads to increased delay violations for queries. For instance, when there are too many Type I queries that have a smaller delay bound waiting in the queue, a QoS aware scheduler will make Type III queries to wait a little longer in the queue since they can tolerate much longer delays. On the other hand a QoS ignorant scheduler continues to favor both Type I and Type III queries equally and hence leading to delay violations for Type I queries. It can be seen from Figure 6 that almost during the entire time, the feedback based integral controller successfully keeps the delays of all 3 types of queries below the bounds specified by their SLA requirements.

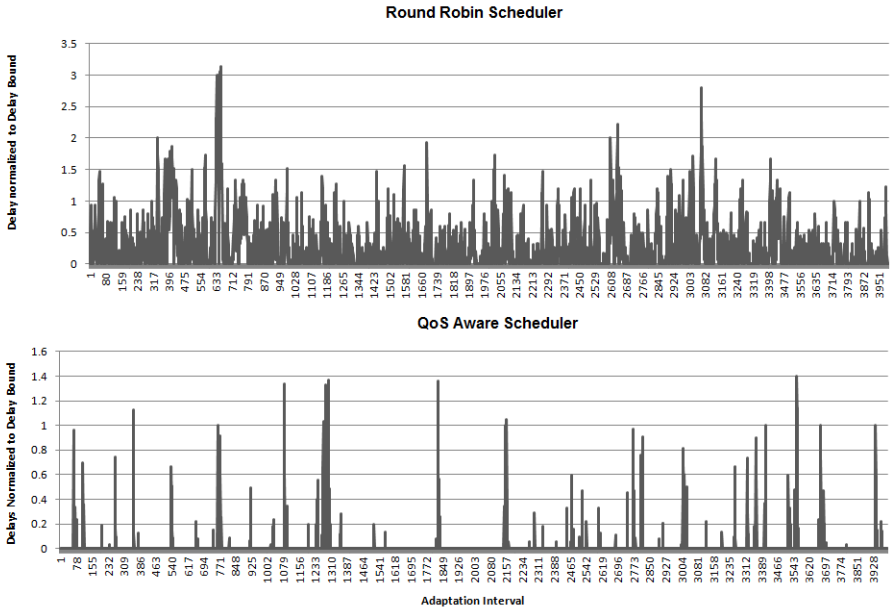


Fig. 6. Time series of average delays of queries normalized to delay bounds with simple Round Robin and QoS Aware Scheduling

We use the response time as a metric to quantify the impact of EAC in the presence of variations in energy. The response time includes the time that the queries wait in

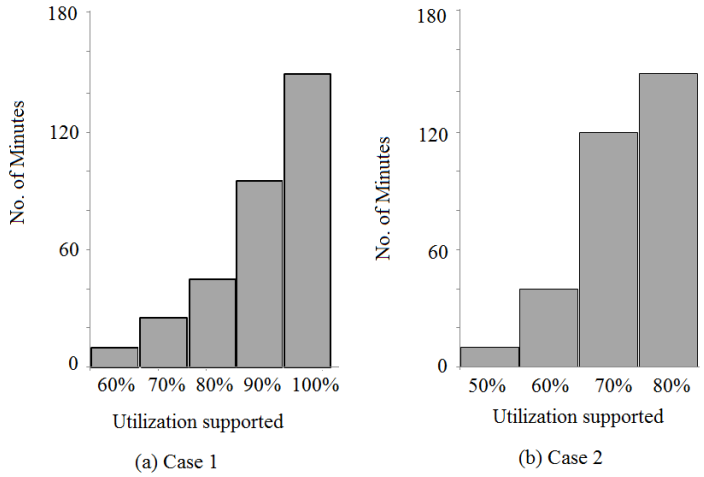


Fig. 7. Power Supply Profiles Used - Histogram of power supply values and the utilization levels supported

the queue to be scheduled and the run time of the queries. We show that the response times are improved when adaptations to the energy variations are done in *Willow*. The significance of Willow is realized the most when the devices are operating at the edge - that is when the power budgets are just enough to meet the aggregate demand in the data center and there is no over provisioning. To demonstrate this, we tested the performance of Willow with two different cases of power budgets. Let P_U be the power required to support the data center operations when the average utilization of the servers is $U\%$. Figure 7 shows the histogram of the total available power budget values during the simulation of 350 minutes and the number of minutes for which the particular power budget value was available. The first case (Case 1) is when the total available power budget varies between P_{100} and P_{60}. The second case (Case 2) is when the total power budget varies between t P_{80} and P_{50}.

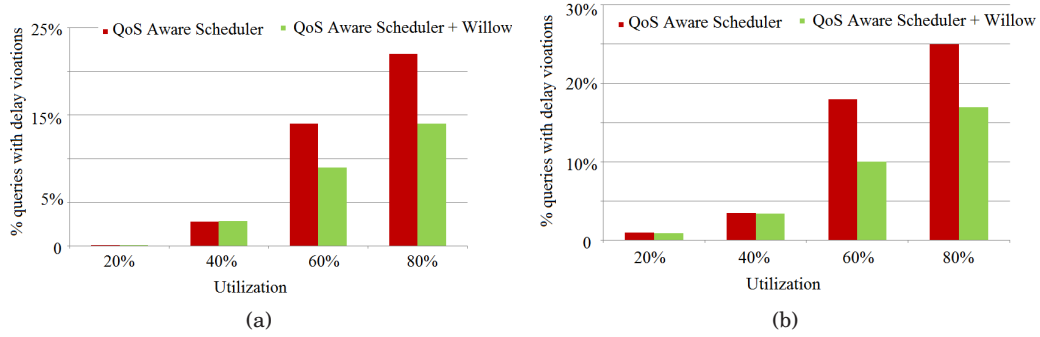


Fig. 8. % of queries with delay violations when total power budget varies around the power supply value required to support (a) 80% utilization (b) 100% utilization

Figure 8 (a) compares the percentage of queries with delay violations in Case 1 when the QoS aware scheduler alone is used and when the scheduler is used in combination with *Willow*. We see that at low utilizations the performance of Willow is not significantly better than when the QoS aware scheduler alone is used. However at high

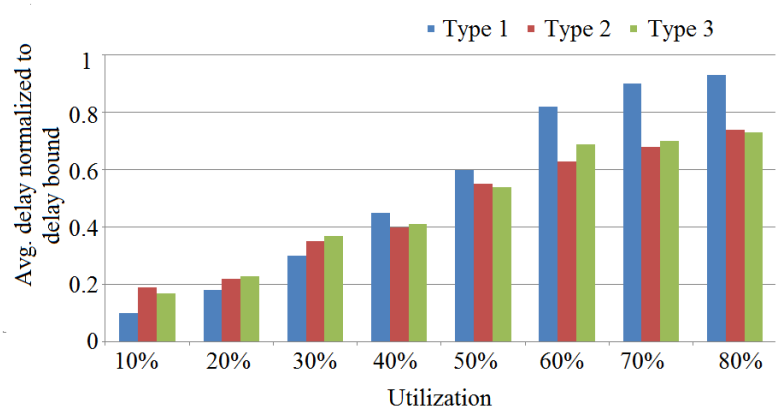


Fig. 9. Average delay values normalized to delay bounds

utilizations *Willow* performs better than the case when the QoS aware scheduler alone is used since there are no adaptation related migrations. Figure 8 (b) shows the percentage of queries with delay violations in Case 2. It can be seen that the benefits of Willow are very significant in Case 2 as compared to Case 1, especially at moderate to high utilization levels. Figure 8 shows that an efficient QoS aware scheduler alone cannot do any good in the presence of energy variations. *Willow* improves the possibility of meeting the QoS requirements significantly with the help of systematic migrations.

Figure 9 shows the average delay of different application types normalized to their delay bounds at different utilization levels for power profiles as in Case 2. As expected the average delays increase with increase in utilization due to increased wait times. It can be seen that at higher utilizations, the average normalized delay is higher for Type I queries. The reason is two fold. Firstly, according to the SLA requirements of Type I queries that we assume in Table I, they cannot be migrated. This reduces the flexibility for these applications when the available power budget in the server becomes very low. Secondly, the absolute values of the delay bound is much smaller than the other two types and hence the average delay when normalized to the delay bound is naturally higher. Since the delay requirements of Type II and Type III are almost the same, they are equally impacted. It can be seen that even at low utilizations there are queries that have delay violations. This is because when the available energy drops, the demand side adaptations are done only after Δ_{Dl} in *Willow*. During that time if the estimated demand is less than the actual demand some queries have to wait longer to be scheduled. Moreover the utilization levels are just an average (calculated at a much higher granularity) and there can be periods of congestion even at those low utilization levels. This is especially handled very poorly when there are no migrations.

Figures 10 and 11 demonstrate the thermal adaptability of *Willow*. We set the ambient temperatures of servers $1 - 4$ at $45°$C and the other servers at $25°$C. In order to enforce power budgets we reduce the available CPU shares proportional to their allocated power budgets. Figure 10 shows the average delays in the servers when the first four servers are at a higher temperature than others. To avoid clutter we show the average response times only for the first 9 servers. As expected in all servers the response time increases with increase in utilization. However at low to moderately high utilization levels ($\leq 60\%$) it can be seen that the average response time is lower in high–temperature severs. This is because *Willow* is able to migrate most of the load

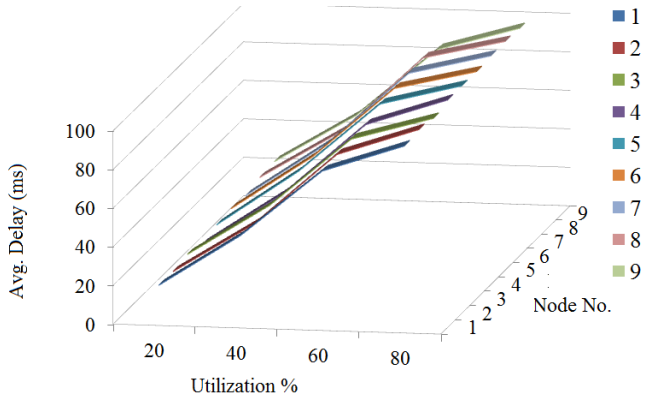


Fig. 10. Avg. delay values for Type I queries when servers $1-4$ are at a higher ambient temperature

away from high temperature servers. Hence the waiting time for queries is reduced which in turn improves the response times in the high–temperature servers.

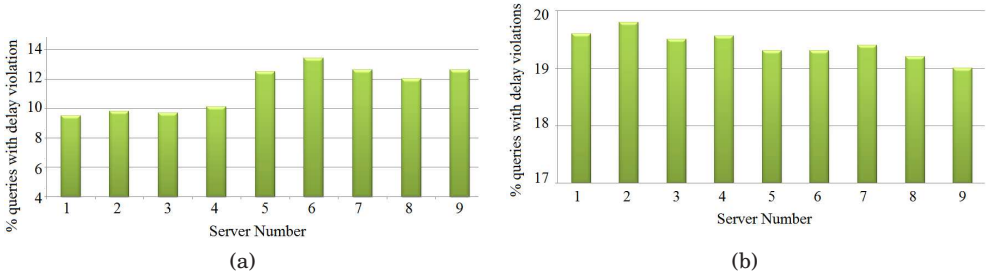


Fig. 11. % of queries with delay violations when servers 1–4 are at high temperature at (a) 60% utilization (b) 80% utilization

Figure 11 (a) shows the percentage of queries with delay violations when the ambient temperature of servers 1–4 is $45°$C at 60% utilization. As explained before, at a moderately high utilization level(60%), Willow migrates applications away from high temperature servers and hence they run at lower utilizations. This in turn reduces the number of queries with delay violations. However at high utilization (80%) the high temperature servers have no choice but to increase the delay of queries because no migrations can be done. This is shown in Figure 11 (b) .

8. CONCLUSION AND FUTURE WORK

In this paper we have discussed the challenges involved in adapting to the energy and thermal profiles in a data center. EAC puts power/thermal controls at the heart of distributed computing. We discussed how EAC can make IT more sustainable and elaborated on three different types of EAC scenarios. We have presented the design of *Willow*, a simple control scheme for energy and thermal adaptive computing. A major goal of this work is to inspire right-sizing the otherwise over designed infrastructure in terms of power and ensure the possibility of addressing the ensuing challenges via smarter control.

In this work, we have tested the performance of *Willow* in the case of transactional workloads in simulations for specificity. In our experiments we used applications that

are CPU intensive. A more complete control system for cluster EAC must be able to measure power consumption and temperature of every component in the server including memory, NIC, hard disks etc. and make fine grained control decisions. In the future, we plan to consider more complex EAC scenarios including the problems of dividing up available energy between servers, storage and network such that the application progress can be optimized. We would also like to analyze the adaptation techniques for cluster EAC under more complex workloads where there is excessive IPC traffic among the servers in addition to requests from clients. A real time implementation might need to consider the migrations that are caused as a result of resource constraints as well. In order to do a holistic power control, the adaptation must consider the energy consumed by cooling infrastructure as well in the adaptation. Dealing with data migrations that involve significant delays is a future direction that we intend to pursue. Another interesting area that we wish to explore is the possibility of predicting the variations in energy profiles and doing some computations in advance in anticipation of energy shortages.

REFERENCES

ANDERSEN, D. G., FRANKLIN, J., KAMINSKY, M., PHANISHAYEE, A., TAN, L., AND VASUDEVAN, V. 2009. FAWN: A Fast Array of Wimpy Nodes. In *SOSP '09*. ACM, 1–14.

B.HELLER, S.SEETHARAMAN, P.MAHADEVAN, Y.YIAKOUMIS, P.SHARMA, S.BANERJEE, AND MCKEOWN, N. 2010. ElasticTree: Saving Energy in Data Center Networks. In *NSDI'10: Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation*.

CHANG, F., DEAN, J., GHEMAWAT, S., HSIEH, W. C., WALLACH, D. A., BURROWS, M., CHANDRA, T., FIKES, A., AND GRUBER, R. E. 2008. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst. 26*.

CHANG, J., MEZA, J., RANGANATHAN, P., BASH, C., AND SHAH, A. 2010. Green server design: Beyond operational energy to sustainability. In *Proceedings of the 2010 international conference on Power aware computing and systems*. HotPower'10.

CHASE, J. S., ANDERSON, D. C., THAKAR, P. N., VAHDAT, A. M., AND DOYLE, R. P. 2001. Managing energy and server resources in hosting centers. *SIGOPS Oper. Syst. Rev. 35,* 5, 103–116.

COLARELLI, D. AND GRUNWALD, D. 2002. Massive arrays of idle disks for storage archives. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1–11.

FLINN, J. AND SATYANARAYANAN, M. 2004. Managing battery lifetime with energy-aware adaptation. *ACM Trans. Comput. Syst. 22*.

FRIESEN, D. K. AND LANGSTON, M. A. 1986. Variable sized bin packing. *SIAM J. Comput. 15,* 1, 222–230.

GREENBERG, S., MILLS, E., TSCHUDI, B., RUMSEY, P., AND MYATT, B. 2006. Best Practices for Data Centers: Lessons Learned from Benchmarking 22 Data Centers.

GUPTA, M. AND SINGH, S. 2007. Dynamic Ethernet Link Shutdown for Energy Conservation on Ethernet Links. In *ICC*. 6156–6161.

GURUMURTHI, S., SIVASUBRAMANIAM, A., KANDEMIR, M., AND FRANKE, H. 2003. DRPM: dynamic speed control for power management in server class disks. *SIGARCH Comput. Archit. News 31,* 2, 169–181.

GURUN, S., NAGPURKAR, P., AND ZHAO, B. Y. 2006. Energy consumption and conservation in mobile peer-to-peer systems. In *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*. MobiShare '06. 18–23.

ISCI, C., BUYUKTOSUNOGLU, A., CHER, C.-Y., BOSE, P., AND MARTONOSI, M. 2006. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget. In *39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-39 2006), 9-13 December 2006, Orlando, Florida, USA*.

KANT, K. 1992. *Introduction to computer system performance evaluation*. McGraw-Hill.

KANT, K. 2009. Power control of high speed network interconnects in data centers. In *INFOCOM'09: Proceedings of the 28th IEEE international conference on Computer Communications Workshops*. 145–150.

KANT, K. 2011. A control scheme for batching dram requests to improve power efficiency. In *SIGMETRICS*. 139–140.

KANT, K., MURUGAN, M., AND H.C.DU, D. 2011. Willow: A Control System for Energy and Thermal Adaptive Computing. In *Proceedings of 25th IEEE International Parallel & Distributed Processing Symposium, IPDPS '11*.

KELENYI, I. AND NURMINEN, J. 2008. Energy Aspects of Peer Cooperation Measurements with a Mobile DHT System. In *Communications Workshops, 2008. ICC Workshops '08. IEEE International Conference on*. 164 –168.

KELÉNYI, I. AND NURMINEN, J. K. 2009. Bursty content sharing mechanism for energy-limited mobile devices. In *Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. PM2HW2N '09.

K.KANT. 2009. Challenges in Distributed Energy Adaptive Computing. In *Proceedings of ACM HotMetrics*.

KRISHNAN, B., AMUR, H., GAVRILOVSKA, A., AND SCHWAN, K. 2011. VM power metering: feasibility and challenges. *SIGMETRICS Perform. Eval. Rev. 38*, 56–60.

M. ARLITT AND T. JIN. 1998 World Cup Web Site Access Logs. `http://www.acm.org/sigcomm/ITA/`.

MOORE, J., CHASE, J., RANGANATHAN, P., AND SHARMA, R. 2005. Making scheduling "cool": temperature-aware workload placement in data centers. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*. 5–5.

NAGARAJAN, A. B., MUELLER, F., ENGELMANN, C., AND SCOTT, S. L. 2007. Proactive fault tolerance for hpc with xen virtualization. In *Proceedings of the 21st annual international conference on Supercomputing*. ICS '07. 23–32.

NATHUJI, R. AND SCHWAN, K. 2007. VirtualPower: Coordinated power management in virtualized enterprise systems. In *SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*. 265–278.

NEDEVSCHI, S., POPA, L., IANNACCONE, G., RATNASAMY, S., AND WETHERALL, D. 2008. Reducing network energy consumption via sleeping and rate-adaptation. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*.

PARIKH, D., SKADRON, K., ZHANG, Y., AND STAN, M. 2004. Power-Aware Branch Prediction: Characterization and Design. *IEEE Trans. Comput. 53,* 2, 168–186.

PETRUCCI, V., LOQUES, O., AND MOSSÉ, D. 2009. A framework for dynamic adaptation of power-aware server clusters. In *Proceedings of the 2009 ACM symposium on Applied Computing*. SAC '09.

RAGHAVENDRA, R., RANGANATHAN, P., TALWAR, V., WANG, Z., AND ZHU, X. 2008. No "power" struggles: coordinated multi-level power management for the data center. *SIGARCH Comput. Archit. News 36*.

SHARMA, N., BARKER, S., IRWIN, D., AND SHENOY, P. 2011. Blink: Managing server clusters on intermittent power. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems (ASPLOS)*.

VENKATACHALAM, V. AND FRANZ, M. 2005. Power reduction techniques for microprocessor systems. *ACM Comput. Surv. 37,* 3, 195–237.

VERMA, A., AHUJA, P., AND NEOGI, A. 2008. Power-aware dynamic placement of HPC applications. In *Proceedings of the 22nd annual international conference on Supercomputing*. ICS '08.

VMWARE VSPHERE. `http://www.vmware.com/products/storage-vmotion/overview.html`.

WANG, X. AND WANG, Y. 2010. Coordinating Power Control and Performance Management for Virtualized Server Clusters. *IEEE Transactions on Parallel and Distributed Systems 99*.

WANG, Y., MA, K., AND WANG, X. 2009. Temperature-constrained power control for chip multiprocessors with online model estimation. *SIGARCH Comput. Archit. News 37,* 3, 314–324.

YANG, C. AND ORAILOGLU, A. 2006. Power efficient branch prediction through early identification of branch addresses. In *CASES '06: Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*.