Understanding the Critical Path in Power State Transition Latencies

Sam (Likun) Xi, Marisabel Guevara‡, Jared Nelson, Patrick Pensabene, and Benjamin C. Lee‡

Systems Architecture Integration Laboratory. Pratt School of Engineering, Duke University
‡ Corresponding authors: mg@cs.duke.edu, benjamin.c.lee@duke.edu

***Abstract*—Increasing demands on datacenter computing prompts research in energy-efficient warehouse scale systems. In one approach, server activation policies invoke low-power sleep states but the power state transition latency must be small to produce effective energy savings. Chrome OS and Arch Linux require 50ms and 650ms, respectively, to enter sleep states. These states consume merely 4–6% of nominal power. By analyzing the critical path, we propose strategies for selecting hardware components and optimizing kernel resume sequences to make datacenter server activation viable. With fast transitions, server activation can provide better performance at lower energy than dynamic voltage and frequency scaling.**

Keywords*—*Transition latency, suspend, sleep, Chrome OS, power consumption, datacenter

I. Introduction

Datacenter power consumption has rapidly grown by 36% in the five year period following 2005. As of 2010, datacenters use 2% of the energy consumed in the United States and 1.3% of the energy consumed around the world [12]. In the USA, this amounted to 77.5 billion kWh consumed [5]. A large portion of this energy is spent powering idle servers performing little useful work. Even at idle, servers typically draw about 60% of peak power. Average datacenter server utilization is $20 - 30\%$, which means a huge amount of power is consumed with no gain [10].

There is a large body of research on reducing the energy consumption of datacenters. Transitioning servers into a low-power sleep state during idle is one approach to improving the energy proportionality of large scale systems [6], [10]. The effectiveness of an activation policy depends largely on the latency to transition between active and sleep power states.

What are contributors to power state transition latencies? Prior work identifies opportunities to reduce transition latencies via hardware choices, e.g. mobile instead of server components [13]. In this work, we consider software contributors to power state transition latencies. In particular, we analyze power states in standard and mobile-class operating systems to draw lessons for datacenter servers.

We find that stock Arch Linux running on a high end custom built desktop has transition latencies of 650ms, which we deem too slow for sleep state power policies. Alternatively, Chrome OS is an operating system that has been optimized for fast boot and resume, and we observe that transition latencies of 50ms are achievable. This sub-100ms latency is instrumental for server activation policies and datacenter energy efficiency.

We evaluate power mode responsiveness in the context of PowerNap, a server activation policy proposed in prior work [10]. By studying two machines and operating systems, we determine that it is feasible to put a server into sleep mode during idle periods and wake it up quickly enough to achieve significant energy savings with minimal impact on performance and response time. Indeed, performance and power trade-offs are often better than those from highly responsive dynamic voltage and frequency scaling.

The rest of this paper is structured as follows. In Section 2, we discuss related work and server sleep states. Sections 3 and 4 describe our experimental methodology and results, respectively. In Section 5, we provide a discussion about the results and their applicability to server sleep states. Finally, we conclude in Section 6.

II. Background

The large variation in transition latencies measured in prior work motivates our study. Prior measurements range from milliseconds to hundreds of seconds across desktop or mobile platforms.

Server Activation. Putting servers into sleep states during activity troughs is a common approach to reducing datacenter power consumption. Building energy-proportional systems at this scale is a crucial part of improving datacenter energy-efficiency [3]. An energy-proportional system consumes power in proportion to its workload. Server activation can improve the energy efficiency of a system given the right system features and a power management policy.

Meisner et al. propose PowerNap, a policy for transitioning servers into sleep states [10]. They compare its power-conserving ability over a range of CPU utilizations and transition latencies, finding that sub-100ms latencies are necessary to provide substantial power savings. However, the PowerNap policy optimistically assumes that a 10ms latency is easily achievable; in practice, this is actually quite difficult.

Gandhi et al. propose a variety of power management policies, and characterize the performance-per-watt (PPW) of each over a range of idle power consumption and setup times [6]. This study shows that transition policies can provide improvements in PPW with transition latencies of 20-50s. Alternatively, Agarwal et al. propose an approach to service a small

number of tasks while a server is in a sleep state [2]. A low power processor on the network interface card allows the system to maintain a low power sleep state until the computational power of the main processor is required again.

Server Sleep States. The Advanced Configuration and Power Interface (ACPI) specification defines a standard for an operating system to perform power management of hardware components [1]. Server activation policies use the sleeping state S3, also known as suspend-to-RAM. In S3 sleep, the processor context, cache contents, and chipset context are all lost, but main memory remains powered. We focus on the S3 state, which provides a balance between power savings and resume time as everything but main memory is powered down.

In contrast, the S2 state only powers down the processor; in modern servers, CPUs account for merely 33% of total power usage [4]. On the other hand, the S4 "hibernate" sleep copies DRAM content to the hard disk and the system is completely powered down. Although S4 achieves greater power savings, DRAM data must be restored upon resume thus incurring an expensive transition latency [10].

Using a power mode incurs a sleep and a resume latency. We focus our analysis on resume latency as this, to a large degree, determines the response time of a datacenter using a power state transition policy. Further, if the resume latency is fast enough, long sleep latencies can be hidden, because if a query arrives when a server is going to sleep, another server can resume and service it. However, for the sake of completeness, sleep latency is briefly discussed.

III. Methods

Kernel Logging. Linux kernels post version 3.6 can report individual device resume times during power state transitions. We enable this functionality with the command `echo 1 > /sys/power/pm_print_times`. Suspend-to-RAM was manually initiated by the command `echo ''mem'' > /sys/power/state`, and resume was initiated by pressing an appropriate key. We capture the contents of the kernel ring buffer for several iterations of the sleep and resume sequence. We then analyze the sequences to find individual device resume times and trends of interest.

To measure power, we use a watts up PRO^{TM}watt meter. The meter plugs into the wall outlet and the device's power supply plugs into the meter. For each device, we initiate sleep and wakeup as described above. The meter samples power usage at 1Hz, which is the highest timing resolution. Finally, we retrieve the data from the meter via USB.

Testing Platforms. We study the transition latencies of two systems: a Samsung Chromebook Series 5 550 and a custom built Arch Linux desktop (Table I). The Arch Linux platform provides a baseline for comparing latencies and orderings of kernel components during system resume. We could not test our blade server because the BIOS did not support S3 sleep.

For comparison, we consider the potential for mobile-class operating systems in datacenter servers. Chrome OS is a lightweight operating system developed by Google which run on laptops called Chromebooks [7]. Google touts fast boot speeds (∼7s) and resume speeds as a major feature of Chrome OS. Our detailed study of the optimized resume sequence in Chrome OS is motivated by its low transition latency from hibernate to active. Coupled with mobile hardware, the Chromebook is a system with the fastest transition latency out of x86-capable systems.

TABLE I. Hardware configurations of test platforms.

Linux Desktop	Chromebook
Intel Core i7-3930K, 3.2GHz	Intel Celeron 867, 1.3GHz
16GB DDR3 RAM	4GB DDR3 RAM
NVIDIA GTX 580	Intel integrated graphics
160GB HDD	16GB mSATA

IV. Experimental Results

Arch Linux. The Arch Linux desktop takes 680ms to suspend all devices and enter S3 sleep. The desktop takes about 650ms to resume from sleep, which can be broken down into three main stages: CPU cores (72ms), no interrupt (184ms), and the rest of the devices (∼ 400ms). Unlike devices which can be suspended in parallel, each core is powered down one by one. We found similarly long core sleep times on other very different Linux machines, suggesting that it is a result of software rather than hardware. The timeline for the 40 devices that took the longest to resume is shown in Figure 1. The devices that are not shown here have fast resume latencies and do not affect the critical path of the power state transition.

Resume begins by waking all CPU cores, each of which takes 12ms (not shown on the timeline). This is followed by a `noirq` (no interrupt request) stage, which allows critical components to be woken up without simultaneously resuming others and potentially creating synchronization conflicts. The `noirq` stage cannot be parallelized as that would defeat its purpose, and thus it takes a considerable amount of time. Kernel logs state that the `noirq` stage requires 184.2ms to complete. After this stage, PCI devices are brought up simultaneously. SATA links follow; they take the longest at over 300ms each, and their attached devices are resumed immediately afterwards. For instance, the attached HDD took about 50ms to resume.

When all the devices have been brought up, the root partition of the filesystem is remounted (0.5s) and then the Ethernet link is re-established (2.5s). These are not hardware components and thus do not appear on the timeline, but they are required. The 0.5s latency in remounting the filesystem can be reduced by switching out the HDD for an SSD; in fact, kernel

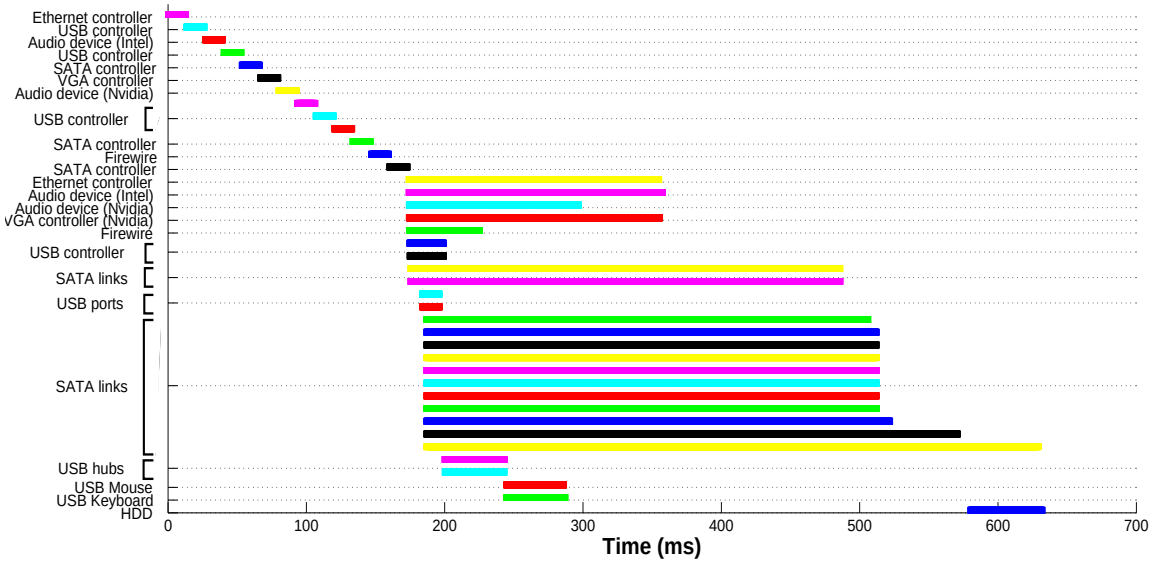

Figure 1. Resume sequence timeline of the Arch Linux desktop. A breakdown of the 650ms resume latency shows that the primary contributors are CPUs (72ms), noirq (184ms), and additional devices (400ms).

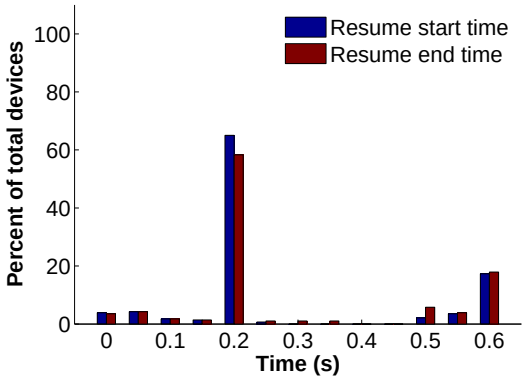

Figure 2. Resume sequence start and finish time distribution on the desktop. Most devices begin resuming around 200ms after the system wakes up and finish resuming within 50ms.

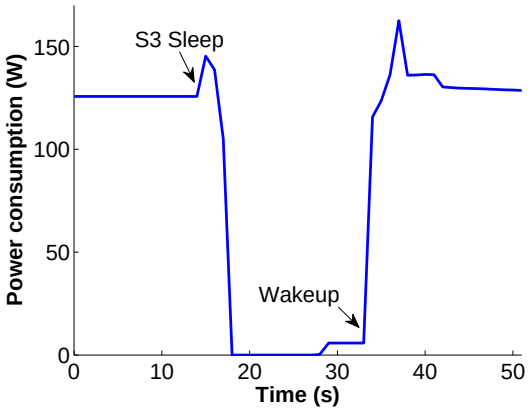

Figure 3. Desktop power consumption while idling and sleeping. The hardware setup causes the reported sleeping power consumption to appear to be 0W, but S3 sleep actually consumes 5W, shown by the jump in power to 5W *before* the wake up signal.

logs on Linux machines with SSDs did not indicate any latency in remounting the filesystem.

We can avoid the costly latency of re-establishing a network link given that servers in a datacenter rarely migrate across networks. Since a network link is essentially a valid IP address, a server that is woken up before its IP address expires (typically on the order of hours) can continue to use the old address. The network driver can be updated to not return its IP address to the available pool of addresses as part of the suspend sequence. Alternatively, a NIC equipped with an embedded processor can provide continuous operation for networked applications while the rest of the platform sleeps [2].

The timeline indicates little opportunity to optimize the resume sequence. By removing components unnecessary to a server (i.e. audio cards, video ports, etc.), the `noirq` stage can be optimistically shortened to 100ms. However, an optimized resume latency cannot be less than 300ms because of the SATA links, which are the slowest individual devices.

Figure 2 shows the histogram of resume start and completion times for each device on Arch Linux. Most devices initiate resume 200ms after the first device and about 20% of devices complete resume at 600ms. Figure 1 demonstrates that most devices complete resume in under 50ms. Thus, any optimization of sleep state responsiveness must address the few devices with long resume latency.

Finally, power measurements shown in Figure 3 demonstrate that the desktop consumes around 130W when idling in a fully powered state compared to a mere 5W in S3 sleep, 3.8% of nominal.

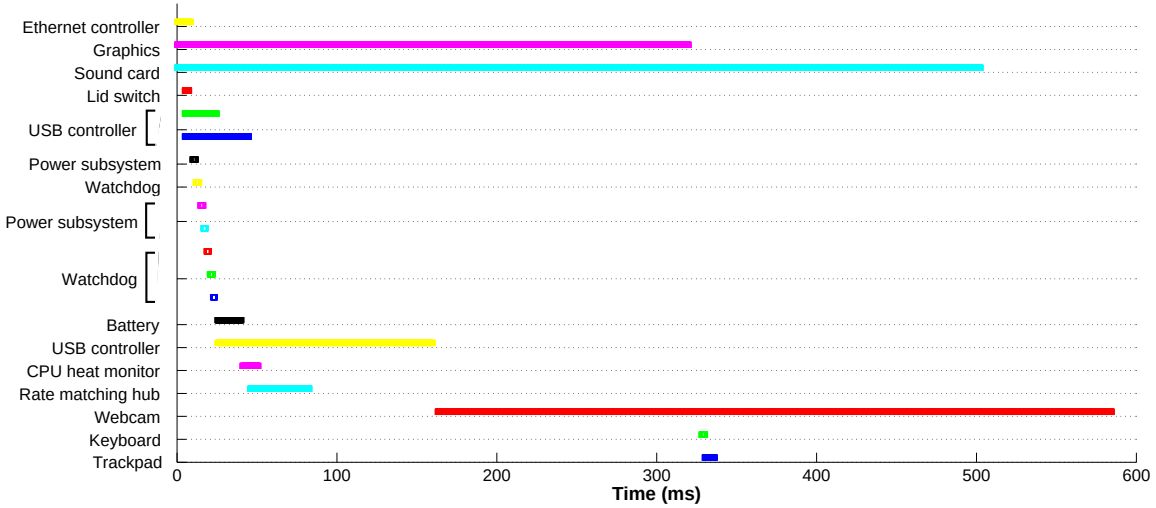

Figure 4. Resume sequence timeline of the Chromebook. A breakdown of the overall 600ms resume latency shows that the primary contributors are the graphics system (300ms), sound card (500ms), and webcam (450ms).

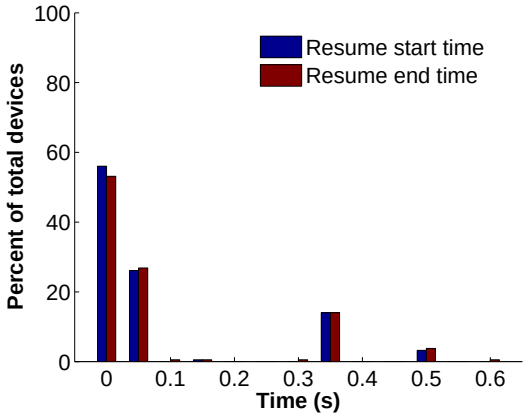

Figure 5. Resume sequence start and finish time distribution on the Chromebook. Most devices begin and finish their resumes within the first 100ms of the overall resume sequence.

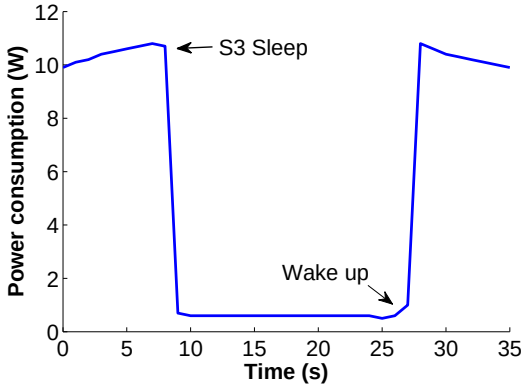

Figure 6. Chromebook power consumption while idling and sleeping. In sleep, only 5% of nominal power is consumed.

Chrome OS. The Chromebook suspends all of its devices in 45ms and then its two physical cores at 2ms each for a total of 49ms. The Chromebook uses Intel Celeron CPUs, whose microarchitecture is much simpler than that of the Core i7 used in the desktop and explains why the Chromebook's sleep latency is so much shorter.

The Chromebook takes about 600ms to resume all of its devices. The timeline of the critical path for the resume sequence is shown in Figure 4. In stark contrast to the desktop, the Chromebook's `noirq` stage takes only 1.74ms; essentially all of the 600ms is spent on simultaneously resuming the components. We find that the wireless network card takes much longer to resume than the the Gigabit Ethernet card (data not shown) typically used in servers.

We can reduce the resume latency to around 50ms with some optimizations. First, note that the sound card[1] and graphics subsystems take up the majority of the 600ms latency, but they are generally not critical or needed in servers (certain applications may benefit from dedicated multimedia processing hardware). This devices can either be removed entirely or have their resume sequences delayed until needed.

Also, observe that USB host devices take roughly 50-100ms to resume, but servers are unlikely to have peripheral devices attached over USB. Thus, USB hosts can have their resume sequences delayed until they are required (if at all). Other irrelevant components like the webcam, lid switch, trackpad, and keyboard can be stripped out entirely.

With these optimizations, resume latency drops to around 50ms. Kernel logs for the Chromebook did not report any latency for remounting filesystems, which we attribute to the use of a SSD instead of a HDD for local storage. Re-establishing a network connection

[1]The sound card takes a long time to resume because sleeps are inserted in its resume sequence to prevent it from making popping sounds.

takes 0.95s, yet servers in a datacenter can avoid requesting a new IP address by the same techniques mentioned for the Arch Linux system.

Figure 5 shows the distribution of all device resume start and completion times on the Chromebook in time bins of 50ms. Nearly 80% of all devices begin resuming within 75ms of the first device and nearly all complete their resume sequences within 50ms. The other 20% of devices begin and finish resume around 300ms, and by examining the complete device resume timeline (not shown), we see that these devices are waiting on the graphics subsystem. If we eliminate such long latency devices, we can either eliminate the devices that are waiting or allow them to begin their resume sequence earlier, thus shortening the overall resume latency.

Finally, power measurements shown in Figure 6 demonstrate that the Chromebook draws on average 10W while idling in a fully powered state, but in S3 sleep, it drops to 0.6-0.7W, which is merely 6% of full power consumption.

V. Analytical Results

Let's consider the implications of fast response times. We evaluate a system using PowerNap, a policy for server activation [10]. In this prior work, Meisner et al. conclude that a transition latency on the order of 10ms is required to achieve significant power savings and performance improvements over a system using DVFS. We apply the same methodology to evaluate the Chromebook and Arch Linux desktop as PowerNap systems with 50ms and 650ms latencies, respectively. For comparison, our evaluation includes data for an optimistic 10ms latency.

Meisner et al. use analytical models to approximate performance and power as a function of power state transition latencies and system utilization. The performance model estimates response time with an M/G/1 queue with exceptional first service time, which accounts for the power state transition latency for the first task that arrives after an idle period. The power model simply accounts for the fraction of time in each power state.

We refer the reader to PowerNap for detailed analytical models [10]. With these models, we compute average power and response time for systems that have different transition latencies: Generic PowerNap $T_t = 10\text{ms}$, ChromeOS PowerNap $T_t = 50\text{ms}$, and Linux Desktop PowerNap $T_t = 650\text{ms}$. Response times are relative to a system with no power management policy.

We further compare against dynamic voltage and frequency scaling (DVFS). In comparison to server activation, DVFS is very responsive (e.g., microseconds) but offers limited benefit. Only processor power is modulated and we estimate that processor power accounts for only 33% of the total [4]. With sufficiently fast resume times, server activation policies may offer more attractive performance-power tradeoffs than DVFS.

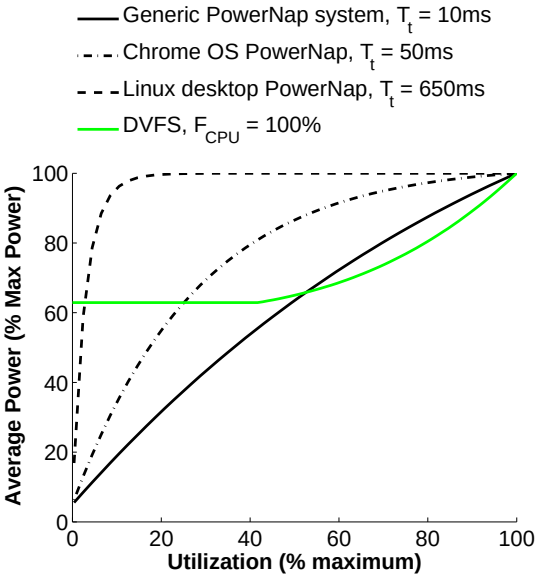

Figure 7. Power scaling comparison between PowerNap and DVFS. The Chrome OS system's power consumption is less than that of DVFS for typical server utilization levels, whereas the Arch Linux system's resume latency is too slow to be of any benefit.

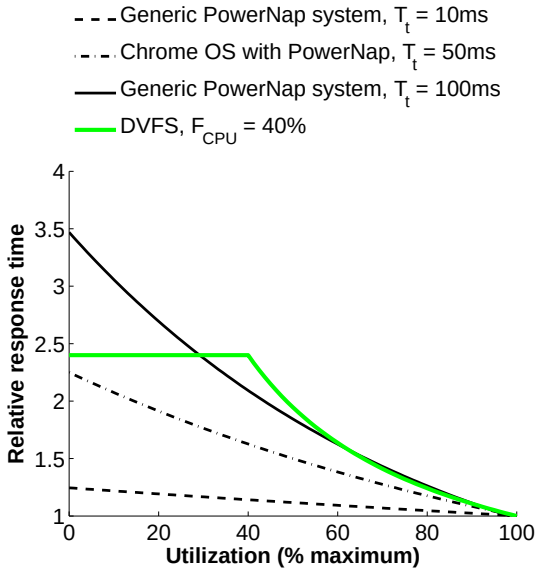

Figure 8. Response time scaling comparison between PowerNap and DVFS. The Chrome OS system has faster relative response times than that of DVFS at all utilizations, but beyond $T_t = 50\text{ms}$, DVFS will provide faster response times at low utilizations.

In Figure 7, we find that PowerNap with transition latency of $T_t = 650\text{ms}$ is highly inefficient, consuming less power than DVFS only for near-idle systems (e.g., utilization below 4%). However, at $T_t = 50\text{ms}$, PowerNap is far more competitive, consuming less power than DVFS even as utilization increases to 35%. Furthermore, if T_t drops to 10ms, then PowerNap's power advantage over DVFS increases up to 45% utilization. Because most datacenter servers

operate under 50% utilization [3], a 10ms latency would be ideal. But we show that a 50ms latency is highly competitive and viable with the optimizations to device and kernel resume sequences.

In terms of performance, Figure 8 indicates that PowerNap will always be slower than DVFS given long resume latencies. $T_t = 650$ms is not competitive and $T_t = 100$ms is only competitive when system utilization is above 30%.[2] As utilization increases, $T_t = 100$ms converges to DVFS. However, at $T_t = 50$ms, PowerNap relative response times are *always* faster than that of DVFS, regardless of utilization.

If we could reduce T_t to 10ms, we could further reduce response times. When utilization is less than 45%, the point at which PowerNap consumes more power than DVFS, response times for $T_t = 10$ms are 28% to 44% lower than response times for $T_t = 50$ms. Our results indicate that ChromeOS is good candidate for implementing PowerNap, whereas generic Arch Linux is not.

These findings motivate further research into optimizing server operating systems for fast transition latencies and specific hardware configurations. ChromeOS is in fact a version of Linux engineered for these targets. As blade servers in a datacenter tend to be identical, from an energy-efficiency perspective it makes sense to optimize their software stacks for specific configurations. Server operating systems exist in many forms, but typically performance takes precedence over energy efficiency in their design. Furthermore, since most datacenters never shut down servers unless absolutely necessary, many servers do not support sleep states in the first place.

Our work shows that optimizing transition latencies in the operating system can significantly reduce power consumption and improve response times for datacenters. This is a first step in studying the impact of mobile computing design choices, such as fixed hardware configurations, low-power sleep states, and fast transition latencies, on server applications.

Many of our conclusions are derived from software-based optimizations, so even though we collect measurements from mobile hardware instead of exclusively desktop/server-class hardware, these conclusions will generalize regardless of the choice of hardware. For instance, Chrome OS, a Linux based system, demonstrates that it is possible to reduce Linux power state transition latency to 50ms. Even though it runs on mobile-class hardware, there is ongoing research in using mobile-class hardware for servers [8] [9] [13].

VI. Conclusion

We present experiments that support the use of server activation policies in datacenters. On Chrome OS, we measure sleep and resume latencies of 50ms each. On a Arch Linux desktop, we observe sleep and resume latencies of 680 and 650ms, respectively. Sleeping systems consume merely 4-6% of the power consumed in an idling but fully powered state.

Using PowerNap, we find that ChromeOS can substantially reduce power consumption and improve performance over an equivalent system using DVFS for dynamic power management, but Arch Linux transition latencies are too slow to provide any appreciable benefits. These results demonstrate that server activation policies for datacenter servers are a promising route towards reducing power with modest performance impact.

[2]With data from Meisner et al. we extrapolate the 50ms response time curve reproduced his figures with estimated fits.

Acknowledgments

This work is supported by NSF grant CCF-1149252 (CAREER) and by STARnet, a Semiconductor Research Corporation program, sponsored by MARCO and DARPA. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of these sponsors.

We would like to thank David Hendricks, Sameer Nanda, Sonny Rao, Hung-Te Lin, and Liam McLoughlin from the Chrome OS team at Google.

References

[1] *Advanced Configuration and Power Interface Specification*. Rev. 5.0, 6 Dec 2011.

[2] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta. "Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage", *Proc. 6th NSDI*, Apr 2009.

[3] L. A. Barroso, U. Holzle. "The Case for Energy-Proportional Computing". *IEEE Computer* 40.12 (2007), pp. 33-37.

[4] L. A. Barroso, U. Holzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2009.

[5] "Electricity Consumption Data, USA". Index Mundi, n.d. Web. 3 Dec 2012.

[6] A. Gandhi and M. Harchol-Balter. "Are sleep states effective in datacenters?", *International Green Computing Conference*, pp. 1-10, Jun 2012.

[7] Google Inc. "The Chromium Projects: Chromium OS". http://www.chromium.org.

[8] M. Guevara, B. Lubin, B. C. Lee. "Navigating heterogeneous processors with market mechanisms", *Proc. 19th HPCA*. Feb 2013.

[9] K. Malladi, I. Shaeffer, L. Gopalakrishnan, D. Lo, B.C. Lee, M. Horowitz. "Rethinking DRAM power modes for energy proportionality", *Proc. 44th MICRO*. Dec 2012.

[10] D. Meisner, B.T. Gold, and T.F. Wenisch. "PowerNap: Eliminating Server Idle Power", *Proc. 14th ASPLOS*, pp. 205-216, Mar 2009.

[11] D. Meisner, C.M. Sadler, L. A. Barroso, W-D Weber, T. F. Wenisch. "Power Management of Online Data-Intensive Services", *Proc. 38th ISCA*, pp. 319-330, Jun 2011.

[12] R. Miller. "Report: Data Center Energy Use is Moderating", *Data Center Knowledge*. N.p., 2011.

[13] V. J. Reddi, B. C. Lee, T. Chilimbi, and K. Vaid. "Mobile Processors for Energy-Efficient Web Search", *ACM Transactions on Computer Systems* 29.4 (2011).