Power Management for Heterogeneous Clusters: An Experimental Study

M. Mustafa Rafique[†‡], Nishkam Ravi[‡], Srihari Cadambi[‡], Ali R. Butt[†], Srimat Chakradhar[‡]
[†]*Dept. of Computer Science, Virginia Tech*; [‡]*NEC Laboratories America*
Email: {mustafa,butta}@cs.vt.edu; {nravi,cadambi,chak}@nec-labs.com

***Abstract*—Reducing energy consumption has a significant role in mitigating the total cost of ownership of computing clusters. Building heterogeneous clusters by combining high-end and low-end server nodes (e.g., Xeons and Atoms) is a recent trend towards achieving energy-efficient computing. This requires a cluster-level power manager that has the ability to predict future load, and server nodes that can quickly transition between active and low-power sleep states. In practice however, the load is unpredictable and often punctuated by spikes, necessitating a number of extra "idling" servers. We design a cluster-level power manager that (1) identifies the optimal cluster configuration based on the power profiles of servers and workload characteristics, and (2) maximizes work done per watt by assigning P-states and S-states to the cluster servers dynamically based on current request rate. We carry out an experimental study on a web server cluster composed of high-end Xeon servers and low-end Atom-based Netbooks and share our findings.**

I. Introduction

The cost of powering servers and cooling infrastructure together accounts for more than 30% of the total cost of ownership (TCO) of data centers, while capital expenditure on servers accounts for more than 50% [1], [2] (over a time period of 3 years). In order to reduce TCO, both power and capital expenditure must be reduced by maximizing work done per watt.

Ignoring the unintentional heterogeneity that arises from server refresh, data centers are typically composed of homogeneous servers with similar power-performance profiles. Recently, there have been proposals to deploy low performance energy efficient servers in data centers [3], [4] in an attempt to maximize work done per watt at the server level. For instance, our experiments show that the most recent Atom N550 based servers can yield significantly higher throughput per watt for light web service workloads as compared to Xeon-based servers. However, heterogeneity in data centers is organized across tiers, with a single tier typically consisting of a homogeneous set of servers. This is based on the assumption that for a given workload a homogeneous server configuration would yield satisfactory efficiency, while providing ease of maintenance. For certain kind of workloads, a homogeneous configuration per tier may also correspond to optimal energy efficiency; for others, the results may be suboptimal.

For web service workloads, characteristics of a particular workload together with the client request rate determine the resource and power utilization of the deployed servers. Prior studies [5] have shown that the request rate can vary significantly, with the average CPU utilization for most data center servers varying between 10% and 50% of the peak utilization. In this paper, we observe that a heterogeneous cluster configuration per tier can be more energy efficient than a homogeneous one and that the relative composition can be experimentally determined based on the power-performance profiles of the different servers, characteristics of the workload and the expected variation in request rate.

Dynamic Voltage and Frequency Scaling [6] (DVFS) is a popular power optimization technique. Typically each server node runs a default policy (such as on-demand) that scales the frequency of the processor based on operating system performance counters. The power consumption of the CPU constitutes a portion of the total power consumption of a system; therefore the gains from DVFS (i.e., P-states) are relatively small as compared to low power sleep states (i.e., S-states). However, in order to provision for peak load which is hard to anticipate, and due to high wake-up times, servers are typically kept awake. Idle power for servers is usually more than 50% of the peak power. High idle power together with hard-to-predict load spikes limits the effectiveness of sleep states. In this paper, we observe that performing DVFS at the cluster level and combining it with sleep states yields slightly better energy efficiency than using the either approach in isolation or at the node level. The contributions of this paper are as follows:

- We design a cluster-level power manager that: (1) profiles a given web service workload, (2) determines the optimal cluster configuration for the workload, and (3) implements a policy manager that maximizes throughput per watt for a given QoS by dynamically assigning P-states and S-states to the different servers in the cluster based on current request rate.
- We evaluate the power manager with representative web server applications, and compare it with different node-level and cluster-level power management policies on a cluster composed of Atom and Xeon based servers. Although ACPI-based power saving mechanisms have been well studied, we argue that since the gains from these mechanisms are closely tied to hardware characteristics (such as idle and peak power, availability of P-states and S-states, transition time) as well as the nature of workloads, it is important to periodically re-evaluate them on prevalent or anticipated server configurations in data centers.

978-1-4577-1221-0/11/$26.00

II. Related work

Power management of server clusters is an active area of research. Placing idle servers in sleep states (hibernate/standby) has been shown to be effective in maximizing energy efficiency [7]–[9]. These schemes are often complemented by load consolidation so as to allow servers to go idle. The usefulness of this mechanism is limited by the time it takes to transition machines in and out of the sleep states, since load spikes are often hard to predict. The other downside of workload consolidation is that it leads to the creation of thermal hotspots, which can increase the power consumption of the cooling infrastructure [10]. Despite the cons, use of the sleep states along with the workload consolidation is one of the most effective techniques for energy efficiency in data centers.

DVFS allows the operating system to change the frequency (and consequently the power consumption) of the CPU. Modern operating systems provide default DVFS policies such as on-demand, which vary the CPU frequency based on the measured load using OS counters. Typically data center servers run a local DVFS policy governor, although coordinated frequency scaling (for the entire cluster) has been proposed as an alternative [11]. Since CPU power is a portion of the total power consumption of the systems, the gains from DVFS are often smaller than techniques that cut down on total system power (use of S-states, load distribution/consolidation).

While most prior studies have been carried out in the context of homogeneous clusters, there is some work on the use of DVFS and standby/hibernate in the context of heterogeneous clusters [12], [13]. We argue that since the gains from energy efficiency mechanisms are closely tied to the hardware characteristics (such as idle power, peak power, availability of P-states and S-states) as well as the nature of workloads, it is important to periodically revisit and re-evaluate them on prevalent or anticipated server configurations in data centers.

There have been proposals to incorporate low performance energy efficient servers (based on Intel Atom) in data centers [3], [14]. NapSac [5] demonstrates the benefits of using a centralized policy for predicting workloads and putting machines to sleep in a heterogeneous cluster composed of Atom and Xeon nodes. They study web server workloads and make a case for power-proportional computing.

Our work is closest to NapSac, although there are some key differences. NapSac focuses on the use of standby/hibernate modes along with workload prediction without incorporating DVFS. Also, many of the results are based on simulations. In the server configuration used in NapSac, Xeon nodes are found to be more energy efficient than Atom nodes at low request rates, and the use of Atom nodes is justified by the low transition times (2.4 sec. for wake up from standby). Xeon servers are used for handling average loads while Atom nodes are woken up when the load spikes.

Our experiments are carried out on a cluster composed of Atom N550 and Xeon E5620. Atom N550 is the most recent in the line of Atom processors and the only one to support DVFS. To the best of our knowledge, there is no prior study to explore the benefits of using DVFS in server clusters composed of low performance energy efficient processors such as Atom. In contrast to the results in NapSac, for the same workload (MediaWiki) we find that Atom N550 nodes are more energy efficient than Xeon nodes for processing low request rates and should be used for handling average loads, while Xeon based servers are better suited for handling load spikes. The difference in results is probably due to the significant difference in the power-performance characteristics of Atom 330 (used in NapSac) and Atom N550 (used in our experiments) including idle power, peak power, transition time, performance etc. This justifies researching existing mechanisms in the context of new architectures.

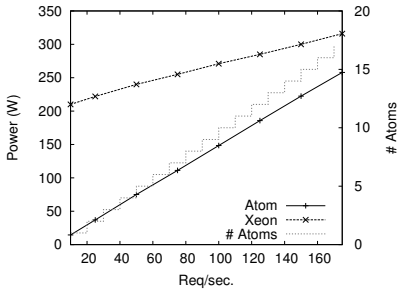
There is some prior effort on combining DVFS and sleep states for clusters [13]. In our design, instead of working with a fixed policy the power manager derives the optimal policy for a given cluster configuration and workload at runtime. The algorithm used for generating the policy simply maximizes throughput per watt. The combination of DVFS and standby mode along with workload consolidation is found to be the most profitable, although the relative gains from DVFS are found to be small (3-6%). In addition, we develop a module for identifying the optimal cluster composition for a given workload.

III. Motivational Data

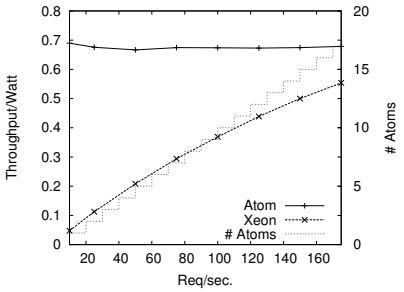
Our experimental heterogeneous cluster consists of Atom N550-based Netbooks and Xeon E5620-based servers. The Atom N550 [15] is the most recent in the line of Atom processors and the only one to support DVFS. It runs at two frequencies: 1.0 GHz. and 1.5 GHz. Table I shows the power-performance profile of Atom N550 with respect to the two web workloads. Table II shows the power-performance profile of the Xeon server. The power range of Xeon (difference between peak and idle power) is much higher as compared to Atom for the two workloads. Similarly the throughput range of Xeon is much higher than that of Atom. This motivates a scale-out approach for Atom's (adding Atom servers incrementally to handle increasing load).

The idle power of Xeon is about 14 times that of the Atom, while the energy efficiency of Atom (throughput per watt) is much higher than that of the Xeon (as shown in Fig. 1(b) and 2(b)). This makes Atom-based servers good candidates for handling light web server workloads.

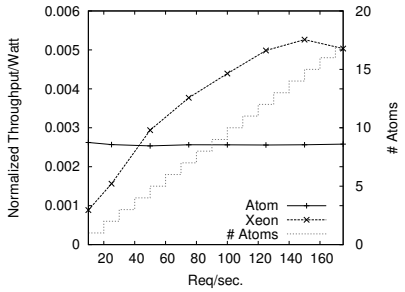
Fig. 1(a) and 2(a) show the raw power consumption of Atom and Xeon as a function of increasing load (requests per second) for the two applications (such that errors/violations=0). As we increase the request rate, a single Atom server cannot sustain the QoS. Therefore, we scale



(a) Power consumption

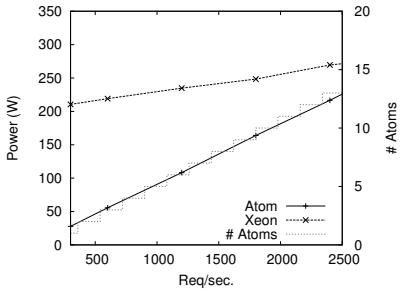


(b) Energy efficiency

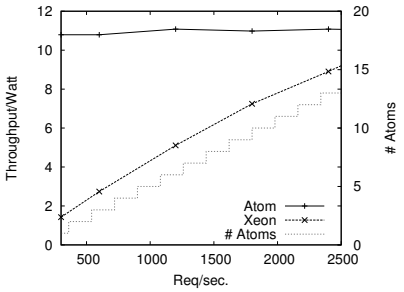


(c) Energy efficiency normalized by response time

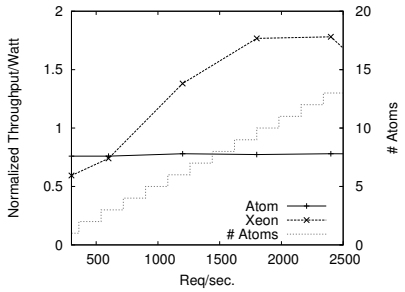
Fig. 1: Atom cluster vs. Xeon for MediaWiki.



(a) Power consumption



(b) Energy efficiency



(c) Energy efficiency normalized by response time

Fig. 2: Atom cluster vs. Xeon for Dynamic Content Server.

Application	Freq. (GHz.)	Peak Power (W)	Req/sec.	Response Time (ms)	Max. Efficiency (throughput/W)
MediaWiki	1.0	12.9	7	391.0	0.54
MediaWiki	1.5	14.5	10	263.0	0.69
Dynamic Content Server	1.0	12.6	105	20.6	8.33
Dynamic Content Server	1.5	14.2	180	14.2	11.2

TABLE I: Power/performance profile of Atom N550 machines for the two workloads, Errors/Violations=0

Application	Freq. (GHz.)	Peak Power (W)	Req/sec.	Response Time (ms)	Max. Efficiency (throughput/W)
MediaWiki	1.6	257.3	110	186.0	0.43
MediaWiki	2.4	316.0	175	110.0	0.55
Dynamic Content Server	1.6	248.5	2250	10.5	9.05
Dynamic Content Server	2.4	281.1	3000	8.9	10.67

TABLE II: Power/performance profile of Xeon E5620 machines for the two workloads, Errors/Violations=0.

out and provision more Atom servers; the number of Atoms provisioned is shown on the right vertical axis. Even with scaling out, the Atoms come out on top in terms of power consumption. When we consider energy efficiency (throughput per watt) shown in Fig. 1(b) and Fig. 2(b), we see that as request rate increases, Xeon starts to catch up with Atom, although the Atom stays ahead.

However, as Table I and Table II show, Atom has a significantly higher latency (response time per request) than the Xeon for both workloads. Since this has implications on both QoS as well energy consumption, we normalize energy efficiency by response time. This is shown in Fig. 1(c) and Fig. 2(c). For low request rates, Atom does better than Xeon on this metric, while the Xeon performs better for high request rates. This motivates heterogeneous clusters for the two workloads (explained in Section IV).

There is a significant difference in the maximum throughput being handled by both Atom and Xeon at low and high clock frequency. The efficiency (throughput per watt) is higher at peak frequency for both. This would suggest that consolidating and directing requests to a few servers and running them at higher frequency would result in good energy efficiency. This would certainly be true if the unused servers could be put into hibernation or standby mode. However, in order to provision for unforeseen load spikes, a certain number of servers would have to be kept awake. The total power consumption in this case would be obtained by adding the power consumed by the heavily utilized servers and the power consumed by the idle servers. The other strategy would be to run the servers at low frequency and distribute the workload among them. In this paper, we compare the differ-

Power State	Atom	Xeon
Idle	11.7	201.1
Standby (S-State)	2.4	24.1
Hibernate	1.0	21.1

TABLE III: Power consumption (in Watts).

ent strategies. Our power manager is able to automatically deduce the correct strategy based on the current request rate and power profiles of the different servers involved (Atom and Xeon in this case).

Minimizing the power spent on idle servers by keeping the minimum number of servers awake for handling spikes would improve energy efficiency [5]. Since the idle power of our Xeon is 14 times that of the Atom, it would be profitable to keep Atom-based servers awake and put Xeon-based servers to sleep. When a spike comes, the idle Atom servers can handle the increasing load, while a Wake-on-Lan signal is sent to Xeon servers. Prior studies [16], [17] have shown that it usually takes a few minutes for a load spike to peak. Xeon servers take around 90 sec. to be brought up from standby. Therefore, in order to plan for load spikes, we need enough idle Atom-based servers to handle the load till the Xeons are brought back into operational state. Our power manager is able to deduce this automatically.

IV. Design

A. Setup

The server cluster is composed of 16 Intel Atom N550 1.5 GHz. nodes each with two cores and 2 GB RAM, and 2 Intel Xeon E5620 2.4 GHz. nodes each with four cores and 48 GB RAM. Both Atom N550 and Xeon E5620 support DVFS, standby and hibernate modes. Atom takes 35 sec. and 90 sec. to wake up from the standby and hibernate modes respectively. Xeon takes 90 sec. and 120 sec. to wake up from the standby and hibernate modes respectively.

For the purpose of informing the design of the power manager, we study two web server workloads. The first workload is a web server application serving dynamic content from the local filesystem. This workload is hosted on a single tier. The second is the MediaWiki+MySQL application. MediaWiki and MySQL are hosted by two separate server tiers; we focus on the front-end tier running the MediaWiki. We studied lightweight workloads to make sure that the QoS/latency is within acceptable bounds. Our experience with Atom and prior studies [18] suggest that Atom is not suitable for computationally intensive workloads.

B. Architecture

Fig. 3 shows the power manager architecture. *Input Handler* receives client requests and stores them in a request queue. *Policy Manager* is the central component. It interacts with the other components and implements the power management policy for the cluster. It reads the requests from the queue, and redirects them to the appropriate server. During the initialization process, *Policy Manager* invokes *Profiler and Cluster Configurator*, which profiles the given workload on the cluster, generates lookup tables and identifies the optimal cluster configuration (described later in this section). *DVFS Driver* remotely sets the appropriate CPU frequency on a given server node. *Standby/Hibernate Driver* implements

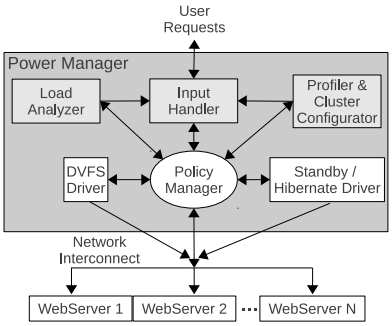


Fig. 3: System architecture.

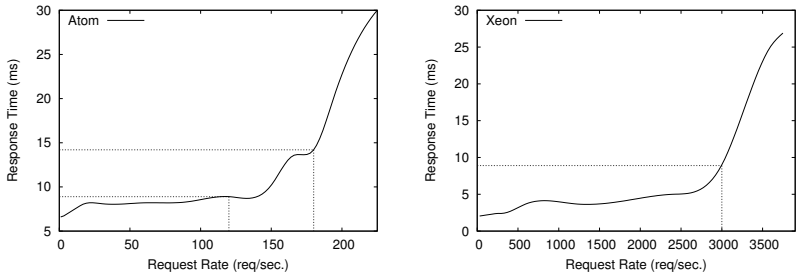
the functionality for putting servers to standby mode and waking them over LAN. *Load Analyzer* interacts with the *Input Handler* and monitors the rate of incoming user requests. The *Policy Manager* periodically polls the *Load Analyzer* to detect any spikes in the incoming user requests.

C. Power Manager

The input to the power manager is workload-independent power-performance characteristics of the different server types in the cluster, such as CPU model, idle power, P-states (CPU frequencies) and S-states (hibernate/suspend). We assume that the characteristics of the workload (e.g., average request rate, peak request rate and transition time) are available to the power manager in the form of traces or otherwise, as well as the SLA. The power manager profiles the workload and derives workload-specific power-performance characteristics for each server, such as power consumption, CPU utilization and response time for different request rates at different CPU frequencies. A lookup table is created for each server type for a given workload.

1) Response Time and Cluster Configuration: The response time knob is crucial in assessing the right composition of a cluster for a given workload. Fig. 4 shows how the response time for the dynamic content server workload varies with the request rate for Atom and Xeon (at peak frequency). The response time increases slowly with increasing request rate and then jumps to a high value (at which point the error rate becomes non-zero). If we assume that 14 ms as response time is acceptable QoS for this particular workload, we find that a cluster composed entirely of Atom nodes would maximize the throughput per watt (as shown in Fig. 2(b)). From Fig. 2(a) it can be seen that around 16 Atom nodes can handle as much load as the Xeon node for this particular workload. Since a Xeon node would typically cost as much as a cluster of 10 Atom nodes or more and the throughput per watt difference is significant, it is safe to say that the TCO for a cluster of 16 Atom nodes would be less than that of a Xeon. The only reason to have Xeon nodes in the cluster at that point would be because the data center already had them before acquiring Atom nodes.

However, if the acceptable response time were 9 ms, the scale-out factor for Atom nodes would increase and so would



(a) Response time for Atom (b) Response time for Xeon

Fig. 4: Response time profiles for Atom and Xeon for Dynamic Content Server.

the capital expenditure as well as power consumption (as suggested in [18]). Fig. 2(c) incorporates the response time factor in the cost analysis. The Y-axis shows throughput per watt normalized by the response time. If the response time for a cluster of Atom nodes were to be matched with that of a Xeon, then for a request rate of more than 50 req/sec. for MediaWiki (and 600 req/sec. for dynamic content server) a Xeon would be preferred over an Atom cluster, while for lower request rates, an Atom cluster would do better (in terms of power consumption), thus motivating a heterogeneous composition. In real life, the QoS would typically be defined by an SLA and the normalization factor would be a function of the response time. Our power manager incorporates the normalization function (response time for now) in deciding the optimal cluster composition/configuration.

In order to identify the optimal cluster configuration, the power manager runs the workload and finds the throughput per watt for different request rates for both Atom and Xeon (such that the number of violations/errors is zero). It notes the corresponding response times. It then divides the throughput per watt by the response time (which is assumed to be the normalization function for the purpose of this paper) and stores the values in a table (one for each server type) indexed by the request rate. From the two tables, it estimates the point at which the normalized throughput per watt values for an Atom cluster would match that of a Xeon (i.e., the intersection point in Fig. 1(c)). If such a point is found then the cluster configuration would be heterogeneous. The number of Atom nodes per Xeon would be equal to the number of Atom nodes needed to handle the request rate at the intersection point while keeping the response time within specified bounds (5 Atom nodes per Xeon for the dynamic content server). If such a point is not found then the cluster configuration is homogeneous. If the normalized values for Atom are higher across the board, the cluster should be composed entirely of Atom nodes. If the normalized values for Xeon are higher then the cluster should be composed entirely of Xeon nodes.

2) Policy Manager: Once the cluster configuration is identified, the policy manager attempts to maximize throughput per watt by assigning P-states and S-states to the server nodes in the cluster for a given request rate such that the response time is within specified bounds and the error rate is zero. In the current implementation, the policy manager only works with the minimum and maximum frequency. That corresponds to 1 GHz. and 1.5 GHz. for Atom N550, and 1.6 GHz. and 2.4 GHz. for Xeon E5620. Only S3 sleep state (i.e., standby) is used. The following values are associated with each server type: P_{idle} (idle power), P_{minf} (peak power for the given workload at the lowest CPU frequency), P_{maxf} (peak power for the given workload for the highest CPU frequency), P_{stdby} (power consumed in standby mode), T_{minf} (max throughput handled at lowest CPU frequency for given response time), T_{maxf} (max throughput handled at highest CPU frequency for given response time). As mentioned before, these values are stored in a lookup table for each server type. For a given request rate, the policy manager tries to find the tuple $\{N_{idle}, N_{minf}, N_{maxf}, N_{stdby}\}$ for each server type (Atom and Xeon in this case) such that throughput per watt represented as:

$$\sum_{servertype} \frac{N_{minf}T_{minf}+N_{maxf}T_{maxf}}{N_{idle}P_{idle}+N_{minf}P_{minf}+N_{maxf}P_{peakf}+N_{stdby}P_{stdby}}$$

is maximized and

$$\sum_{servertype} ((N_{minf} + N_{maxf} + N_{idle}) \times T_{maxf}) - k \times reqRate$$

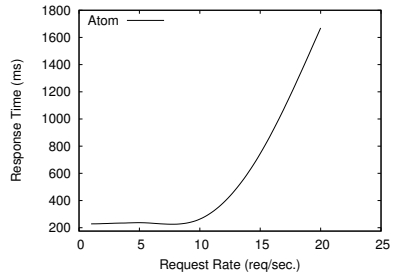
is minimized, subject to the constraints:

$$\sum_{servertype} ((N_{minf} + N_{maxf} + N_{idle}) \times T_{maxf}) - k \times reqRate \geq 0$$

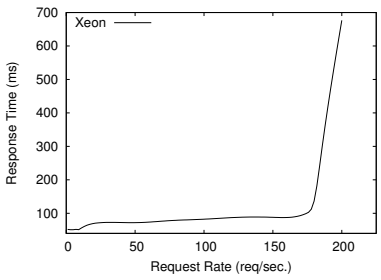
and $N_{idle} + N_{minf} + N_{maxf} + N_{stdby} \leq N_{max}$.

N_{max} is the maximum number of servers of each type available in the cluster. The factor k is determined by the nature of the workload. We set k as 2 in our experiments implying that the cluster is always ready for handling twice the current request rate. k is typically greater than 1 so as to handle unanticipated increase in load.

Under normal operation, the policy manager executes periodically and sets the power states of the servers appropriately. In order to handle load spikes, the policy manager constantly monitors the request rate and whenever it detects a sudden increase in request rate that persists for some time, the standby servers are alerted. The value of k (as mentioned before) is chosen in such a way that there are enough servers to handle the increase in load till the standby servers are ready for work. This design decision is based on the assumption



(a) Response time for Atom



(b) Response time for Xeon

Fig. 5: Response time profiles for Atom and Xeon for MediaWiki.

that most load spikes take a minute or more to peak, which would give enough time for the standby servers to become operational. Once the spike subsides, the policy manager returns to its normal operation.

The power manager automatically derives a policy that uses both DVFS and standby mode. Under normal operation, the Xeon nodes are put to sleep, and the Atom nodes operate at low frequency (with the load distributed among the them intelligently). While a spike detector is used, no explicit workload prediction is carried out.

V. Evaluation

We experiment with two web server workloads: dynamic content server (serving HTML pages from the local filesystem) and MediaWiki+MySQL. We compare the performance of Atom and Xeon nodes for the two applications to understand the power/performance tradeoffs (as shown in Fig. 1, 2) in order to identify the scale-out factor for Atom nodes. The acceptable response time and corresponding request rate handled for both workloads at different CPU frequencies is shown in Tables I, II. The acceptable response times were set so that the throughput is maximized while keeping errors/violations=0 (this happens at the knee of the curve such as the one shown in Fig. 5). For instance, at peak frequency Xeon node starts to drop requests and response time shoots up at about 176 req/sec.

In our evaluation, we focus on understanding the performance of different power management policies for a fixed cluster composition and workload. Since 1 Xeon can handle as much load as about 16-17 Atom machines for the two workloads, we fix the cluster size at 16 Atom netbooks + 1 Xeon server. We provision for twice the peak load, therefore peak is set at the capacity of 1 Xeon/16 Atoms (which is about 3000 req/sec. for dynamic content server and around 160 req/sec. for MediaWiki). 1 Xeon and 32 Atom D510 servers are used for for generating client side requests using *httperf*. Power is measured using Watts Up Pro power meters. Before evaluating the power manager on a cluster of Atoms and Xeon, we try to understand the potential of DVFS on a cluster of Atom nodes alone.

A. Impact of DVFS on Power Consumption of Atom Cluster

Since Atom N550 is the most recent in the line of Atom processors and the only one to support DVFS, we believe this is the first attempt at evaluating DVFS on Atom for web server workloads. For this experiment, we turn the Xeon server off completely and use only 8 Atom nodes. 100% load corresponds to the capacity of 8 Atom nodes i.e., 1500 req/sec. for dynamic content server and 80 req/sec. for MediaWiki. Standby/hibernate modes are not used, therefore all the 8 Atoms are awake at all times. The load is gradually increased from 20-90%. We compare five different policies:

No DVFS: All the Atom nodes run at peak frequency (1.5 GHz.). No power management policy is used; the load is equally distributed among all the nodes.

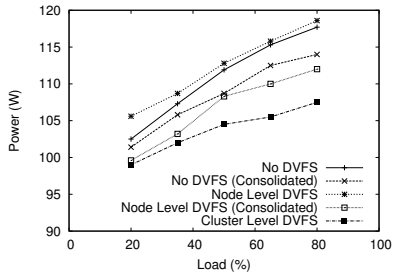
No DVFS (Consolidated): All the nodes run at peak frequency. The load is consolidated and directed to the fewest number of nodes in the cluster.

Node Level DVFS: The default Linux policy governor (on-demand) is activated on all Atom nodes; each node is responsible for scaling its frequency based on CPU utilization. The load is equally distributed among the nodes.

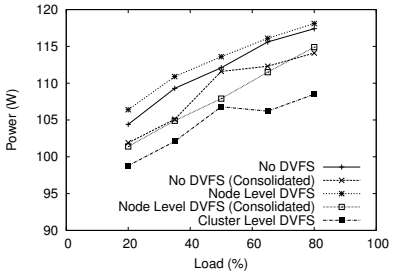
Node Level DVFS (Consolidated): This policy is similar to the previous one, however the input requests are consolidated and directed to the fewest number of nodes possible.

Cluster Level DVFS: All the Atom nodes are initialized to low frequency (1 GHz.). The load is balanced among the nodes. The frequency of a node is scaled up only when the capacity of the entire cluster at low frequency is saturated, which would happen when the load exceeds around 60%, since the maximum capacity of an Atom at low frequency is about 60% of the capacity at high frequency.

Fig. 6 shows the power consumption of the cluster with respect to MediaWiki and Dynamic Content Server. Both applications show similar power consumption trends for the five policies. Interestingly enough, *Node Level DVFS* is the least power efficient among the five while *Cluster Level DVFS* comes out on top. Workload consolidation also helps: we observe an average improvement of 2.5% between *No DVFS* and *No DVFS (Consolidated)*, and 4.6% between *Node Level DVFS* and *Node Level DVFS (Consolidated)*. We observe an average gain of 6.5%, 4.3%, 7.6%, and 4% when using *Cluster Level DVFS* as compared to *No DVFS*, *No DVFS (Consolidated)*, *Node Level DVFS* and *Node Level DVFS (Consolidated)* respectively. Although the relative gains with *Cluster Level DVFS* are small (4-7%), they could translate

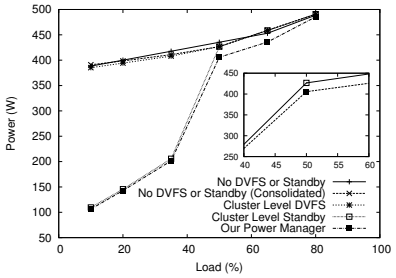


(a) MediaWiki

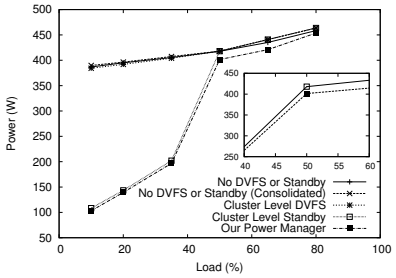


(b) Dynamic Content Server

Fig. 6: Evaluation of DVFS on Atom cluster.



(a) MediaWiki



(b) Dynamic Content Server

Fig. 7: Power consumption with different power policies under increasing input load using heterogeneous cluster. DVFS+Standby gives an additional 3-4% savings as compared to Standby alone.

to a few hundred thousand dollars to a corporation in energy savings per year.

B. Evaluation of Power Manager

We now evaluate our power manager on a cluster of 16 Atom nodes and 1 Xeon server. The power manager implements a meta-policy, which is to assign P-states and S-states to the cluster servers such that throughput per watt is maximized. In our design, the use of P-states and S-states is optional not mandatory. The policy generated by the power manager is compared against other well known policies:

No DVFS or Standby: All nodes run at peak frequency, no power management is carried out. The load is distributed among the nodes in the cluster.

No DVFS or Standby (Consolidated): All the nodes run at peak frequency. The load is consolidated and directed to the fewest number of nodes in the cluster.

Cluster Level DVFS: Described in Section V-A.

Cluster Level Standby: All the nodes run at peak frequency. The load is consolidated and directed to the fewest number of nodes in the cluster. The remaining nodes are put into standby mode.

As described in Section IV, in order to be able to sustain sudden load increases, when the request rate is r req/sec., the cluster should be prepared to handle $2 * r$ req/sec. (value of $k = 2$). The policy generated by the power manager uses a combination of P-states and S-states. At any given point in time, some of the nodes are in standby mode, some are idle, some are running at low frequency and others are running at peak frequency.

Fig. 7 shows the power consumption of the policy generated by the power manager at equilibrium point. We find that when the load is low: 10-40%, the Xeon is in standby mode along with some of the Atom nodes, while others operate at either low or high frequency. When the load exceeds 50% all the standby nodes in the cluster are alerted. Note that the power consumption of the cluster suddenly goes up when the load exceeds 50%, which is due to the waking up of Xeon. As evident from Fig. 7, the gains from standby are most significant when the load is $< 50\%$. Due to the scale of Fig. 7, the power consumption curve of our policy manager seems to coincide with that of *Cluster Level Standby*. A closer look (as shown in the nested graph) reveals that there is a 3-4% net average gain with our policy manager when the load is $< 40\%$, which is due to DVFS. For higher load ($> 50\%$), the different policies tend to converge, and the gains from our policy manager (relative to Cluster Level Standby) become more pronounced (around 6%).

1) Workload Emulation: In order to evaluate the power manager in the presence of load spikes, we emulate a web server workload as shown in Fig. 8. The request rate is varied such that the load is between 30-50% for about 27 minutes, around 95-100% for about 6 minutes, 195-200% for about 9 minutes and the remaining 8 minutes are spent in between. Taking a cue from prior studies, we model the spikes such that it takes 90 sec. or more from the time the spike occurs till it reaches the peak. This gives enough time for the standby servers to wake up. Note that this workload pattern will not benefit our power manager, which yields higher energy savings when the load is between 50-80% (Fig. 7). The workload emulation is meant to stress test the power manager.

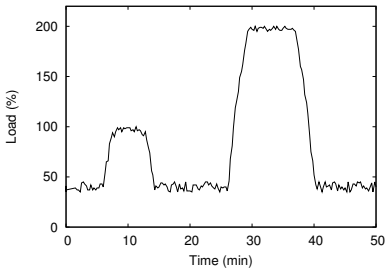


Fig. 8: Generated workload.

Power Management Scheme	Energy (kJ)	Energy Savings (%)
No DVFS/Standby (Consolidated)	413.5	0
Cluster Level Standby	304.9	26.2
Our Power Manager	295.2	28.6

TABLE IV: Energy consumption with different power management schemes for MediaWiki for the generated workload. Our power manager yields 3.2% improvement relative to Cluster Level Standby.

We measure the total energy consumed by the cluster for MediaWiki application with the generated workload. Table IV shows the energy savings obtained with our power manager and *Cluster Level Standby* as compared to the baseline: *No DVFS or Standby (Consolidated)*. The relative gain from the power manager with respect to the baseline is about 28.6%. The relative gain with respect to *Cluster Level Standby* is about 3.2%.

VI. Conclusion

In conclusion, we find that the composition of a cluster depends on the workload characteristics and the characteristics of the servers involved–the decision of whether or not a cluster should be heterogeneous and if so what the right composition should be can be automated as part of cluster/power management. We find that Atom N550 nodes are more energy efficient than Xeon nodes for processing low request rates and should be used for handling average loads, while Xeon based servers are better suited for handling load spikes. We conclude that working with a generic policy manager that maximizes throughput per watt is beneficial. Our policy manager was able to generate the policy that yields maximum energy savings. The generated policy suggests the use of cluster-level DVFS and standby mode. The relative gains from DVFS were found to be small (3-6% in our setup). Although, 3-6% energy savings might translate to a few hundred thousand dollars per year for a corporation. As compared to the baseline, our generated policy shows significant energy savings (28.6%) for the generated workload.

In summary, we have presented the design of a power manager that finds the optimal cluster configuration for a given workload and then goes on to maximize the work done per watt by assigning P-states and S-states to cluster servers dynamically based on current request rate. We have evaluated the power manager on a cluster composed of Atom and Xeon servers for lightweight web server workloads. To the best of our knowledge, this is the first attempt at exploring the potential of frequency/voltage scaling on low performance high efficiency mobile processors such as Atom in the context of web servers.

We plan to extend this work to handle frequency scaling for GPUs with a different set of workloads. We also plan to understand the implications of a wider frequency range and more P-states on Atom processors using simulations. As it has been noted before, the significant gains will come from reducing idle power, minimizing transition time and/or being able to predict load spikes.

Acknowledgment

This work is supported in part by NSF (CNS-1016408). M. Mustafa Rafique is partially supported by a scholarship from the Fulbright Foreign Student Program.

References

[1] APC-American Power Conversion, "Determining Total Cost of Ownership for Data Center and Network Room Infrastructure," 2003, http://www.apcmedia.com/salestools/CMRP-5T9PQG_R2_EN.pdf.

[2] U.S. Environmental Protection Agency, "Report to congress on server and data center energy efficiency," Aug. 2007.

[3] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan, "Fawn: a fast array of wimpy nodes," in *Proc. ACM SOSP*, 2009.

[4] K. Lim, P. Ranganathan, J. Chang, C. Patel, T. Mudge, and S. Reinhardt, "Understanding and designing new server architectures for emerging warehouse-computing environments," in *Proc. IEEE ISCA*, 2008.

[5] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. Katz, "Napsac: design and implementation of a power-proportional web cluster," *SIGCOMM Comput. Commun. Rev.*, vol. 41.

[6] I. Corporation, "Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor," March 2004.

[7] R. Bianchini and R. Rajamony, "Power and energy management for server systems," *Computer*, vol. 37, Nov 2004.

[8] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, *Dynamic cluster reconfiguration for power and performance*. Kluwer Academic Publishers, 2003, pp. 75–93.

[9] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *Proc. ACM ASPLOS*, 2009.

[10] F. Ahmad and T. N. Vijaykumar, "Joint optimization of idle and cooling power in data centers while maintaining response time," in *Proc. ACM ASPLOS*, 2010.

[11] E. N. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," in *Proc. PACS*, 2003.

[12] T. Heath, B. Diniz, E. V. Carrera, W. Meira, Jr., and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *Proc. ACM PPoPP*, 2005.

[13] C. Rusu, A. Ferreira, C. Scordino, and A. Watson, "Energy-efficient real-time heterogeneous server clusters," in *Proc. IEEE RTAS*, 2006.

[14] B.-G. Chun, G. Iannaccone, G. Iannaccone, R. Katz, G. Lee, and L. Niccolini, "An energy case for hybrid datacenters," *SIGOPS Oper. Syst. Rev.*, vol. 44, March 2010.

[15] I. Corporation, "Intel Atom Processor N550," Jan 2011, http://ark.intel.com.

[16] P. Bodik, A. Fox, M. J. Franklin, M. I. Jordan, and D. A. Patterson, "Characterizing, modeling, and generating workload spikes for stateful services," in *Proc. ACM SoCC*, 2010.

[17] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Proc. IEEE IISWC*, 2007.

[18] W. Lang, J. M. Patel, and S. Shankar, "Wimpy node clusters: what about non-wimpy workloads?" in *Proc. ACM DaMoN*, 2010.