Research Directions in Power and Energy Conservation for Clusters *

Ricardo Bianchini

Department of Computer Science
Rutgers University

{ricardob}@cs.rutgers.edu

Technical Report DCS–TR–466, November 2001

Abstract

Data centers, hosting centers, and Internet companies typically rely on large clusters of computers to interface with users and store data. The computational, networking, and cooling infrastructures of these organizations consume significant amounts of power and energy. Research on power and energy conservation for these systems can have several benefits. In this paper, we motivate this research direction, point out a few of the related issues and challenges, overview the small body of previous work, and suggest several specific research avenues that are being addressed in the Rutgers DARK Laboratory.

1 Introduction

Power and energy consumption have always been critical concerns for laptop and hand-held devices, as these devices generally run on batteries. The amount of power consumed by these devices determines the temperature of the device components, whereas the energy consumed ultimately determines the battery life. As a result, a tremendous amount of research has been directed towards power and energy conservation for battery-operated devices (e.g. [7, 12, 8, 5, 6]).

In contrast, we advocate research on power and energy conservation for clusters of servers, i.e. ensembles of networked computers connected to the power grid. Our motivation is that large clusters, such as those that support most data centers, Internet companies, and a large number of teaching and research institutions, consume significant amounts of power and energy. Power and energy conservation can reduce the installation and operational costs of these clusters, as well as protect the environment. Section 2 discusses our motivation in more detail.

The research we advocate is based on the observation that clusters and their workloads have certain characteristics that can be leveraged to achieve significant power and energy savings. In particular, we see at least four characteristics that can be exploited in clusters: their widespread replication of resources; the time-varying intensity of certain cluster workloads; the disparate resource requirements of certain workloads; and the amenability of clusters to heterogeneous configurations.

We can take advantage of these characteristics by intelligently scheduling the demand for resources across the cluster. The goal of demand scheduling would be to idle resources for relatively long periods of time. During these periods, resources could be transitioned from active mode to a mode with lower power requirements

*This research has been supported by NSF under grant # CCR-9986046.

(a low-power mode). The main challenge is to create these periods of idleness without degrading overall performance, or at least without degrading it beyond a pre-established, "acceptable" level. To overcome this challenge, we need to determine the power, energy, and performance implications of various cluster-wide demand assignment strategies, to determine the implications of different resource power modes and the cost of transitions between modes, and to determine the role of different software and hardware cluster configurations in conserving power and energy. Section 3 discusses these issues.

We are addressing these challenges in the Rutgers DARK Laboratory. To make our investigation concrete, we are designing, implementing, and evaluating cluster-wide scheduling techniques and systems that can conserve power and energy without compromising performance excessively. We overview our first results in section 4. By studying these systems in detail, we will be able to understand the implications of the techniques and determine a priori what are the most appropriate techniques for other systems. Section 5 describes the main ideas behind our techniques and uses a power-aware WWW server to illustrate them.

2 Motivation

The importance of conserving power. Power consumption is an important concern in the context of clusters as it directly influences the cooling requirements of the cluster. In fact, a medium to large number of high-performance nodes racked closely together in the same room, as is usually the case with clusters, requires a significant investment in cooling, both in terms of sophisticated racks and heavy-duty air conditioning systems. At each Google site as of Spring/2001, for instance, there were 40 racks, each of which with 80 PCs, for a total power consumption of nothing less than 180 kWatts. Cooling such a large installation is an expensive challenge. Besides cooling under normal operation, power consumption also influences the required investments in backup cooling and backup power-generation equipment for clusters that can never be unavailable, such as those of companies that provide services on the Internet.

Taking a broader perspective, at the height of the economic boom in the United States in 2000, the power requirements of clusters had become a major issue for several states, such as California, New York, and Massachusetts. The New York Times reported that the planned 46 data centers for the New York city metropolitan area alone had asked for a minimum draw of 500 megawatts of power, which is enough to power 500,000 households [11]. The inability of these states to meet such enormous power requirements had already resulted in data center projects being cancelled or delayed [11].

Even if these states were to make a tremendous investment in new power plants, power conservation should still be an important goal in that most power-generation technologies (such as nuclear and coal-based generation) have a negative impact on the environment. The Energy Star program was the first to promote power-efficient computers due to environmental considerations. Unfortunately, power efficiency had not become an issue for the server market until recently.

The importance of conserving energy. Besides power consumption, energy consumption is also important for large clusters. Both the computational and the air conditioning infrastructures consume energy. This energy consumption is reflected in the electricity bill, which can be significant for a large and/or dense cluster in a heavily air-conditioned room. Research and teaching organizations, in particular, may find it difficult to cover high energy costs.

Again, the news services provide anecdotal confirmation of how critical energy consumption is for large computer systems. An article from ComputerWorld [4] discusses last year's surge in energy consumption and includes a quote from a spokesman for Silicon Valley Power saying that a single data center can consume more energy than the largest manufacturing plant the company serves. A Google site, for instance, consumes no less than 130 MWh per month just with the cluster infrastructure.

The importance of this research in the long run. At one end of the spectrum, the increasing complexity (and power consumption) of high-end server hardware and software will only increase the importance of this

research in the future. For these systems, this research will enable power and energy conservation without degrading performance.

At the other end of the spectrum, we believe that advances in power and energy conservation for battery-operated devices will start to make their way into low-end servers. This research will benefit directly from these orthogonal developments, as we suggest a focus on clusters, rather than on stand-alone machines. One example of such developments is the cluster of blade servers. The role that clustered blades will play in the marketplace is still unclear, since they will very likely have a worse cost/performance ratio than clusters of traditional servers. For instance, the hard disk drives used in the commercially available blades cost more and perform worse than traditional server drives. Nevertheless, this research can benefit from blades, as they permit a finer control of power modes than traditional servers. This finer control will increase our ability to conserve power and energy without degrading performance beyond a desired level.

Furthermore, one of the main uses of clustered blades will be to increase system capacity in the same amount of physical space as traditional clusters. In these scenarios, the need for power and energy conservation will become even more significant. For instance, a commercially available cluster of blades packs 24 nodes into the same volume as just one of our single-processor servers. Unfortunately, the overall power consumed by the clustered blades will be about four times higher than that of our server. Cooling such ultra-dense clusters will also be a tremendous challenge.

The **bottom line** is that to conserve the power and energy consumed by clusters eases deployment and installation, protects the environment, and can potentially save a lot of money. In fact, even when it is not possible to reduce the maximum power requirements of a cluster (i.e. it is not possible to cut down the one-time cost of cooling and backup power-generation systems), reducing the common-case power and energy consumption reduces the operational cost of the cooling system. Given a fixed budget, one dollar spent on primary or backup air conditioning, backup power generation, or electricity is one dollar not spent in things that can actually produce real benefits (e.g. higher revenue), such as more cluster nodes, faster disks, more qualified personnel, etc.

3 The Problem and Potential Solutions

The ultimate goal of the research we propose is to conserve power and energy without excessively degrading performance. To achieve this goal, we would like the maximum power consumed by the power-optimized system, $MaxPower_{opt}$, to be at most the same as that of the original, non-optimized system, $MaxPower_{orig}$, i.e. $MaxPower_{orig} \geq MaxPower_{opt}$. Reducing the maximum power reduces the required capacity of cooling systems. Unfortunately, lowering the maximum power may not always be possible. Nevertheless, guaranteeing that the average power consumed by the power-optimized system during the execution of each workload is lower than that of the original system, i.e. $AvgPower_{orig} \geq AvgPower_{opt}$, is also useful to lower the operational cost of cooling systems. Similarly, we would like to guarantee that the energy consumed by the system for each workload is conserved, i.e. $Energy_{orig} > Energy_{opt}$, to lower electricity bills. Finally, the runtime (throughput) performance of the original system should not be greater (smaller) than the runtime (throughput) of the optimized system by more than a pre-defined threshold, i.e. $RT_{opt} - RT_{orig} \leq runtime\_threshold$ and $TP_{orig} - TP_{opt} \leq throughput\_threshold$. These thresholds can be set to 0, if we decide not to accept any performance degradation.

Formally, we define the power and energy consumed by a cluster as:

$$Power_t = \sum_{n}^{nodes} (P_p^n + \sum_{r}^{resources} P_{rm}^n) + P_s \ ,$$

$$MaxPower = max(Power_t), AvgPower = (\sum_{t}^{time} Power_t) \div time, and$$

$$Energy = \sum_{n}^{nodes} (P_p^n \times T_p^n + \sum_{r}^{resources} ((\sum_{m}^{modes} P_{rm}^n \times T_{rm}^n) + M_r^n)) + P_s \times T_s,$$

where $Power_t$ is the instantaneous power consumed at time t, P_p is the fixed power consumed by each node's power supply, T_p is the amount of time during which the power supply is on, P_{rm} is the power consumed by resource r (CPU, memory, disk, etc) in power mode m, T_{rm} is the amount of time during which resource r is in mode m, M_r is the energy consumed in transitions between modes, P_s is the power consumed by the communication switch, and T_s is the time during which the switch is powered on.

Note that these formulas are intended simply as an illustration of the important factors determining power and energy consumption in clusters. In particular, the formulas disregard the variations in resource power consumption associated with differences in offered loads. For instance, the power consumed by a disk varies with the locality of the load requested of it. Determining complete and precise models of the power and energy consumption of clusters is an open research topic that we discuss later.

Nevertheless, it is clear from the formulas that we can conserve power and energy in a cluster by: (a) reducing the time during which power supplies, resources, or the switch are on or in high power modes, while increasing the off or low power times; or (b) reducing the power consumed by power supplies, by resource modes, or by the switch.

These objectives can be achieved by resorting to standard, stand-alone system optimizations, such as transitioning to low power modes during periods of idleness (e.g. [5]) or light load (e.g. [12, 1]) and designing hardware components that are more power-efficient (e.g. [7]). However, we are more interested in research that concerns ensembles of computers and the opportunities for power and energy conservation for clusters, in particular. Cluster-oriented and stand-alone conservation are orthogonal, in that the former can benefit from the latter as well as increase its achievable gains.

In the context of clusters, objective (a) can be achieved by scheduling the offered load across the cluster so that resources or entire nodes can be idled and then transitioned to a low-power mode or powered off altogether. For some resources, e.g. the CPU, degrading the performance of the resource rather than increasing its idle time is an option. Objective (b) can be achieved by intelligent scheduling that can exploit heterogeneous cluster nodes.

Note, however, that such optimizations may have negative side-effects, if they are not applied carefully. Idling certain resources often has the side-effect of increasing the utilization of others, which may in turn increase power consumption and degrade performance. Exploiting heterogeneous configurations with low power but slower nodes, such as laptop-style or blade servers, can degrade performance excessively.

In summary, we feel that power-aware cluster-wide scheduling for both homogeneous and heterogeneous clusters should be a key focus of research in power and energy conservation. For this research to succeed, another important research goal should be a deep understanding of the implications of varying the amount and type of load offered to resources and nodes on the power and time consumed by different parts of the cluster.

4 Previous Research: Load Concentration

We recently proposed a simple technique, called load concentration, and algorithm that dynamically schedule the workload offered to a cluster of servers so that resources can be idled and moved to low-power modes [9]. Our work proposed the load concentration technique and algorithm as applicable to any clustered server. To demonstrate their benefits more concretely, we applied them in the implementation of a clustered WWW server and an operating system for clustered cycle servers. Researchers from Duke University [3] have also considered load concentration in the context of clustered WWW servers. Here we present an overview of our approach to load concentration and some interesting results.

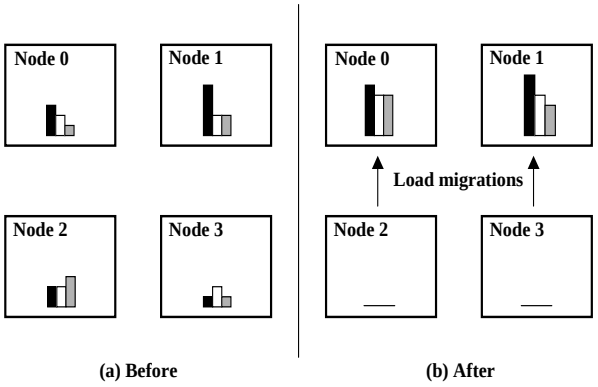

Figure 1: Before (a) and after (b) load concentration. The bars represent the utilization of various resources, such as CPU, disk, and network interface. Nodes 2 and 3 become idle after load concentration and thus can be powered off.

```
Periodically do
  if it is acceptable to put resources in low-power mode
    choose nodes (savers) with low demand for the resource
    if necessary, determine nodes to receive load of victims
      and ask savers to migrate their load out
    ask savers to put their resources in low-power mode
  else
    if the activation of new resources is necessary
      activate the resources in some of the savers
      if necessary, determine load to be sent to new nodes
        and ask nodes to share their load with new nodes
```

Figure 2: Pseudo-code for load concentration algorithm.

4.1 Overview

The basic idea behind load concentration is to dynamically distribute the load offered to the cluster so that, under light load, some cluster resources can be idled and put in low-power modes. In effect, this technique concentrates the need for certain resources in a subset of the nodes, performing the inverse operation of load balancing. For example, consider a clustered WWW server. Under light load, the server could forward incoming requests within the cluster so that disk accesses would only be performed at certain nodes; the server could then power down the disks of the other nodes.

Load concentration can be applied to idle multiple resources, such as network interfaces, disks, and CPUs, at the same time. In fact, an extreme approach we experimented with is to idle entire nodes, so that they can be shutdown and put in the lowest possible power mode, i.e. off. Figure 1 depicts this scenario. However, load concentration has to be performed carefully in the interest of performance. For example, load concentration operations may involve significant overhead (and additional power and energy consumption) and thus can degrade performance if executed too frequently. As another example, concentrating disk-intensive loads in too small a subset of the cluster can cause the saturation of one or more disks, which would degrade overall performance. When some resource becomes saturated, the system should activate additional resources and redistribute the load to alleviate or eliminate any performance degradation. Thus, load concentration and load balancing have to coexist and cooperate.

Figure 2 shows the general form of a simple algorithm that represents our first attempt to capture this cooperation. Based on the current resource configuration and offered load, the algorithm periodically decides whether to deactivate (first branch of the main *if* statement in the figure) or activate (second branch of the *if* statement) resources according to the expected performance and power and energy implications of the decision. Deactivation may be acceptable when performance would not be excessively degraded *and* less power and/or energy would be consumed. Activation may be necessary when performance is being degraded excessively *or* less power and/or energy would be consumed.

4.2 Preliminary Results

We developed a simple load concentration-based WWW server for a homogeneous cluster. The server dynamically turns entire nodes on – to be able to handle heavy load efficiently – and off – to save power under lighter load. In this extreme case, the load concentration algorithm determines the cluster configuration.

Cluster configuration algorithm. For each current configuration and currently offered load, the algorithm decides whether to add (turn on) or remove (shut down and turn off) nodes, according to the expected throughput and power implications of the decision. Files are replicated at all nodes, as is commonly the case in clustered WWW servers.

This algorithm is a simpler version of the one in figure 2 in that it only removes or adds a single node to the cluster at a time, due to the high overhead of reconfigurations. Furthermore, for the WWW server, it is not necessary to migrate load away from (to) a node that is about to be excluded from (added to) the cluster. The load can be naturally redistributed among the available nodes, by a load balancing front-end and/or the server's own request distribution algorithm.

Measuring our cluster we found that: (a) there is a relatively small difference (about 24 Watts) in power consumption between an idle node and a fully utilized node; and (b) the penalty for keeping a node powered on is high (70 Watts), even if it is idle. Thus, for these specific cluster nodes, turning a node off always saves power, even if its load has to be moved to one or more other nodes. As a result, our system always decreases the number of nodes, provided that the expected utilization of the remaining nodes is acceptable.

In more detail, a node addition is deemed necessary if any resource of any node is more highly utilized than a fixed threshold (90% in our experiments) and the time since the last reconfiguration is larger than another fixed threshold, `elapse` (200 seconds in our experiments). Setting the utilization threshold at 90% rather than at 100% provides some slack to compensate for the time it takes for a node to be rebooted.

Node removal is deemed acceptable if, after the reconfiguration, no resource would have a utilization that is higher than 90% and the time since the last reconfiguration is larger than `elapse`. To see how we predict utilizations, suppose a scenario with 3 cluster nodes, each of which with disk utilizations of 80%, 30%, and 20% of their nominal bandwidth. By adding up all of these disk utilizations (and disregarding other resources to simplify the example), we find that the server could run with no throughput degradation on 2 nodes ($130 < 90 \times 2$), but not on one node ($130 > 90$).

To simplify our implementation of the algorithm, all nodes periodically inform a master node (node 0) about their approximate CPU, disk, and network interface utilizations. To smooth out short bursts of activity, each of these utilizations is exponentially amortized over time. The interval between utilization computations is 10 seconds.

Methodology. We performed experiments with a cluster of 8 PCs connected by a Fast Ethernet switch and a Giganet switch. Each of the nodes contains an 800-MHz Pentium III processor, 512 MBytes of memory, two 7200 rpm disks (only one disk is actually used), and network interfaces for the switches. All machines are connected to a power strip that allows for remote control of the outlets. Shutting a node down and turning it off takes approximately 45 seconds and bringing it back up takes approximately 100 seconds.

Besides the main cluster, we use another 12 Pentium-based machines to generate load for the WWW servers. For simplicity, we did not experiment with a front-end device that would hide the powering down of cluster nodes from clients. Instead, clients poll all servers every 10 seconds and can thus detect cluster reconfigurations and adapt their behavior accordingly. The clients send requests to the available nodes of the server in randomized fashion and reproduce the accesses made to the main server for the Computer Science Department of Rutgers University in the first 25 days of March 2000. Requests are directed to the cluster with a bell-shaped distribution. To shorten the length of the experiments, we generate significant changes in offered load in very little time.

Experimental Results. Figure 3 presents the evolution of the cluster configuration and utilizations for each resource as a function of time in seconds. The utilization of each resource is plotted as a percentage of the nominal throughput of the same resource in one node. The figure shows that for this workload the network interface is the bottleneck throughout the whole execution of the experiment (1 hour and 20 minutes), followed closely by the disk.

Figure 4 presents the power consumption of the whole cluster for two versions of the same experiment as a function of time. The lower curve (labeled "Dynamic Configuration") represents the version in which we

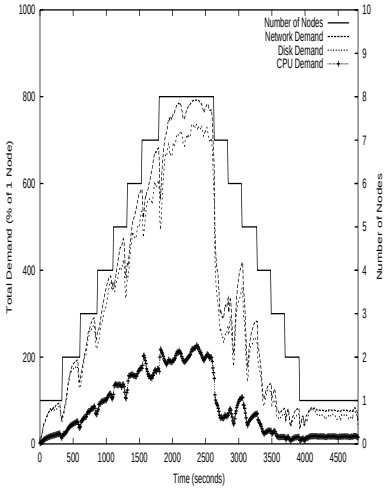

Figure 3: Cluster evolution and resource utilizations.

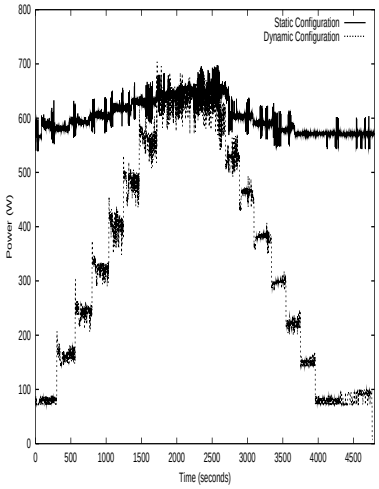

Figure 4: Power consumption under static and dynamic configurations.

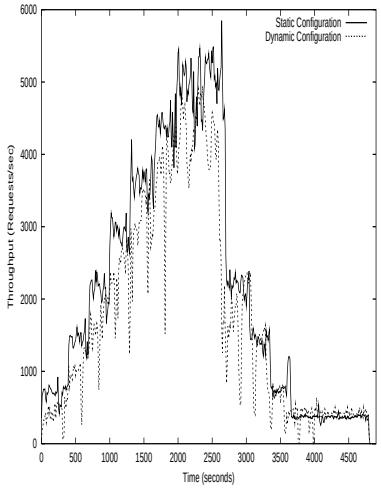

Figure 5: Throughput under static and dynamic configurations.

run the power-aware server, i.e. the cluster configuration is dynamically adapted to respond to variations in resource utilization. The higher curve (labeled "Static Configuration") represents a situation where we run the original server, i.e. the cluster configuration is fixed at 8 nodes. As can be seen in the figure, our modified WWW server can reduce power consumption significantly for most of the execution of our experiment. Power savings actually reach 86% when the resource utilizations require only a single node. Our energy savings are also significant. Calculating the area below the two curves, we find that the modified WWW server saves 43% in energy.

Finally, it is important to make sure that throughput is not sacrificed excessively in favor of power and energy savings. Figure 5 shows the throughput of the server in requests serviced per second for the two versions mentioned above as a function of time. The figure shows that throughput only suffers significantly in comparison to the static system during the times in which nodes are being added to or removed from the system. During these times, each node of the server has to update its internal data structures and communication channels. A new node has to load its cache. All of these operations, especially the latter, induce overheads that are responsible for the lower throughput. Overall, the dynamic configuration services 19% fewer requests than its static counterpart in this experiment. This degradation is relatively small compared to the significant reductions in power and energy consumption achieved by the dynamic system.

5 Research in the Rutgers DARK Lab

Our research in the Rutgers DARK Lab is currently focused on the analytical modeling and implementation of techniques and systems that can conserve power and energy in clusters. We have been working with three different techniques: load concentration, functionality specialization, and heterogeneous configuration.

As just described, load concentration schedules the cluster load dynamically to create idle time. In contrast, functionality specialization biases the load distribution statically. Heterogeneous configuration relies on functionality specialization and applies a mixture of power-hungry, high-performance nodes and lower power, lower performance nodes.

All these power and energy conserving techniques involve a complex performance vs. power and energy tradeoff, as they idle resources and/or use low-performance nodes. We intend to model this tradeoff. The techniques will then use the models we develop to conserve power and energy while limiting performance degradation. In the process of studying these techniques we will develop several power-aware systems. Here, we again use a power-aware WWW server to illustrate our ideas.

5.1 Modeling Power, Energy, and Performance

Understanding the implications of cluster-wide scheduling on power, energy, and performance is the key to designing techniques and systems that can effectively conserve these resources without unacceptable performance degradation. In our preliminary work with load concentration, we did not need a deep understanding of these implications, as a result of the high power consumption of our cluster nodes' power supplies. However, blade servers do not have the same characteristics and, thus, require more detailed analysis.

We are starting to develop analytical models that consider these implications explicitly. In particular, we plan to extend the formulas we present in section 3 to take variations in load into account. Unfortunately, this is a complex proposition, especially when we consider heterogeneous machines. Exact power consumption predictions are not straightforward, although it might seem that way superficially. The problem is that it is difficult to predict the power to be consumed by a node after it receives some arbitrary load. Conversely, it is difficult to predict the power to be consumed by a node after some of its load is moved elsewhere. Similarly, exact performance (and thus energy consumption) predictions of the effect of load changes can be difficult.

Our initial approach to this research essentially involves observing the power and performance behavior of each resource as a function of the amount and type of load imposed on it. When the load characteristics can make a significant difference in behavior, we plan to observe the extreme, i.e. best case and worst case, behaviors. We will then fit the profile of each resource obtained by these observations into a model that can be used to predict resource behavior. When all resources are understood and modeled, we will program a combined model into our systems to guide their decisions dynamically.

Researchers from IBM Austin developed a model of CPU and disk behavior for the specific case of a single-node WWW server in [1]. In contrast with their approach, we seek to develop models that are applicable to any type of workload and that take cluster-wide demand scheduling into account explicitly.

5.2 Further Research on Load Concentration

We plan to improve and extend our load concentration work in several ways. First, we will extend our algorithm and implementations to transition multiple devices between multiple sleep states, rather than just turning entire nodes on and off. This will give us finer control of power and performance. For instance, we will exploit load concentration to enable frequency and voltage scaling of CPUs.

Second, we will extend our algorithm and implementations to consider load migration and state transition costs (in terms of both power and performance) explicitly. By considering these costs, we expect to be able to predict the effect of algorithm decisions more accurately, as well as increase our ability to conserve power without degrading performance.

Third, we will extend our algorithm and implementations to explicitly consider energy as well as power tradeoffs. This will be possible with the more accurate performance predictions provided by our analytical models. We also envision using on-line monitoring of power consumption (using feedback from our power measurement equipment), performance, and resource utilization, so that load concentration decisions can be reverted, in case initial measurements suggest that they will not produce the estimated results. This strategy will be particularly useful if we find that exact predictions are impractical.

Fourth, we plan to experiment with real, non-accelerated workloads. If these workloads require, i.e. offered loads increase too quickly in comparison to resource activation and load migration times, we will also experiment with resource pre-activation strategies to avoid performance degradation. An example of such a strategy is to keep more activated resources than strictly necessary and trigger new activations long before reaching full utilization.

Finally, we will compare the benefits of conserving power and energy by exploiting load concentration against strictly local or stand-alone conservation approaches.

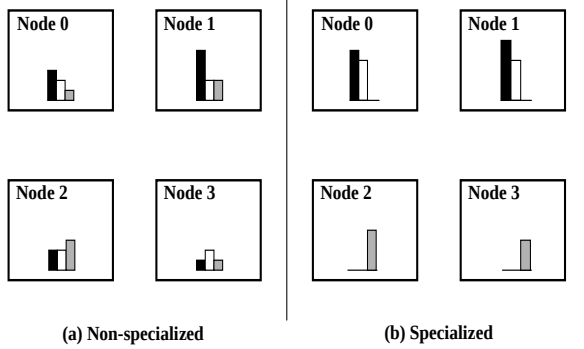

Figure 6: Non-specialized (a) and specialized (b) organizations. Again, the bars represent the utilization of various resources. Note in (b) that nodes 2 and 3 have two resources that are frequently idle due to specialization.

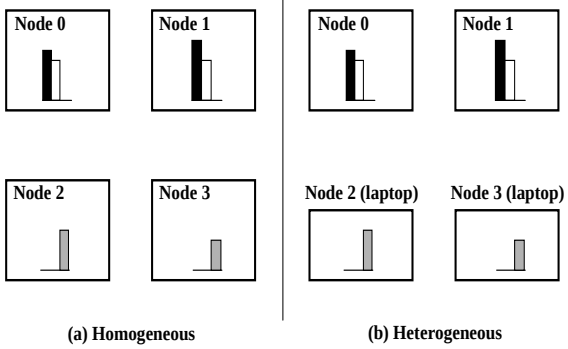

Figure 7: Homogeneous (a) and heterogeneous (b) organizations. Again, the bars represent the utilization of various resources. Note in (b) that nodes 0 and 1 are servers, whereas nodes 2 and 3 are laptop-type machines.

5.3 Functionality Specialization

The idea behind functionality specialization is to *statically* assign different functionality to different cluster nodes, so that functionalities that are frequently and rarely required can be segregated explicitly. This makes the offered load distribution become skewed towards the nodes with frequently required functionality, while the resources of the remaining nodes become idle longer and more often. By relying on homogeneous nodes, the static assignment of functionalities can be changed at runtime, as a result of exceptional/unexpected and potentially long-lasting changes in behavior. We expect these changes to occur rarely if at all for well-designed systems.

In effect, functionality specialization statically imposes a hierarchy on the cluster, such that the resources of nodes that lie towards the top of the hierarchy are more likely to be active than resources of bottom nodes. During idle times, resources can be put in low-power modes. Figure 6 depicts this organization. The key issues in functionality specialization are determining the different functionalities, dimensioning (or provisioning) each level of a hierarchy, and determining the ideal power modes for each level. All of these issues directly affect conservation and performance.

An interesting example of functionality specialization is that of a two-level WWW server or WWW accelerator [10]. Such a server divides the cluster nodes into two groups, front and back-end nodes. Front-end nodes handle the incoming traffic and service requests for cached files and dynamic content. Back-end nodes are responsible for disk accesses when requests for static content miss in a front-end cache. To conserve power and energy, the back-end nodes can keep their resources in low-power mode most of the time. In fact, for high enough hit rates, entire back-end nodes can remain in low-power mode most of the time; other nodes would be responsible for re-activating them when necessary. The challenges are then to provision front and back-end levels, to select power modes for back-end resources, and to to devise schemes for dynamically changing the hierarchy and modes as a result of variations in amount and/or characteristics of the offered load.

Similarly to load concentration, functionality specialization may involve load migration and require light-sleep modes or resource pre-activation techniques for ideal performance under fast-varying offered loads. However, functionality specialization has two main advantages over load concentration: (1) by relying on a static bias for the load distribution, functionality specialization will entail lower power and energy costs to distribute load; and (2) nodes at different levels of the hierarchy can apply different policies for changing power modes. For example, back-end nodes can be more aggressive in terms of decreasing the frequency and voltage used by their CPUs than front-end nodes.

We are currently addressing these issues and challenges in the context of the hierarchical power-aware WWW server and a hierarchical power-aware file system. Our hierarchical power-aware WWW server is organized in two levels as in the example mentioned above. However, our front-end nodes communicate with each other to increase their effective cache space [2]. Our hierarchical power-aware file system allocates files to different disks according to their popularity. For file systems that exhibit a significant disparity in file popularity (such as those of WWW servers, for instance, which are known to exhibit a Zipf distribution of file accesses), this allocation strategy causes certain nodes to be active more frequently than others, allowing the system to put the latter nodes in low-power modes frequently and for long periods of time.

5.4 Heterogeneous Configuration

Functionality specialization is intended to conserve power and energy for homogeneous cluster systems. However, it should be possible to achieve further gains by exploiting software and hardware that are optimized for the different levels of a hierarchy. This is the basic idea behind the heterogeneous configuration technique.

In general, a heterogeneous configuration will involve high-performance, power-intensive resources towards the top of the hierarchy. To see why this is, recall that the top nodes will implement the frequently required functionality, so they will be key in achieving high performance. The nodes toward the bottom of the hierarchy will generally require lower performance and power. In fact, we envision hierarchical cluster systems comprised of high-performance servers at the top and laptop-type computers (or blade servers) at the bottom of the hierarchy. Figure 7 depicts this organization.

Let us use the example of the hierarchical WWW server we introduced in section 5.3 to illustrate this technique. Under heterogeneous configuration, the software and the hardware of the front and back-end nodes can be optimized for the functionality they must perform. For instance, front-end nodes can be implemented by diskless multiprocessors and can use (operating system and server) software that deals solely with communication and cache hits. Restricting the scope of the software should translate into power and energy gains, as well as performance improvements.

Given that heterogeneous configuration as described above relies on a hierarchical organization with functionality specialization, these two techniques share some of the same issues and challenges. However, dimensioning the different levels of a hierarchy can be more challenging under a heterogeneous configuration, since some cluster nodes may not be able to perform certain functions. Getting back to the example above, if front-end nodes do not possess disks, they cannot be used to increase disk bandwidth when the working set size is much larger than the sum of the front-end caches.

Even when all cluster nodes can perform all functions, dimensioning can be difficult under a heterogeneous configuration because the resources at different levels have different performance and power consumption. In other words, some nodes may not be able to perform all functions efficiently. Thus, achieving power and energy savings without degrading performance excessively will involve a tradeoff between the power, energy, and performance of the different cluster nodes.

We are currently experimenting with a heterogeneous WWW server in which front-end nodes are implemented by high-performance four-processor servers and back-end nodes are implemented by laptop-type (or blade) machines. In our heterogeneous file system experiments, we will evaluate a scenario in which nodes that store popular, fairly popular, and unpopular files will be based on high-performance four-processor servers, single-processor servers or desktops, and laptop-type machines, respectively.

6 Conclusions

The impact of the research in power and energy conservation for clusters can be substantial, given the large number of clusters deployed around the world and the recently renewed focus on energy conservation. We

have discussed several open problems and potential solutions to these problems. In particular, we in the Rutgers DARK Lab are studying three techniques that can achieve significant power and energy savings for clusters: load concentration, functionality specialization, and heterogeneous configuration. The challenge we face is to understand the implications of each of these techniques as they are applied to real cluster systems.

Finally, it is important to note that, even though we focus on clusters, the same ideas and approach can be applied to other multi-node or multi-disk systems, such as a single server with a disk array for storage, geographically distributed servers, and multiprocessors.

Acknowledgements

We would like to thank the other members of the DARK Lab, especially Enrique V. Carrera, Eduardo Pinheiro, and Taliver Heath, for their contributions to the ideas described here. We would also like to thank Uli Kremer for discussions on the topic of this paper and for letting us use the power measurement infrastructure of the Energy Efficiency and Low-Power (EEL) lab at Rutgers.

References

[1] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. The Case for Power Management in Web Servers. In Graybill and Melhem, editors, *Power-Aware Computing*. Kluwer Academic Publishers, January 2002.

[2] E. V. Carrera and R. Bianchini. Efficiency vs. portability in cluster-based network servers. In *Proceedings of the 8th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, June 2001.

[3] J. Chase, D. Anderson, P. Thackar, A. Vahdat, and R. Boyle. Managing energy and server resources in hosting centers. In *Proceedings of the 18th Symposium on Operating Systems Principles*, October 2001.

[4] ComputerWorld. Net blamed as crisis roils california. January 2001. http://www.idg.net/go.cgi?id=426336.

[5] Fred Douglis and P. Krishnan. Adaptive disk spin-down policies for mobile computers. *Computing Systems*, 8(4):381–413, 1995.

[6] Jason Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *Proceedings of the 17th Symposium on Operating Systems Principles*, pages 48–63, 1999.

[7] T. Halfhill. Transmeta breaks the x86 low-power barrier. In *Microprocessor Report*, February 2000.

[8] Alvin R. Lebeck, Xiaobo Fan, Heng Zeng, and Carla S. Ellis. Power aware page allocation. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*, pages 105–116, 2000.

[9] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. In *Proceedings of the International Workshop on Compilers and Operating Systems for Low Power*, September 2001.

[10] J. Song, E. Levy-Abegnoli, A. Iyengar, and D. Dias. Design Alternatives for Scalable Web Server Accelerators. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, April 2000.

[11] The New York Times. There's money in housing internet servers. April 2001. http://www.internetweek.com/story/INW20010427S0010.

[12] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. Scheduling for reduced cpu energy. In *Proceedings of the 1st Symposium on Operating System Design and Implementation*, 1994.