Distributed Energy Adaptive Computing

Krishna Kant
Intel Corp.

Abstract—**Fueled by burgeoning online services, power and thermal issues are becoming crucial both in terms of utility costs and environmental impact. In this paper, we motivate an approach that puts power/thermal issues at the heart of distributed computing, and strives to dynamically optimize energy use, heat dissipation and energy supply in order to deliver an acceptable user experience. The paper discusses how such an approach can enhance the sustainability of computing. It also compares and contrasts two basic techniques for energy adaptation in servers, namely power capping and use of sleep modes.**

I. Introduction

Traditionally, computing has been all about performance – new hardware is always touted for its higher performance or performance per unit cost and new algorithms are always about improving performance over the old ones. However, energy and sustainability issues are now becoming front and center, and a rethink of the traditional approach is in order. Although mobile clients have always been energy (and hence performance) constrained, this has been traditionally handled by the "thin client" model where the servers do bulk of the work. From the energy perspective, the thin client model is based on two assumptions – that servers have unlimited energy at their disposal and clients interact primarily via servers – both of which are increasingly unlikely to be true.

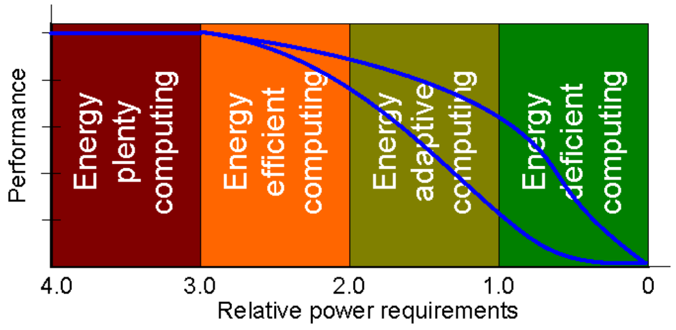

Fig 1. Illustration of energy adaptation loops

With clients going mobile in all sorts of form factors (laptops, netbooks, PDAs, smartphones, etc.), a substantial rise in high-quality multimedia content, and significant penetration of peer-to-peer (P2P) services and social networks, all require a varied set of capabilities from the clients. The unscalable power consumption of servers, desktops and mobile devices has been recognized for over a decade and has led to very agressive improvement in power efficiency at a variety of levels – from low-power HW design to aggressive use of available power modes to intelligent load and activity management [3]. These efforts are expected to continue in the foreseeable future. Yet, the sheer increase in the computing base and environmental concerns require that we move beyond *energy efficient computing* to what we call *energy adaptive computing* or EAC. The main point of EAC is to consider energy availability as the primary resource that determines how much computation we can afford. We then need to come up with appropriate adaptation mechanisms which may range from simply letting the processing run slower to migration to another node to changing the nature of work so that it is less energy intensive.

EAC has two additional objectives beyond localized techniques for making devices more energy efficient: (a) Enabling use of locally produced, renewable energy, and providing for adaption to its intermittent and limited nature, and (b) A global view of energy management that accounts for inter-dependencies between various agents involved in providing a service. In this paper, we elaborate on the essential aspects of energy adaptive computing, discuss various challenges and show how EAC can have positive impact on not only energy efficiency but also sustainability.

Figure 1 shows a rough schematic to place EAC with respect to the energy consumption plotted on the x-axis. This energy consumption is shown as a ratio of "useful" energy (i.e., energy actually required by the computation) and the y-axis shows performance relative to the maximum achievable performance. The two curves show the upper and lower bounds on the relative performance achievable under a given energy constraint. The diagram is merely illustrative and hence the precise values are unimportant. Without an effective power management, a data center can waste 75% or more of the power it consumes. An aggressive use of the available power management capabilities such as the use of sleep modes, dynamic voltage and frequency control, power/thermal aware scheduling, etc. can reduce this wastage significantly [3]. This is the energy efficient computing (EEC) regime in the diagram. Although there is no inherent reason to do so, most such techniques are designed and calibrated from a performance perspective, i.e., reduce power consumption subject to only a small hit in performance. EAC goes a step beyond and attempts to achieve the best performance for energy constraints that may be externally imposed. The main challenge of EAC is to "pull the curves up" as much as possible via control schemes that not only manipulate the power management knobs but also alter the workload to cope with the energy constraints. Finally, energy deficient computing refers to techniques such as RAZOR that allow operating voltages to dip close to threshold voltages and thus move into the realm where we either give up guaranteed accuracy or use techniques to deal with unreliable computation. [1].

II. Towards Sustainable Computing

The ongoing rapid expansion in online services is expected to require more and bigger data centers to sustain them. In particular, in spite of aggressive efforts at power consumption reduction of computing systems, we are likely to see 2-4X increase in overall power consumption of servers, clients and the intervening network in the next decade. This increase is primarily related to the significantly higher number of server and client devices in operation and their greater capabilities. Thus, the carbon footprint of information technology (IT) is expect to expand significantly. In this section, we argue that it is possible to arrest this by evolving the data center infrastructure towards using renewable energy.

It is well recognized by now that much of the power consumed by a data center is actually wasted. In particular, roughly 50% of the data center power goes into non-IT equipment with a rough breakup as follows: cooling (2̃5%), air movement (1̃2%), electrical conversion and distribution (1̃0%) and lighting (3̃%). Many of these functions not only consume operational energy but have massive energy needs when considered in the entire life-cycle (cradle to grave) context. For example, electrical conversion and distribution involves large amounts of materials in form of electric substation, UPS (uninterruptible power supply), diesel generators, step-down transformers, cabling, etc. Similarly, cooling infrastructure involves large amounts of materials and water usage.

From a sustainability perspective, it is crucial to examine energy needs during the entire life-cycle of data center assets rather than just the operational energy. In other words, the substantial energy required build (and eventually recycle) the infrastructure or to supply other resources such as water also needs to be considered in the mix. Thus, energy efficiency must consider minimization of not just the operational energy but the entire *energy footprint*.

Towards this end, we consider data center that can be operated directly via locally produced renewable energy (wind, solar, etc.) with minimal dependence on the power grid or large energy storage systems.[1] Such an approach requires smaller current capacity substation to draw grid power and smaller UPS and thereby reduces the energy footprint of the data center. This is different from the traditional approach where the renewable energy, even if available locally, is fed to the power grid, which then drives the data center. The approach advocated here attempts to reduce energy footprint at the cost of two additional capabilities: (a) built-in intelligence at multiple levels to adapt to variable and less reliable energy supply, and (b) exploitation of networking infrastructure to migrate data and/or computation as needed. Surely, the greater reliance on networking infrastructure implies its own energy footprint; however, given that such an infrastructure already exists, the additional energy footprint to exploit locally available energy is expected to be small. In a sense, the approach exploits IT to provide more flexibility in energy generation and usage and thereby reduce the energy footprint.

The energy footprint of data center cooling infrastructure can be reduced by either opting for ambient cooling [2] or undersized cooling plants. This again requires intelligence at multiple levels to adapt to inadequate or less reliable cooling. Furthermore, simultaneous use of renewable energy and ambient cooling may itself have interesting couplings (e.g., higher solar energy production coupled with higher cooling requirements) that need to be considered.

Let us now consider energy footprint issues at the level of racks and servers. It was mentioned above that only 50% of the power supplied to a traditional data center goes into the racks. Of this, at least 50% is again wasted due to power supply and voltage regulator (VR) inefficiencies, cooling fans, semiconductor leakage, etc. Reducing this wastage is primarily the realm of energy efficient computing (EEC) and includes better materials, architecture, design and power management. However, there is another energy footprint issue that is often overlooked and is important from EAC perspective. The issue is the significant overdesign not only in terms of computing capacity but also in terms of the capacity of power supplies, voltage regulars, heat-sinks, cabling, etc. In particular, it is well recognized that the computing resources (CPUs, DRAM, storage, etc.) are typically significantly underutilized. A more serious issue is that the physical design often assumes that the CPU, memory, network adapter and disk are simultaneously running close their capacities. Real workloads almost never behave this way – it is usually very difficult to saturate more than one resource at the same time.

Data centers are beginning to "derate" specified power and cooling requirements to address this lack of realism; however, it is possible to go significantly beyond this practice and reduce the overall energy footprint of both the servers and the clients. There are two directions to consider in this regard: (a) outright reduction in capacities, i.e., power supplies with lower current rating, smaller heat-sinks, smaller disks, etc., and (b) "smarter" components that can adapt their capacities to the offered load.

Reduced capacity requires more intelligent control so that the application performance or the component does not suffer. For example, an under-capacity power supply needs to be protected by intelligent power consumption capping and the application w/o inadequate resources needs to be migrated. Phase shedding power supplies and VR's are examples of dynamic adjustment of capacity in order to gain better energy efficiency. It is well known that the efficiency of these components increases monotonically with the utilization level; therefore, by enabling only the minimum number of phases, significant efficiencies can be obtained.

III. Types of Energy Adpative Computing

It is clear from the above discussion that adaptation to the available energy is the key to reducing the overall energy footprint of computer systems. An intelligent and agile adaptation to energy, power and thermal constraints allows greater use of renewable energy, less intensive cooling, more frugal designs, and lower resources. In addition, the same adaptation techniques can deliver better user satisfaction by seamlessly compensating for lack of client resources. This is particularly important because the clients are becoming increasingly smaller in size, mobile but want richer capabilities.

[1]Complete independence from power grid is not possible for technologies such as solar or wind, but may be feasible for geothermal.

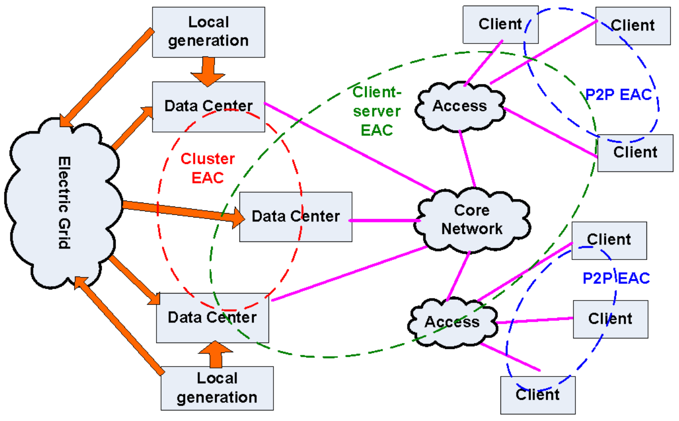

Fig 2. Illustration of energy adaptation loops

For example, the adaptation can deliver better user experience when the client is low on battery power or operating in a hot environment. In fact, if we consider resources other than energy (e.g., available storage), similar adaptation mechanisms can help slow down client obsolescence by transparently compensating for the lack of client resources. This too effectively helps to reduce the energy footprint of client devices.

The purpose of EAC is to maximize overall computation under varying constraints of both energy supply and cooling abilities. Since cooling is required to remove energy wasted as heat, it can be referred to as "energy out" (as opposed to energy taken in, or "energy in"). Maximizing computation while honoring constraints on both energy in and energy out involves a complex optimization at multiple levels involving clients, servers, storage, and networking infrastructure. Thus we can identify 3 types of energy adaptive computing scenarios: (a) Client-server, (b) Peer to Peer (usually client to client), and (c) Cluster computing (or server to server). These are shown in Fig. 2 and are discussed in some detail in the following.

A. Client-Server EAC

The 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. To this end, we introduce the concept of a *client contract*, which is a triplet consisting of the following parts:

1) Client Energy State. For a mobile client, three states may be adequate: normal, energy-out (or thermal) limited, and energy-in limited (e.g., low-battery).
2) Setup QoS: This indicates the QoS that the client is willing to tolerate when either the client or the server that the client is interacting with experience a voluntary or involuntary change of power state. Here the QoS may be application specific and may relate to switchover latency, minimum time between successive interruptions, or other suitable parameters.
3) Operational QoS: This indicates the acceptable QoS after the client moves to the stated power state. The QoS specification is necessarily application specific as explained below.

In most cases, it is impractical to allow a user level control of QoS since it could lead to inappropriate or infeasible choices. Instead, it is expected that the application designer will provide a few "QoS levels" that the application can choose from. So, for example, the operational QoS may be simply one of the following set: "rich", "normal", "degraded", with the precise meaning of those obtained via some table lookup on client and/or server side. A similar comment applies to setup QoS. In fact, the operational QoS category may automatically define the corresponding setup QoS category in order to avoid incompatible pairings of two types of QoS.

In case of client power state transitions, the client will make an explicit request with appropriate contract as a parameter, and the challenge is to handle the request in such a way so as to satisfy the setup and operational QoS with a very high probability. In case of server power state transitions and resulting actions such as migration, the middleware on the server side needs to ensure that client's setup QoS for the current state is honored during the power state transition and the operational QoS is honored *after* the transition.

B. Peer to Peer EAC

In a peer to peer setting involving multiple mobile clients, energy adaption has very different characteristics than in a client-server setting. For simple file-download scenario from a single peer, it is easy to consider the energy state of both requesting and serving peer; however, the fundamental P2P issue of get-give makes this more interesting. In particular, if a peer is in a power constrained mode, it can be allowed to be more selfish by requesting others for low-resolution content but refusing to honor any request. In a more general situation such as BitTorrent where portions of file may come from different clients, assembling the file becomes more challenging. In particular, it might be preferable in this case to use another client as a proxy that assembles the file and then sends it to the requesting node. In general, addressing these issues requires defining appropriate energy related metrics relative to the content requester (client), all potential suppliers (or "servers"), and transit nodes. A framework that allows minimization of global energy usage while satisfying other local performance and energy requirements is essential here.

C. 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, although the client could certainly provide requirements to the servers in terms of its capabilities and even the energy state. In this sense, cluster EAC differs significantly from client-server EAC. Cluster EAC also differs substantially from P2P EAC since servers are supposed to work cooperatively and there is no issue of get-give.

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 profile of the renewable energy infrastructure. As usual, the limits placed at the lower level must necessarily be more flexible than at higher levels. For example, a rack with 10 KW power capacity and 20 servers must make the per-server power limit higher than 500W so that it is possible for some servers to consume more than 500W while others are below the limit. Translating higher level limits into lower level limits is a challenging problem; moreover, it cannot be viewed in isolation. For example, if the servers in rack1 frequently hit the limits and need to be throttled but those in rack2 stay below the limit, a load balancing is in order so that the performance loss due to power limit and the overhead of enforcing limits can be minimized. On the other hand, having many racks stay under the limit constantly is also not optimal since it may be more power efficient remove the load of some servers or racks altogether and let them go into deep sleep mode.

Although in the above we discussed the three EAC scenarios separately, they all need to be addressed together. In particular, while a server responds to client adaptation needs, it itself may need to adapt due to power/thermal limits being exceeded (possibly due to an unrelated HPC application).

IV. Energy Adaptation Techniques in Data Centers

In this section we consider the problem of operating data center elements under given energy constraints. In the following, we use energy and power (really average power) interchangeably; however, it may be necessary to enforce limits on both the average power and instantaneous power. The latter is related to the capacity of power circuits and is most relevant at the level of racks. The average power limitation has to do with variable energy supply as discussed earlier.

Given the total power constraint for a data center, the highest level problem is how to set an "energy budget" for various applications. Certain applications – particularly those involved in background activities – don't even need to run when the energy availability is scarce. Others may require satisfying given SLAs. We assume here that the energy budget for an application has already been estimated. A related problem is the division of energy between servers, networking equipment and storage for each application. We again assume that this is already done, and instead focus on the servers.

The question now is how to operate the servers so that the specified limits can be respected. The net effect of any such mechanism will be to reduce the effective computation rate. In a transactional environment driven by external requests, this reduction may require admission control so that the system remains stable. Since the purpose of this paper is not to visit the well-researched topic of admission control, we assume a simple queue-length threshold based control that limits the congestion resulting from power limits that we impose. Although, in theory, admission control itself can be driven by the available power, such a control is practically unworkable since the relationship between queue length and power is weak at best.

There are basically two ways of reducing the power consumption of a server: (a) exploit the inactive (or sleep) states, or (b) exploit active states via the well known dynamic voltage-frequency scaling (DVFS). Of course, the two can be combined, and we shall consider that possibility as well. These techniques can even be applied to network and storage components, although we shall not delve into that. For a comprehensive discussion of these and many other issues in power/thermal management of data centers, please see [3], [5] and references contained therein. Traffic batching can help reduce the overhead of entering and exiting sleep states [4]; however, this aspect is not directly relevant in this paper.

A. Power Capping with Sleep States

Modern computer systems provide low power *inactive* modes at the level of platform components (e.g., CPU, memory, interconnection links, solid-state disks, etc.) which can be exploited during short idle periods for energy efficient computing (EEC). For example, a memory rank can transition to the so called CKE mode when idle for a few hundred ns or to self-refresh when idle for 10's of microseconds. Similarly, a CPU package can utilize C3 or C6 states depending upon the duration of the idle period [3]. At the higher level, one could use the system level inactive states such as S3 (suspend to RAM) or even S5 (suspend to disk). In the EAC context, suspend to memory type of control is quite adequate since the overall energy availability is unlikely to vary much over several seconds or more.

Traditionally, sleep states are designed to be used when there are no pending requests. For energy capping purposes, however, it becomes necessary to enhance the low power modes so that it is possible to transition the devices to low power mode even when a request is pending. This capability may require additional buffers to hold requests when used at low levels (e.g., for the interconnect); however, at the higher levels (e.g., the entire server, disks, etc.), the requests are managed externally and no additional buffers are necessary.

We now analyze the performance of sleep based power capping under a simple setting. The purpose of these assumptions is to show some basic properties w/o cluttering the analysis with specific details that may be implementation specific. We assume a "well-provisioned" system, defined as follows: (a) the computing resources such as CPU, memory, etc. allocated for the application are adequate to satisfy highest demand; the only resource that may be rationed is the power, and (b) the CPU (rather than the interconnect, memory channel, network, or storage) is the first bottleneck. For such a system, the CPU utilization is a reliable indicator of "system utilization", and we further assume that utilizations of other devices are proportional to the CPU utilization. That is, the nature of the workload does not change with its intensity. In this case, the active system power, i.e., the power over and above the idle power can be easily related to CPU utilization.

Consider the case of a single rack with N identical servers. There are 3 power levels that are relevant for each server: (a) suspend to memory power, denoted as P_{low}, (b) idle power, denoted as P_{idle}, and (c) power at 100% utilization, denoted as P_{full}. Let W_{target} denote the target power available for the entire rack. Suppose that to achieve this target, we keep only n servers (out of N) in active state, and rest in inactive state. Then, if the active servers operate at utilization level U, we have

$$W_{target} = P_{low}(N-n) + n[P_{idle} + U(P_{full} - P_{idle})] \quad (1)$$

where U must be chosen less than some specified maximum value U_{max}. U_{max} depends on the burstiness of the workload and is usually kept less than 80%. Given W_{target} and U_{max}, we can estimate U and n. (Note that the choice of U should be such that n is a whole number.)

Let f_{max} denote the frequency that the CPU operates at w/o DVFS controls. Let us consider a transactional workload where each transaction has a *path length* PL, i.e., it takes PL instructions to complete a transaction. Let CPI denote the average number of CPU cycles per instruction. Then, the rate of work accomplished with this control is simply $nUf_{max}/(PL \times CPI)$.

B. Power Capping with DVFS Controls

The CPUs active states are popularly known as P states, numbered as $P_0, P_1, \ldots P_K$ that correspond to decreasing voltage/frequency combinations (K is total number of P states provided). Other platform components such as interconnect and DRAM also have similar states. In fact, the RPM control of disks and speed control of fans also falls in the same class. For simplicity, we assume that all major resources have appropriate speed controls. Notice that for an effective management the active power states of multiple components of a platform must be coordinated. For example, if the CPU is running at half the maximum frequency, running the interconnect and memory at full rate is pointless. Similarly, if only the memory is slowed down, the performance will take a serious hit due to excessive CPU stalls.

For simplicity, we shall illustrate the scheme in the following for only CPU power. It is easy to extend the results by summing up the power contributions of all significant components in the platform.

Power capping can be achieved with P states by simply oscillating between two adjacent states whose power consumption bounds the desired power. For example, if $P1$ and $P2$ power consumptions are 90 and 60 watts respectively, a cap of 70 watts can be achieved by oscillating between these states with 1/3 residence in $P1$ state. That is, if ν is the time fraction spent in the lower power state, the effective full power, denoted P^i_{eff}, is given by:

$$P^i_{eff} = \nu P^i_{full} + (1-\nu)P^{i-1}_{full} \quad (2)$$

Let τ denote the cycle time for this control. Then the time spent in lower & higher power states in each cycle is $\nu\tau$ and $(1-\nu)\tau$ respectively. It is desirable to choose τ rather large so that the state change overhead can be kept negligible. The limiting factor is the probability that the rack level power budget will violate the power spike specifications. With a typical spike tolerance in 10's of ms, we can choose rather large τ and switching overhead is not an issue.

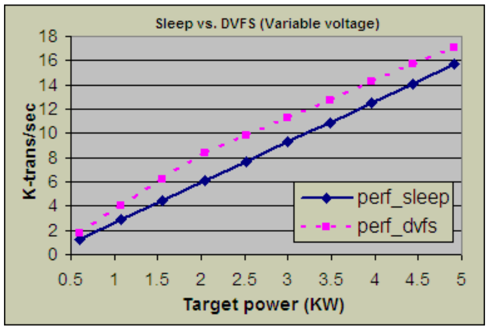

Fig 3. Comparative performance with Variable Voltages

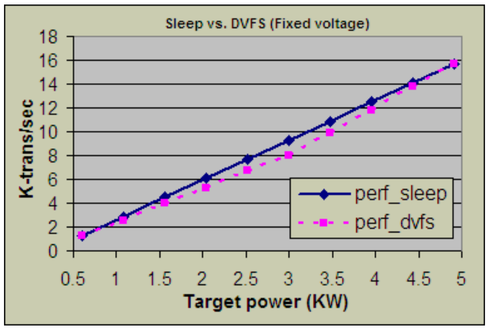

Fig 4. Comparative performance with Fixed Voltages

Let V_i and F_i denote, respectively, the voltage and frequencies of these states relative to those for state 0. That is, $V_0 = F_0 = 1$ and all others are < 1. Then,

$$P^i_{idle} = P^0_{idle}V_i^2 P^i_{full} = P^i_{idle} + (P_{full} - P_{idle})V_i^2 F_i \quad (3)$$

The basic DVFS control algorithm is as follows: Given the rack power constraint W_{target}, we determine the appropriate bounding states i and $i+1$ such that $P^{i+1}_{full} \leq W_{target}/N < P^i_{full}$, and compute ν from eqn (2). Note that if the power constraint is too tight, i.e., $W_{target}/N < P^{K-1}_{full}$, then we have no choice but to put appropriate number of servers in the low power mode. Thus the active number of servers, say N', could be less than N, and the power consumption will also have a low-power component for $N - N'$ servers.

Under the (rather idealized) assumption of a proportional DVFS control of all platform components, the relative speed of all platform components will stay the same. Consequently, the CPI will stay the same in all P states. Therefore, when the system is in state P_i the rate of accomplishing the work will be $f_{eff}/(PLCPI)$ per server, where f_{eff} is the effective frequency given by $f_{eff} = f_{max}[\nu F_i + (1-\nu)F_{i-1}]$. This multiplied by N' gives the overall work rate.

C. Comparison of Power Capping Methods

In this section we show some representative results from our analysis. Given a power cap, the main quantity of interest is the effective processing rate with sleep state and DVFS controls. For DVFS, we assume four P states with relative frequencies of 1.0, 0.8, 0.6 and 0.4. However, in terms of voltage levels, we consider two cases: (a) decreasing relative voltage levels of (1.0, 0.95, 0.90, 0.80), and (b) no decrease in voltage levels. Situation (a) corresponds to current technology where a modest voltage decrease is still possible, whereas situation (b) depicts the case in the near future when the voltages would have already shrunk close to the threshold and hence no further decrease in voltages will be possible with lower frequencies.

Figs 3 and 4 show the performance for cases (a) and (b) as a function of power cap. It is seen that in case (a), DVFS yields better performance and hence is a better technique all around. However, for case (b), the sleep state control works better except at very low and very high target power levels. The reason for the dramatic impact of voltage scaling is that the power increases as a square of voltage, and thus going to a lower power P state is highly beneficial. With only frequency control, however, the sleep state control works much better since it is able to significantly reduce the static power consumption. It is seen that in the future as voltage margins shrink, we would need to exploit sleep states for energy adaptive computing.

V. Conclusions

In this paper we discussed the concept of energy adaptive computing (EAC) which puts power/thermal controls at the heart of distributed computing. We discussed how EAC can make IT more sustainable and elaborated on 3 different types of EAC scenarios. We also discussed and contrasted two basic techniques for forcing energy adaptation, namely the use of sleep states and dynamic frequency/voltage scaling. It was shown that in the future, we may need to depend more on sleep states since DVFS will not be very effective.

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.

References

[1] D. Ernst, S. Das, S. Lee, et.al., “Razor: Circuit-Level Correction of Timing Errors for Low-Power Operation”, IEEE Micro, vol. 24, no. 6, pp. 10-20, Nov 2004.

[2] http://www.itbusinessedge.com/cm/community/news/inf/blog/intel-tests-free-cooling-in-the-data-center/?cs=20341

[3] K. Kant, “Data Center Evolution: A Tutorial on State of the Art, Issues, and Challenges”, to appear in Computer Networks Journal.

[4] A. Papathanasiou and M. Scott, ”Energy Efficiency through Burstiness”, Proc of the 5th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA’03), pp. 44-53, Oct 2003.

[5] V. Venkatachalam and M. Franz, “Power Reduction Techniques for Microprocessors”, ACM computing surveys, Vol 37, NO 3, Sept 2005, pp 195-237. (http://www.ics.uci.edu/ vvenkata/finalpaper.pdf)