Vision: The Case for Context-Aware Selective Resume

Eric J. Wright
The University of Toronto
ejwright@cs.toronto.edu

Eyal de Lara
The University of Toronto
delara@cs.toronto.edu

Ashvin Goel
The University of Toronto
ashvin@eecg.toronto.edu

ABSTRACT

The main approach for conserving energy today is to place idle system into one of several low-power system sleep states. However, current transition times between the power states are long, limiting the usefulness of these states. We propose using *a context-aware selective resume* to wake a system with only the minimal set of devices needed for the waking task. This approach provides access to system resources with the lowest power consumption possible and with the shortest transition latencies. In this paper, we discuss the classes of applications that would benefit from selective resume, discuss the design considerations for implementing selective resume, and profile system sleep cycles to demonstrate that OS modifications can reduce cycle time by as much as 87% and energy use by up to 50%.

Categories and Subject Descriptors

D.4.0 [**Software**]: OPERATING SYSTEMS—*General*; C.0 [**Computer Systems Organization**]: GENERAL—*System architectures; Hardware/software interfaces*

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Power Management, Selective Resume

1. INTRODUCTION

Modern laptops, tablets and handhelds continue to increase in complexity and energy use as they increase screen size, add discrete GPUs, multicore CPUs, and multiple radios such as cellular and WiFi. These increases in complexity require additional energy, but users also want longer battery run times, hence there is an increasing need to minimize energy consumption. Currently, laptop, netbooks and tabletPC's utilize ACPI sleep states to put idle systems to sleep and conserve energy when not in use. When a request is made to use the system (by the user or another device) the system is resumed from sleep to process the request.

Currently, systems do not know why resume requests were made, i.e., they are not aware of their resume context. As such, both the BIOS and the OS initialize all devices on the system at each resume, with the intent of returning the system to its pre-sleep state, regardless of the intended use. This monolithic approach is inflexible and causes long resume latencies. While some devices quickly enable the display to make systems appear more responsive, it can still take up to 30 seconds to fully resume from S3 sleep [2]. Furthermore, our findings indicate that when a system is repeatedly cycled between sleep states, the resume cycle times are significantly higher. A shorter cycle time is important as it means we can take advantage of low power states by putting the system to sleep for shorter periods that would not be practical with longer cycle times.

A system that needs to resume quickly cannot use sleep states when the resume latency is long. For example, consider a remote access scenario in which users, wanting their systems to remain responsive to network requests, configure their systems to resume upon receiving network traffic [5]. A large portion of this traffic does not require the system to resume [5], but to know that, the system must first wake up to analyze the traffic and then return to sleep. However, the average time between packets that trigger an inspection is shorter than current resume latencies, and so the system cannot evaluate the traffic itself and remains awake or fully cycles in and out of sleep modes.

This paper introduces *context-aware selective resume*, a technique that reduces latencies by resuming only the hardware that is specifically needed to complete a task. Selective resume enables applications to optimize resume operations for both power efficiency and latency. A power efficient resume seeks to minimize the power consumed during the transition process and by the subsequent tasks. A fast resume seeks to minimize the latency of the transition process and the completion of tasks. The approaches are related, since we may skip devices during a fast resume to both lower latency and save power consumption.

To gain a better understanding of the potential benefits of context-aware selective resume, we have conducted experiments to characterize the resume latencies and energy use associated with the ACPI sleep states. Our results show that the time is dominated by BIOS and OS processing time and not by hardware initialization time. This is a positive result because selectively initializing only the required set of de-

MCS'11, June 28, 2011, Bethesda, Maryland, USA.

vices can significantly reduce BIOS and OS processing time. We also measure *cycle times*, defined as the time it takes a computer to resume and then return to sleep. We compare non-selective to selective cycle times to demonstrate that a selective resume approach based on some OS modifications can reduce cycle times by as much as 87%, with further gains possible through BIOS changes. We also show that the potential energy savings of selective resume can approach 50% by examining the energy use of system components. Finally, we discuss the design considerations for building a context-aware selective resume system.

2. SELECTIVE RESUME

Selective resume provides benefits to applications that want access to only a part of a system, such as just CPU and memory, quickly and efficiently. Below, we briefly describe a normal and selective resume operation, and then present various scenarios that would benefit from selective resume.

2.1 Resume Operations

The normal resume process begins when a system receives a power management event (PME) triggered by a device, such as a Wireless Wake-On-LAN (WoWLAN) packet from a network device or an interrupt from a CMOS timer. In the normal resume process, the hardware initialization powers on the devices, then the BIOS code performs low-level initialization of the CPU, chipsets and other devices on the system, and then returns control to the OS by jumping to a pre-specified resume vector, which iterates through devices to reinitialize them to their pre-sleep state, and then starts threads that were previously suspended. The BIOS and OS follow a pre-determined order of events during every resume.

In contrast, with selective resume, after the hardware initialization, the BIOS would initialize only the CPU, then determine its resume context by reading the source of the power management event. For example, if the wake-up was caused by a WoWLAN packet, it reads this packet from the Wake-Up Packet Memory (WUPM) of the NIC. Based on the resume context, which specifies the devices needed, the system would then selectively initialize devices and then jump to one of several selective resume vectors. For example, a selective resume vector could be a memory address that transfers control back to the OS which then continues the selective resume process, or a special in-memory user-code segment designed to operate without the OS, or a firmware code segment.

2.2 Selective Resume Scenarios

The ability to provide context to a sleeping system upon resume allows fine-grained control over device use, enabling the various scenarios described below.

Computational Offloading: In cloudlet scenarios, as described by [6], mobile systems take advantage of low-latency, high-bandwidth connections to a nearby cloud environment running Virtual Machines to offload computationally complex tasks. In this approach, a mobile device sends a VM overlay (the state difference between a base VM and the instance on the device) to a cloudlet server that merges it with the base image, performs the computation, and returns the residual VM difference back to the device when complete. A selective resume approach would enable the size of a VM overlay to be reduced as certain components, for example data files, could be left on the device but would still be accessible using a pull-based approach if needed. This smaller VM overlay means only the minimal set of data is transferred to a cloudlet, saving both transmission energy usage and device energy usage, as the device can be put to sleep sooner. As an example after transferring only a small VM overlay, a device would enter a sleep state. If more data from the device is needed it would receive a WoWLAN packet to wake the device that contained context specifying a request for a specific data file. The device would initialize only the CPU, memory and Flash RAM , but not the GPU, display or cellular radio, fetch the required data from the device and send it to the cloudlet for continued processing. Additionally, the ability to rapidly resume the device means more opportunities for putting it to sleep can be exploited since we can take advantage of using the cloudlet for shorter tasks that would not be worth offloading with longer resume times.

Power Efficient Data Transfers: A selective resume system would also enable energy savings and quick response times in scenarios that wish to transfer data to or from a sleeping machine. For example in CatNap [4], the authors present a system that exploits the bandwidth differential between LAN and WAN links. Transfer data is buffered at a proxy device connected to the system by a fast LAN link (e.g. a 100Mb Ethernet). The system is put to sleep while the data is sent over a slow WAN link (e.g. a 1Mb ADSL), or the data received from the WAN is buffered on the proxy device, and the system is only woken up once the proxy cache is full. With selective resume, a WoWLAN packet would be sent with the data transfer request size in bytes, and the storage device (memory or disk) needed. The firmware would resume the CPU to examine the resume context, initialize the requested storage device and the NIC, while skipping any input devices, optical drives, and graphics cards. It would then transfer control to the selective resume vector, which would send or receive data, and then put the system back to sleep. For a receive operation, this could occur in the firmware using a pre-determined memory buffer, resuming the OS to process it only once the buffer was full. This would maximize the sleep time and reduce the power consumption while awake, lowering the overall energy usage of the system.

Partial Laptop Migration: A partial laptop VM migration scheme [3] would also benefit from selective resume. In this approach, a minimal working set of memory from an idle virtualized laptop is migrated to a consolidation server to enable it to remain active while the host machine is put to sleep. However, if an activity occurs on the migrated VM that triggers a page fault, the system must wake up the original system and fetch a page of memory. This incurs a significant delay as the system completes a full resume, enabling all devices and loading the OS, only to fetch a 4KB page. With selective resume, a WoWLAN packet with a memory page address would be sent, and it would selectively resume only the CPU, memory, and NIC, skipping the disk, graphics cards, and other devices, and transfer control to the selective resume vector. This operation would not require resuming the entire OS as the firmware would have direct access to memory, be able to read the page, and return the page by constructing and sending the packet directly via the NIC. This approach would allow the remote VM to quickly fetch the page and respond to an incoming packet within potential socket or application time-outs.

Component	Component System	Integrated System
CPU	Dual-core 2.4GHz Intel, 1066Mhz FSB, 4MB L2	Dual-core 2.6GHz Intel, 800Mhz FSB, 2MB L2
Memory	4GB	2GB
Hard Drive	74G SATA	74G SATA
Graphics	Intel X4500, integrated	Intel X4500, integrated
Sound	Realtek, integrated	Realtek, integrated
NIC	Realtek, integrated	Realtek, integrated
Optical	<none>	<none>
Power	650 Watt PSU	200 Watt PSU

Table 1: Experimental Systems

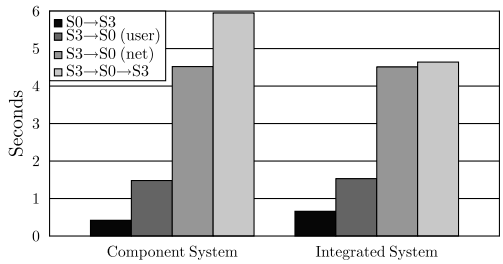

Figure 1: Suspend & Resume Times (w/o Hardware)

3. PROFILING SLEEP TRANSITIONS

Since power consumption, BIOS and hardware latency times are hardware specific, we conducted experiments on two different systems, an integrated system, representing an office PC, and a component system, representing a higher-end machine, as shown in Table 1. We used the same hard drive in both systems, configured with Debian Linux and the 2.6.18.8 kernel. Our experiments utilize the ACPI S3 sleep state, since it uses less energy than the ACPI S1 state, but doesn't incur a disk read like the ACPI S4 state. Below, we describe a series of experiments we conducted to better understand a normal resume operation, power use on our component system, and the impact of our selective resume approach.

3.1 System Transition Times

We conducted experiments on our test systems and investigated the system log files to measure the transition times from the power on state (S0) to the suspend-to-RAM (S3) sleep state, as shown in Figure 1. The S0→S3 sleep transition shows the OS time from issuing a pm-suspend command from the shell until the final call to suspend the hardware. The BIOS is not involved in the S0→S3 transition. The S3→S0 (user) resume transition shows both the BIOS and OS time until control was returned to the shell. The S3→S0 (net) resume transition shows both the BIOS and OS time until a connection was established to a remote server. In all these cases, the systems were given at least 30 seconds before another transition was initiated to stabilize. The S3→S0→S3 case shows a complete cycle, consisting of resuming the system, then suspending it, but without any time to stabilize between transitions. These results do not include hardware latencies, as those are not shown in the log files (results with hardware latencies are shown later). We conducted multiple runs and the average values shown have a standard deviation of less than 0.01.

Our findings show the resume operations typically take considerable longer than the suspend operation, 1.48s vs. 0.42s, for the component system, and 1.53s vs. 0.66s, for the integrated system. More interesting was the impact of calling a suspend directly after a resume. The cycle times were 5.95s, for the component system, and 4.64s, for the integrated system, which are significantly larger than the sum of the S3→S0 and S0→S3 times. This large increase was a result of the systems having to complete the background device resume operations, prior to beginning the subsequent suspend operation.

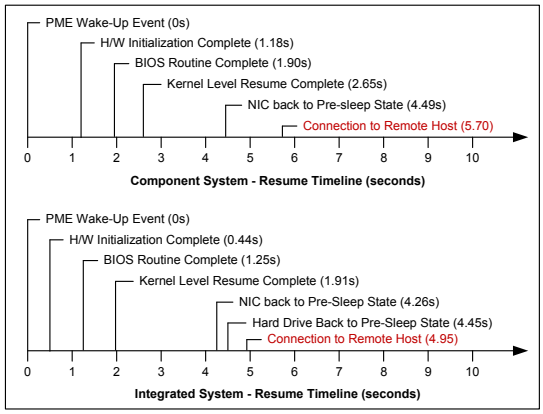

Figure 2: System Resume Timelines

3.2 System Resume Profiling

Our next experiment is designed to estimate the potential benefits of selective resume, by breaking down the time spent in different components of the system during a resume operation. The test systems were connected to a server on the same subnet. The server issued a WOL packet to initiate a resume, and we measured how long it took before a response was received from the test systems by the server. We instrumented the kernel to log key events, then conducted multiple runs and used average times based on the timestamps found in the system log (/var/log/syslog). Hardware initialization time was inferred by subtracting the BIOS and OS time accounted for in the system logs from the total time taken to open a connection to the remote server (measured on the server).

As shown in Figure 2, the activities and time spent initializing the hardware and executing the BIOS code is specific to each system, but the majority of the resume time is spent in software, either the BIOS or the OS. Comparing the two systems, the time spent in the OS is roughly equal at 3.7s and 3.8s respectively. We can also see how quickly the hardware responds, indicating that sub-second response times are achievable. For example, our integrated system completes its hardware initialization in 0.44s, so a selective resume that does not waste any time initializing unused devices (e.g., with a BIOS implementation) could complete a memory page fetch or packet inspection operation without running the full BIOS code or starting the OS and return to sleep in under one second.

Our resume times are comparable to previous work, although our suspend times are lower [1, 2]. However, it is hard to compare these times because the hardware is different and their methodology for collecting timing was not fully specified. We plan to run these experiments on other hardware to get a more comprehensive set of results.

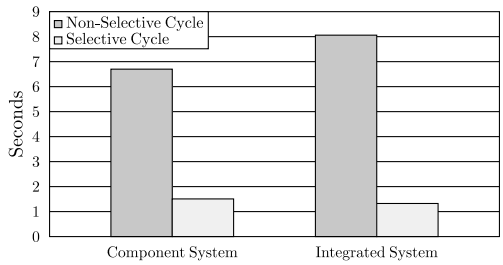

Figure 3: System Cycle Times (w/ Hardware)

3.3 Cycle Time

Our next experiment demonstrates the potential cycle time improvement possible with selective resume. Cycle time improvement would benefit certain applications such as the remote access scenario, described in Section **??**. In this scenario, if we find that the incoming packet is not worth waking the system, then we put the system to sleep as quickly as possible without initializing additional devices in the BIOS or the OS.

This experiment uses a modified version of the Linux kernel. After the first suspend, the subsequent resumes and suspends take a highly optimized kernel path designed to resume and suspend the system as quickly as possible. To measure the total cycle times, including hardware initialization times accurately, we used a digital oscilloscope and current probe connected to the common ground of the mainboard power supply. We conducted multiple runs and used average times based on the oscilloscope readings. In these experiments the unmodified BIOS transfers control back to the kernel via the normal ACPI resume vector. We do not return control to the user space, and instead put the system to sleep as quickly as possible in the kernel.

Figure 3 shows the complete cycle times, including hardware suspend and resume latencies. Compared to the results in Figure 1that show only BIOS and OS time, we see that the cycle time increases from 5.95s to 6.7s (0.75s) for the component system, which is reasonable based on observed hardware initialization times, but for the integrated system the increase from 4.64s to 8.06s (3.42s) is higher than expected. Examining the oscilloscope readings, we found that for roughly 3s of the 3.42s, the hardware is not powered up. We believe that the hardware is in some transient shutdown state and does not respond to WoL packets during the 3s.

Our results also demonstrate that the opportunity for reducing cycle times is high, with our streamlined kernel being able to reduce the overall cycle time from 6.7s to 1.51s for the component system, a 77% reduction, and from 8.06s to 1.33s for our integrated system, an 83% reduction. Based on the oscilloscope readings, the 3s delay described above does not occur with selective resume, but unfortunately, we cannot pinpoint the reason for this difference, without more control over the BIOS. We believe that BIOS programming will also allow cycle times to be reduced much closer to the hardware times.

3.4 Device Energy Usage

To demonstrate the potential energy savings that can be achieved through selective resume, we profiled the power usage of each device in our component system. We did not have access to the BIOS source to selectively enable devices upon resume, so device power was measured manually by removing each device and measuring the change in energy draw using a Watts-Up power meter. We measured and averaged power levels over 2 minute intervals with the system idling at the boot loader prompt. The graphics card used the largest amount of energy, consuming 84.4 of the 190 watts used by the system, followed by the hard drive at 9.5 watts, sound card at 3.1 watts, optical drive at 2.5 watts and NIC at 0.7 watts. This shows that a selective resume that does not enable these devices could reduce system energy consumption by over 50%.

4. SELECTIVE RESUME DESIGN

Having demonstrated the potential of our selective resume approach, we consider the design requirements for building a context-aware selective resume system.

4.1 Specifying the Resume Context

A fundamental aspect of the selective resume approach is that a system must have knowledge of the context under which it is being resumed. To specify the resume context, the system will need to know, at a minimum, the list of devices we wish to resume and where to pass execution control after devices are initialization. The resume context can be provided to the sleeping machine either prior to sleeping or upon the resume. In the former case, systems are provided with resume context data prior to suspending (e.g., through a user command or system policy). In the latter case, systems are provided with resume context data as part of the resume operation (e.g., embedded in a WOL packet). A system could also be given two resume contexts, one prior to sleeping and another on resume, and could be designed to handle both contexts by enabling the union of the two contexts.

4.2 Initiating the Resume

System resumes occur whenever a device generates a Power Management Event (PME) and these can be triggered either remotely or locally. Remote initiation is typically done through inbound communication devices such as a NIC receiving a Wake-On-Lan (WOL) packet or a modem configured to Wake-On-Ring. WOL packets are particularly useful as we can embed our resume context in the WOL the packet, which is then held in the NIC's Wake-Up Packet Memory (WUPM). The WUPM typically holds the first 128 bytes of the incoming packet, including the packet header, giving us up to 108 bytes to store the resume context, and is accessible by either the BIOS or the OS. Local initiations can be done through either the CMOS timer event or an attached device, such as a keyboard. While it may be possible to provide some limited context on local initiations, the resume context will likely have to be specified prior to sleep. To address security concerns, selective resume requests will need to be authenticated, and the selective resume vectors must be chosen from a well-known set of vectors.

4.3 Modifying Firmware/BIOS

Modifications to the OS and firmware will be required to enable fine-grained control of device operations to realize time and energy savings. We must determine the timing and order of device initializations, taking into account hardware dependencies, such as initializing the SATA bus prior to ac-

cessing the hard drive. System firmware code has historically been closed source, but recently open-source alternatives such as the Coreboot BIOS and Intel's TianoCore UEFI will make it easier and more practical to modify firmware.

4.4 Firmware / OS Interaction

Selective resume operations require the firmware and OS to work together and ensure that the required devices are available and no other devices are activated. With selective resume, the firmware will need to determine if it should call the OS resume vector at all, as there may be cases where the context specifies that all the processing happens without the OS being required. The firmware will also need to pass context information to the OS to inform the OS of which devices it has enabled, so that the OS knows which devices are available for its use. Otherwise, the OS may attempt to use uninitialized devices or mistakenly believe that the device have been removed from the system.

5. RELATED RESEARCH

While ACPI sleep states have been available on systems for many years, prior research to take advantage of them has focused primarily on avoiding unnecessary resumes. These approaches have, in part, been driven by the lack of flexibility and high latency of resume transitions, and they all require system augmentation with external servers or embedded systems. In contrast, our research seeks to remove the underlying problem by making sleep states more flexible and responsive.

Many systems have been designed to provide the illusion of on-line availability, by building a networking proxy that can filter and/or respond to incoming network requests [2, 5]. These approaches all require an external proxy server to check incoming requests and determine when the resume a system. By contrast, our approach would increase sleep times and provide the illusion of being online without requiring additional hardware.

Other research has focused on taking advantage of sleep states by offloading processing. For example, virtualized desktops can be offloaded to a separate server [3]. The GumStix device augments the system NIC for machine offloading [1]. Our work complements these works by providing fast fine-grained access to resources as they are needed.

6. CONCLUSIONS

We have made the case that the current monolithic resume functionality does not have the flexibility to meet the needs of modern and emerging mobile applications. We have shown that resume latencies are mostly dominated by software and can be reduced significantly. We have outlined specific improvements that can be made at the firmware or OS level to improve both power efficiency and response times. We envision that context-aware selective resume functionality will enable new uses for putting systems to sleep for shorter durations. If we can reduce the cycle times below an application's or user's ability to perceive them then we have effectively created an always-on system that consumes almost no power.

7. REFERENCES

[1] Agarwal, Y., Hodges, S., Chandra, R., Scott, J., Bahl, P., and Gupta, R. Somniloquy: augmenting network interfaces to reduce PC energy usage. In *Proceedings of the USENIX symposium on Networked systems design and implementation (NSDI)* (2009), pp. 365–380.

[2] Agarwal, Y., Savage, S., and Gupta, R. SleepServer: a software-only approach for reducing the energy consumption of PCs within enterprise environments. In *Proceedings of the 2010 USENIX annual technical conference* (2010).

[3] Bila, N., de Lara, E., Hiltunen, M., Joshi, K., Lagar-Cavilla, H. A., and Satyanarayanan, M. The case for energy-oriented partial desktop migration. In *Proceedings of the USENIX conference on Hot topics in cloud computing* (2010).

[4] Dogar, F. R., Steenkiste, P., and Papagiannaki, K. Catnap: exploiting high bandwidth wireless interfaces to save energy for mobile devices. In *Proceedings of the international conference on Mobile systems, applications, and services (MobiSys)* (2010), pp. 107–122.

[5] Nedevschi, S., Chandrashekar, J., Liu, J., Nordman, B., Ratnasamy, S., and Taft, N. Skilled in the art of being idle: reducing energy waste in networked systems. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation (NSDI)* (2009), pp. 381–394.

[6] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing 8* (October 2009), 14–23.