The Case for Energy-Oriented Partial Desktop Migration

Nilton Bila[†], Eyal de Lara[†], Matti Hiltunen[⋆], Kaustubh Joshi[⋆],
H. Andrés Lagar-Cavilla[⋆] and M. Satyanarayanan[‡]
[†]*University of Toronto,* [⋆]*AT&T Labs Research,* [‡]*Carnegie Mellon University*

Abstract

Office and home environments are increasingly crowded with personal computers. Even though these computers see little use in the course of the day, they often remain powered, even when idle. Leaving idle PCs running is not only wasteful, but with rising energy costs it is increasingly more expensive. We propose partial migration of idle desktop sessions into the cloud to achieve energy-proportional computing. Partial migration only propagates the small footprint of state that will be needed during idle period execution, and returns the session to the PC when it is no longer idle. We show that this approach can reduce energy usage of an idle desktop by up to 50% over an hour and by up to 69% overnight. We show that idle desktop sessions have small working sets, up to an order of magnitude smaller than their allocated memory, enabling significant consolidation ratios. We also show that partial VM migration can save medium to large size organizations tens to hundreds of thousands of dollars annually.

1 Introduction

Why are desktop PCs left running idle, if the hardware is typically capable of transitioning to power saving sleep states [13, 18]? Two reasons prevent the adoption of traditional power saving techniques. First, it is not uncommon for desktops to experience periods of brief idleness interspersed with active use. Second, desktops typically run a number of applications with always-on semantics, many of them requiring persistent network presence, such as VoIP, IM, e-mail, personal media sharing, etc [5, 8].

We propose to utilize *partial virtual machine migration* to solve this problem. By encapsulating the desktop session in a Virtual Machine (VM), the desktop VM can be temporarily staged in a consolidation backend or cloud during idle periods, allowing the actual physical desktop to be put to sleep. We note that cloud here can refer to a third-party infrastructure as a service (IaaS) cloud, an intranet backend, or even a federation of other computers in the same domain. Our approach will thus retain always-on semantics as the session itself never sleeps. When the idle period is over (e.g. user interacts with the system, or we detect significant background activity, such as system backup or virus check), the desktop hardware is woken up, and the VM is migrated back onto it. In this manner, our approach also leverages the user's desktop hardware and the capabilities of modern VM stacks to support hardware-based acceleration of graphics and multimedia, a fundamental challenge for thin client-based solutions to consolidation and power savings [11].

The key insight of our approach is that an idle desktop VM, even in spite of background activity, will need to access very little of its memory and disk state to function. If this working set is small enough, then a very large number of desktop VMs could be consolidated in a small set of cloud hosts. In addition, because only limited state needs to be transferred, migration can be performed almost instantaneously and at low energy cost, creating opportunities to save desktop power even for short periods of idleness.

To validate the feasibility of partial VM migration, we conducted an experiment that mimics a naïve implementation that fetches VM state on-demand. Our results show that, first, even with a naïve implementation it is possible to reduce desktop energy use by up to 50% over an hour of idleness, and up to 69% overnight; second, that idle desktop sessions have an order of magnitude smaller working sets than their allocated RAM; and third, that because small working set sizes enable high consolidation ratios, the energy efficiency of running idle VMs in the cloud can be high. Our conservative estimates show that small organizations can achieve savings in overnight energy use of at least 44%, and medium to large organizations can reach savings of more than 55%, amounting to tens to hundreds of thousands of dollars

annually.

In the following sections we describe partial VM migration for desktops, and motivate its usefulness as an energy saving technique. In Section 2, we discuss background and related work. In Section 3, we provide experimental results that show potential benefits of the approach. In Section 4, we identify the challenges that need to be addressed for the proposed approach to work well in practice and outline possible solutions, and finally, in Section 5, we conclude the discussion.

2 Background and Related Work

Modern computers ship with built-in mechanisms to reduce power usage of idle systems by entering low power, sleep states [1] . However, recent studies have found that users are reluctant to put their idle systems in low power states. Nedevschi et al. [13] find that desktop systems remain powered but idle for an average of 12 hours daily. Webber et al. [18] find that 60% of corporate desktops remain powered overnight.

Why are users reluctant to put their systems to sleep? People refuse to put their system to sleep either for the off-chance they may require remote access, run background applications (IM, e-mail), among other uses [5], or because many idle periods are short, often interspersed with active periods [8].

While there have been approaches to support remote access [2, 3, 15, 4], they do not support always-on application semantics.

Support for always-on applications have been proposed [6, 5, 13, 9, 16] either through remote proxies or specialized hardware. These approaches require developers to re-engineer most applications to support bimodal operation, transferring control and state between the full-fledged application and its proxied instance.

Alternatively, thin clients [14] allow users to run low power clients, which waste little power when idle. However, thin clients remain unpopular due to poor interactive performance, lacking crispness in response and local hardware acceleration. Also, while thin clients reduce energy usage of the client, they do little to improve energy efficiency of the servers, as they run sessions with full state and must support the peak loads of those sessions.

LiteGreen [8] takes a similar approach to ours. It migrates idle desktop VMs to a server and returns the VMs to the dedicated desktops when no longer idle. In contrast to our approach, LiteGreen migrates the VM's entire memory state to the consolidation server and expects the disk state to be readily available in network storage. Instead, partial VM migration migrates only the working set of the idle VM. Because this working set is small, as shown in Section 3.4.1, partial VM migration is able to migrate VMs almost instantaneously and at low energy cost, even across the wide area. This creates opportunities to save desktop power even in short periods of idleness. Partial VM migration can also achieve high consolidation ratios, and offers flexibility in the choice of consolidation platforms and deployment costs, allowing consolidation to occur on the desktops themselves or even on an IaaS cloud.

3 Feasibility Study

To evaluate the feasibility of partial virtual machine migration, we conducted an experiment that mimics a naïve implementation of the approach that fetches state exclusively on-demand. State is migrated on-demand so we can gain insights on what memory and disk pages need to be migrated and when. We use this experimental tool to answer the following fundamental questions:

1. While the VM runs on the cloud, can its desktop host save energy by sleeping?

2. Does the entire domain save energy by migrating idle sessions from desktops to the cloud?

These questions identify the minimal requirements for this approach to work. The former, tells us whether the sleeping intervals are long enough to offset energy costs associated with power state transitions. The latter, takes a holistic view of the architecture, including both, the desktops and the cloud, and compares their energy efficiency.

We answer these questions with a characterization of real, idle desktop workloads. To answer Question 1 we study in detail the memory and disk *page migration* caused by consolidated VMs. Page migrations tell us when the desktop must be awake to service page requests. We use these migration intervals to estimate the energy usage of each workload under the naïve prototype and contrast that with the cost of idling the desktop.

To answer Question 2 we measure the *working set* of each session, defined as the sum (in KB) of pages accessed by the VM during the session, and estimate the energy costs of running both desktops and cloud nodes. The working set offers an estimate of the consolidation ratio, the number of VMs that can run concurrently on a single cloud node. Consolidation ratios allow us to compare the energy efficiency of running VMs on the cloud against the efficiency of running on dedicated desktops. In addition to deriving the consolidation ratios, the working set also offers an upper bound on the amount of state that must be migrated back to the VM's home system during reintegration.

3.1 Methodology

We prototyped the on-demand migration functionality of partial virtual machine migration by using the SnowFlock [12] VM fork abstraction. SnowFlock supports rapid cloning of VMs by allowing cloned VMs to begin execution with minimal state, and fetching additional state on-demand, as needed by the VM. However, SnowFlock does not support reintegration of VM state, nor does it support idle session migration policies. Although SnowFlock is an incomplete representation of a naïve partial virtual machine migration prototype, it allows us to measure important attributes of such system.

We prepared a VM image with the applications described in the workloads section (3.1.2) below, and ran the VM. We allowed the session to become stable, and after approximately five minutes used SnowFlock to clone the running image into a cloned copy. We ran the cloned copy for a full hour and recorded all disk and memory pages it retrieved from its master copy (including the initial state fetched to instantiate the copy). In a real system, we would put the desktop to sleep while the session runs on the cloned VM. Each experiment was repeated three times, unless otherwise noted.

3.1.1 Platform

SnowFlock is implemented on the Xen [7] VMM version 3.4.0. Both the host and the VM use Debian Linux 5.0 with x64 version of the kernel 2.6.18.8. The VM image was configured with 1GB of RAM and a 12GB disk.

We collected our traces on a Dell Optiplex 745 with 2.66GHz Intel Core 2 Duo CPU and 4GB of RAM. A similar Optiplex 745 system was used by Agarwal et al. [5], who, experimentally profiled the system and found it to draw 102.1W of power when idle and 1.2W when suspended in sleep state. Agarwal et al. also report measuring 10 seconds of suspend and 5 seconds of resume times for the system.

Workload	Description
Login	The login screen (GDM) of a Linux desktop system.
E-mail	Mozilla thunderbird connected to an IMAP e-mail server. The client polls the server every 10 minutes.
IM	The Pidgin multi-protocol IM client connected to an IRC room with more than 100 users.
Multitask	A Gnome Desktop session with the E-mail client, IM client, Spreadsheet (OpenOffice Calc), PDF Reader (Evince) and file browser (Nautilus)

Table 1: Idle session workloads.

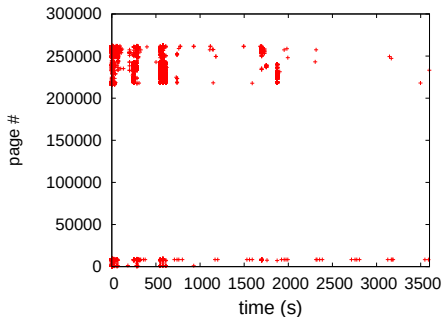


Figure 1: Memory page migration for E-mail workload.

3.1.2 Workloads

Table 1 describes the workloads we studied. The workloads consist of typical desktop applications. Login illustrates the nightly behaviour of desktop systems whose users log out at the end of the work day. E-mail and IM are minimal workloads, consisting of an X session, xterm and the the subject application. These micro workloads are intended to give us a detailed look at the behaviour of applications that maintain presence with external hosts. Multitask illustrates the behaviour of the desktop of a multitasking office worker.

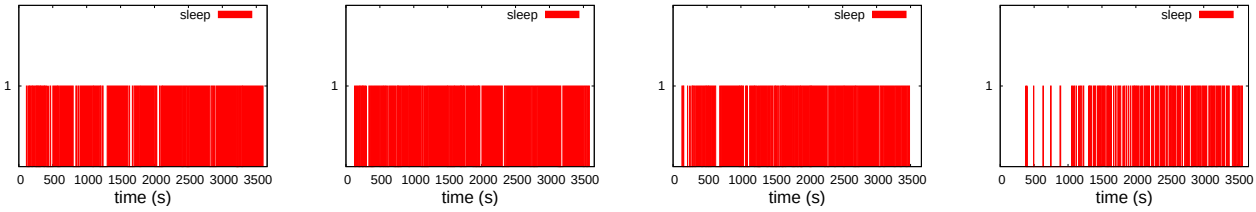
3.2 Memory and Disk Access Patterns

Figure 1 shows the memory page migration pattern for the E-mail workload. First, the figure shows a degree of locality in page migrations. Pages tend to migrate in clusters that include, on the one hand, application code and data, and kernel data, and on the other kernel code, represented in the figure by the high and low address spaces, respectively. The other workloads studied show similar clustering patterns. We exclude those figures for brevity. As we argue in Section 4, this locality of page accesses can be exploited for pre-fetching strategies that improve energy efficiency.

3.2.1 Desktop sleep opportunities

We now investigate in detail the frequency and duration of potential sleep intervals, exploitable by a naïve implementation of partial VM migration, that migrates pages on-demand. Potential sleep intervals are all periods in which the desktop is not serving disk or memory pages to the remote VM. This definition assumes instantaneous state transitions, an assumption we relax in Section 3.3.

Figure 2 shows sleep opportunities available to the desktop over an hour-long run of the workloads in the cloud. The figure shows that, for most workloads, the desktop can spend a large period of time sleeping. For the micro-workloads, it can sleep for 42 to 46 minutes, and for the multitask workload it can sleep for a total of 17 minutes. Each sleep interval lasts, on average, 50 to



(a) Login (b) E-mail (c) IM (d) Multitask

Figure 2: Potential desktop sleep intervals.

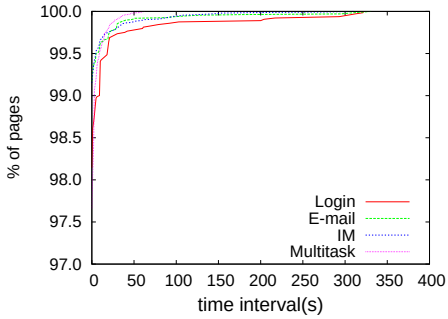


Figure 3: Page request interarrivals.

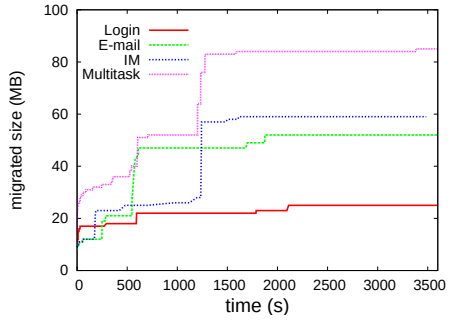


Figure 4: Cumulative size of migrated pages.

73 seconds for the micro-workloads, and 13 seconds for the multitask workload. While these workloads migrate between 6,413 to 21,777 pages, Figure 2 shows that the desktop only wakes up to service pages between 37 and 81 times. This indicates that the desktop is able to batch multiple page migration requests in each wake up.

Figure 3 shows the page request interarrivals. The figure confirms that most page requests arrive on the desktop simultaneously (more than 98% of pages across workloads). These results show significant potential for energy savings for the desktop through sleep.

Figure 2 also shows that page requests, while frequent at the start of the session, their frequency decreases over time, resulting in longer sleep intervals. Indeed, Figure 4 further shows that the VM's demand for pages flattens in the first 25 minutes across all workloads.

3.3 Desktop Energy Savings

We now estimate potential energy savings by the idle desktop system. Given the distribution of page migrations, we developed *naïve wait*, a strategy that demonstrates potential energy savings over the above sleep intervals. This strategy relaxes the assumption of instantaneous sleep and wake ups, and takes into account the energy costs associated with state transitions.

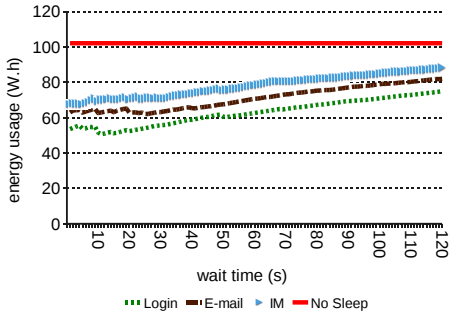


Figure 5: Effect of t_w on energy usage.

Naïve wait has no advance knowledge of incoming page requests. Rather, it decides to put the system to sleep only if it has not received a page request after a pre-determined waiting period (t_w). Whenever a page migration request arrives, it wakes the system up to service the request.

3.3.1 Energy savings during hour-long idle intervals

The potential for energy savings for the naïve wait approach depends on the choice of parameter t_w. With shorter t_w, naïve wait aggressively puts the system to sleep, which may extend sleep intervals, but can also mean more frequent state transitions, which are power intensive. We vary t_w from one second to two minutes, in increments of one second, aggregate the times spent in each state as well as in transition, and use these, together with the power profile of the system, to compute the total energy used. As reported earlier, sleep and idle power draw of the system are 1.2W and 102.1W, respectively. For the transition state, we make the conservative assumption that the system draws its peak power. We use the capacity of the system's DC power supply of 280W as a reasonable estimate of peak power.

Figure 5 shows the energy usage for all hour-long micro-workloads as we vary the parameter t_w of the naïve scheduler. We contrast these usage profiles with the energy of the base case, in which the desktop always

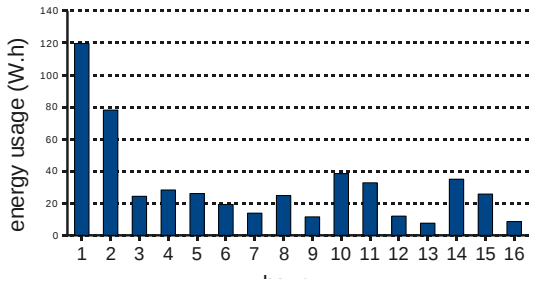


Figure 6: Hourly energy usage for multitask workload.

runs the idle workloads locally. The workloads demonstrate larger energy gains with short wait periods. The minimal energy points for all three micro-workloads occur between $t_w = 10s$ and $t_w = 11s$. With $t_w = 10s$, the Login and E-mail workloads reach their lowest energy usage of 51.5W.h and 62.6W.h, respectively. With $t_w = 11s$, IM reaches it's lowest of 69.1W.h. These results demonstrate potential energy savings of 32% to 50% for the micro-workloads.

3.3.2 Overnight energy savings

While naïve wait can reduce energy usage of the micro-workloads in the course of an hour-long session, we find it unable to produce similar savings for more demanding workloads, such as multitask, in the same short period. This is a challenge that motivates better designs than the naïve example. We defer this discussion to Section 4.

Naïve wait is, however, still capable of reducing energy usage greatly in overnight sessions, even for more demanding workloads. Figure 6 shows hourly energy usage for a 16-hour overnight run of multitask workload on a system that implements naïve wait. The energy usage is computed with $t_w = 10s$. The figure shows that, while in the first hour the system is unable to save any energy, in subsequent hours it registers significant energy savings. In the second hour, energy usage drops to 78.1W.h., leading to savings of 23% over the base, no-sleep case. In subsequent hours the system consistently saves at least 62% of energy hourly. Overall, in the course of 16 hours, naïve wait saves the desktop system 69% of energy, while running the multitask workload.

3.4 Domain-wide Energy Savings

We now evaluate the potential energy savings an organization can achieve by migrating idle desktop sessions to the cloud. These estimates include both the energy costs of running the desktops and the cloud systems. We first show that idle desktop VMs can be more energy efficiently ran on the cloud than on dedicated desktops, and then show that partial VM migration can save organizations conservatively thousands of dollars annually.

3.4.1 Cloud Energy Efficiency

Workload	Working Set (KB)	Memory Migration (KB)	Disk Migration (KB)
Login	43,012(3,851)	23,511(1,971)	2,217(3,827)
E-mail	71,376(424)	53,580(379)	25(8)
IM	80,351(1,147)	57,031(1,158)	3,176(38)
Multitask-1h	119,896(10,802)	93,195(9,744)	248(327)
Multitask-16h	148,224	110,436	7524

Table 2: State footprints of workloads. Standard deviations are given in parenthesis.

To determine whether, on aggregate, the systems in a domain, including desktops and cloud nodes, attain net energy savings, we look at how effectively partial virtual machine migration consolidates multiple idle VMs.

Table 2 shows the working set of the VM under each workload, as well as the total page migrations broken down into memory and disk pages. Memory migration size differs from working set size because, the VM is capable of allocating memory locally on the cloud (for example, when applications request page allocations to be immediately overwritten). As the table shows, the working set across workloads is small, under 85 MB for the micro-workloads and under 145MB for the multitask workload, even when ran for 16 hours. Recall that the VM is configured with 1GB of RAM and 12GB of disk. Looking at the memory requirements alone gives us consolidation ratios of at least seven (in the worst, 16 hour of multitask case). From previous experience, we expect the consolidation ratios to be even higher on systems configured with large RAM. We find that the working set size does not grow proportionally with the allocated VM memory. In fact, we experimented with the above workloads under a VM with 512MB of RAM, and found the working set and migration sizes to be very similar to those of the 1GB configuration. We therefore expect partial virtual machine migration to scale well with the size of available RAM. And, because increased RAM in the cloud nodes does not increase the energy footprint proportionately, larger RAM will lead to higher energy efficiency, as long as other system components are not the bottleneck.

3.4.2 Annual Energy Savings

We now estimate the annual energy savings an organization can obtain by employing the naïve implementation of partial virtual machine migration to migrate desktops during overnight hours. These numbers do not include additional savings obtained by migrating the desktop throughout the day as a function of idleness.

The annual overnight energy cost for an organization without partial virtual machine migration is given by:

$$\$NoSleep = N \times I \times E \times 365$$

where:

N is the number of desktops in the organization

I is the total energy consumed by an idle desktop overnight ($102.1W \times 16h$ for the Dell Optiplex)

E is the dollar cost of electricity. The U.S. Department of Energy estimates the average residential cost of electricity to be 9.4 cents per kWh [17].

The annual overnight energy cost for an organization with partial virtual machine migration is given by the following formula:

$$\$PartialMigration = [C \times B + (N - C) \times S] \times E \times 365$$

where:

$C = \lceil \frac{N}{V} \rceil$ is the number of systems needed to form the cloud

B is the total energy consumed by a busy system. We make the conservative assumption that the cloud nodes are constantly busy, and draw peak power of 280W.

S is the total energy consumed by a sleeping desktop overnight (505.7W for the 16 hour multitask workload, as shown in Section 3.3.2). It includes the cost of waking up to service page requests.

V is the number of VMs per cloud host. On the Dell Optiplex with 4GB of RAM, we were able to repeatedly run 23 VMs, each configured with 145MB of RAM, sufficient to accomodate the working set of the 16 hour multitask workload measured in Section 3.4.1. In addition to VM memory allocation, Xen allocates approximately 64MB to itself (hypervisor), and 416MB to the administrative system (dom0).

# of Desktops	Energy Cost ($)	
	No Sleep	Partial Migration
10	560.49	309.86
50	2,802.44	1,276.61
100	5,604.88	2,416.86
1,000	56,048.82	23,350.43
10,000	560,488.16	232,822.49

Table 3: Annual overnight energy costs for organizations of various sizes.

Table 3 shows potential energy savings for organizations of various sizes in which partial virtual machine migration is used to reduce energy usage during the overnight hours. Small organizations can benefit from at least 44% in energy savings, while medium to large organizations benefit from more than 55%. For a large organization with 10,000 desktops, this translates to $327,665 saved annually by simply allowing idle desktops to partially migrate during the overnight hours.

So far, the discussion has made an implicit assumption of homogeneity of hosts on the cloud as the dedicated desktops. This is done only for simplicity of the discussion. The cloud may very well run powerful servers (or even low power devices), so long as the per session energy efficiency of each node individually is better than that of the dedicated desktop.

4 Challenges of partial virtual machine migration

While the previous results indicate that partial migration of desktops can result in high consolidation ratios and make substantial energy savings possible, several challenges still remain.

Desktop power cycling is necessary for achieving the energy savings we desire, and is an integral part of our approach. This raises dual concerns. First, frequent power cycles may lead to reduced life expectancy of computer components, especially mechanical hard disks that are rated for a limited number of lifetime spin up and down cycles. Second, power cycles also lead to higher transient energy use compared to idle power consumption, and can reduce the savings realized by partial desktop consolidation. A fundamental challenge for our approach then is to reduce the number of sleep cycles and increase sleep duration, and we propose to do so using a variety of methods.

Memory pre-fetching can potentially result in significant reductions in demand-paging requests from the cloud and allow the desktop to sleep longer. The memory access patterns shown in Figure 1 show excellent locality and indicate that contiguous page pre-fetching may help over short periods of time. An outstanding challenge is to develop techniques to allow the host VMM to access per-application working sets. Over longer periods of time, current working sets may not be sufficient as scheduled background activities change the memory access patterns, and additional techniques will be needed to predict prefetch sets. One possibility is to use VM time-travel [10] techniques to estimate future access patterns.

Since we strive to maintain a seamless user experience in spite of sleeping through even short periods of inactivity, fast reintegration of a VM from the cloud back to its desktop is a crucial requirement. Different pre-fetching strategies than those used for migration are likely to be useful during reintegration. For example, while quickly transferring read-only code pages is essential during migration, transferring dirty pages especially from frequently used structures such as the stack is more important for reintegration. In addition, periodic resynchronization of dirty pages whenever the desktop is woken up for demand fetching can provide a cheap way of

keeping the state transfer requirements during reintegration limited.

The proposed approach supports migration to local backends as well as to third party IaaS clouds. In the latter case, because the desktop VM is migrated to a different IP subnet, it requires the use of traffic redirection techniques such as MobileIP or VPN tunnels.

Finally, several policy questions will need to be addressed to ensure that both energy efficiency and user experience are optimized. These include heuristics to decide when to partially migrate a host to the cloud, when to migrate on-demand missing pages from the desktop as opposed to reintegrating the VM back to the desktop, how long to keep the desktop awake after it has been woken up to serve pages, and how long to retain a reintegrated VM image in the cloud in case it is migrated again. While the answers are obvious in some scenarios, e.g., local user activity should trigger reintegration, other situations require energy and capacity tradeoffs to be considered.

5 Conclusion

We proposed the use of partial virtual machine migration to improve energy efficiency of idle desktop computers. Partial VM migration works by encapsulating the desktop session in a virtual machine and migrating only a small footprint of state of the idle VM needed to continue execution remotely, while putting the desktop to sleep. When the user returns to the desktop, or otherwise the session ceases to be idle, partial virtual machine migration reintegrates the changed state of the remotely running VM with the desktop and resumes local execution. We have shown that even a naïve implementation of this approach can reduce energy usage of an hour-long idle desktop session by up to 50%, and of an idle overnight session by up to 69%. We have shown that the working set of an idle desktop session is an order of magnitude smaller than its total allocated memory, enabling high consolidation ratios and energy savings across the domain. We have shown that, conservatively, partial VM migration can reduce annual energy costs in overnight hours for organizations with 100 or more desktops by at least 55%, amounting to $3,100 for an organization with 100 desktops to more than $300,000 for organizations with 10,000 desktops. We expect page pre-fetching for cloud execution to help reduce the frequency of power cycles in desktops, and improve energy savings further. And finally, we have identified remaining challenges that must be addressed for the proposed system to work well in practice, and outlined possible solutions.

References

[1] ACPI specification. `http://www.acpi.info/DOWNLOADS/ACPIspec40.pdf`.

[2] Wake on LAN technology. `http://www.liebsoft.com/pdfs/Wake_On_LAN.pdf`, Jun 2006.

[3] Wake on Wireless LAN (WoWLAN). `http://www.intel.com/support/wireless/wlan/sb/CS-029827.htm`, 2008.

[4] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless wakeups revisited: Energy management for VoIP over Wi-Fi smartphones. In *MobiSys '07*, Jun 2007.

[5] Y. Agarwal, S. Hodges, J. Scott, R. Chandra, P. Bahl, and R. Gupta. Somniloquy: Augmenting network interfaces to reduce PC energy usage. In *NSDI '09*, Apr 2009.

[6] Y. Agarwal, S. Savage, and R. Gupta. SleepServer: Energy savings for enterprise pcs by allowing them to sleep. Technical Report CS2009-0953, University of California, San Diego, Nov 2009.

[7] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03*, Oct 2003.

[8] T. Das, P. Padala, V. N. Padmanabhan, R. Ramjee, and K. G. Shin. LiteGreen: Saving energy in networked desktops using virtualization. In *2010 USENIX ATC*, Jun 2010.

[9] C. Gunaratne, K. Christensen, and B. Nordman. Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed. *IJNM*, 15(5):297–310, Sep 2005.

[10] D. Gupta, K. Yocum, M. McNett, A. C. Snoeren, A. Vahdat, and G. M. Voelker. To infinity and beyond: Time-warped network emulation. In *NSDI '06*, May 2006.

[11] H. A. Lagar-Cavilla, N. Tolia, E. de Lara, M. Satyanarayanan, and D. O'Hallaron. Interactive resource-intensive applications. In *Middleware '07*, Nov 2007.

[12] H. A. Lagar-Cavilla, J. A. Whitney, A. M. Scannell, P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan. SnowFlock: rapid virtual machine cloning for cloud computing. In *EuroSys '09*, Mar 2009.

[13] S. Nedevschi, J. Chandrashekar, J. Liu, B. Nordman, S. Ratnasamy, and N. Taf. Skilled in the art of being idle: Reducing energy waste in networked systems. In *NSDI '09*, Apr 2009.

[14] T. Richardson, Q. Stafford-Fraser, K. Wood, and A. Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1), Jan/Feb. 1998.

[15] E. Shih, P. Bahl, and M. J. Sinclair. Wake on Wireless: An event driven energy saving strategy for battery operated devices. In *MobiCom 2002*, Sep 2002.

[16] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins. Turducken: Hierarchical power management for mobile devices. In *MobiSys '05*, Jun 2005.

[17] U.S. Department of Energy. Energy savers: Tips on saving energy & money at home. `http://www1.eere.energy.gov/consumer/tips/appliances.html`, 2009.

[18] C. A. Webber, J. A. Robertson, M. C. McWhinney, R. E. Brown, M. J. Pinckard, and J. F. Busch. After-hours power status of office equipment in the USA. *Energy*, 31(14):2487–2502, Nov 2006.