Oasis: Energy Proportionality with Hybrid Server Consolidation

Junji Zhi
University of Toronto
zhij@cs.toronto.edu

Nilton Bila
IBM T.J. Watson Research Center
nilton@us.ibm.com

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

Abstract

Cloud data centers operate at very low utilization rates resulting in significant energy waste. Oasis is a new approach for energy-oriented cluster management that enables dense server consolidation. Oasis achieves high consolidation ratios by combining traditional full VM migration with partial VM migration. Partial VM migration is used to densely consolidate the working sets of idle VMs by migrating on-demand only the pages that are accessed by the idle VMs to a consolidation host. Full VM migration is used to dynamically adapt the placement of VMs so that hosts are free from active VMs. Oasis sizes the cluster and saves energy by placing hosts without active VMs into sleep mode. It uses a low-power memory server design to allow the sleeping hosts to continue to service memory requests. In a simulated VDI server farm, our prototype saves energy by up to 28% on weekdays and 43% on weekends with minimal impact on the user productivity.

1. Introduction

Electricity consumption by data centers is steadily increasing. In 2013, US data centers alone consumed 91 billion kilowatt-hour, or the equivalent of the annual output of 34 coal-fired power plants. Remarkably, this demand is anticipated to increase by over 50% by 2020 [1].

While virtualization technology was intended to increase resource utilization, the reality is that cloud data centers operate at very low utilization rates. For example, a recent study of Amazon's EC2 [16] reports average server utilization over a whole week of only 7.3%.

CPU power management technologies like Dynamic Voltage and Frequency Scaling (DVFS) have drastically reduced CPU energy consumption. However, other server components such as DRAM, motherboard and peripherals, have come to dominate overall energy usage during low utilization periods. As a result, idle servers consume 60% of their peak power [3].

Suspending idle VMs to disk and powering down under-utilized hosts is not preferred because it causes disruptions to applications. Cloud services such as Hadoop, Elasticsearch and Zookeeper require that members of a cluster send periodic heartbeat messages to maintain membership in the cluster. User applications such as VoIP and remote desktop access clients, and background processes such as data replication services, require their VMs to remain always on and network present despite their idle state.

VM migration is a more attractive solution since it causes minimal disruptions to applications. Migrating VMs from under-utilized physical hosts and then turning idle hosts off has been proposed to achieve energy-proportionality at the cluster level [25]. A simple approach used by previous works is live VM migration [5, 15, 22, 28]. Unfortunately, full VM migration requires the target host to have enough resource slack to accommodate the oncoming VMs, resulting in low consolidation ratios. Moreover, migrating an entire VM with gigabytes of memory state creates network congestion and incurs in long migration latencies.

Partial VM migration [4] has been used to save energy in desktop deployments by consolidating desktop VMs densely. Partial VM migration consolidates only the working set of idle VMs and lets VMs fetch their memory pages on-demand. The desktop transitions from low-power sleep mode to full-power mode, in order to service the page requests from its migrated partial VM, and returns to low-power. This approach does not work for hosts with co-located VMs for two reasons. First, as some VMs in the host become idle, others remain active and prevent the host from sleeping. Second, even when all the VMs in the host become idle and their working sets are consolidated, the frequency of aggregate on-demand page requests from the multiple VMs greatly limits the server sleeping opportunities.

This paper introduces Oasis, a new approach to energy-oriented cluster management that makes dense server consolidation possible. Oasis achieves high consolidation ratios by combining traditional full VM migration with partial VM

[1] http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IB.pdf

EuroSys '16 April 18–21, 2016, London, United Kingdom

ACM 978-1-4503-4240-7/16/04…$15.00
DOI: http://dx.doi.org/10.1145/2901318.2901333

migration. Partial VM migration is used for dense consolidation of idle VMs. Full VM migration is used to free servers from hosting active VMs that prevent sleep. Oasis augments the partial VM migration technique with a low-power memory server that enables its host to continue to service memory page requests while the host is in sleep mode.

We evaluated our prototype on a simulated cluster of virtual desktop servers (VDI) using usage traces collected from real desktop users. Our results show that Oasis reduces energy usage by up to 28% on weekdays and 43% on weekends with minimal impact on user experience.

This paper makes the following contributions: (i) it introduces a new energy-oriented VM consolidation approach that uses a hybrid approach that combines full and partial VM migration to achieve high consolidation density; (ii) it shows that this approach can save significant energy for a variety of workloads; and, (iii) it introduces a low-power memory server that can efficiently serve memory requests.

The remainder of this paper is organized as follows. § 2 provides an overview of live and partial VM migration. § 3 introduces hybrid server consolidation. § 4 describes the implementation of our prototype and presents results from micro benchmark experiments. § 5 presents results from our trace-driven simulation of cluster deployments of Oasis. Finally, § 6 and § 7 discuss related work and conclude the paper.

2. Background

VM migration has been employed for consolidation of idle VMs. Previous works [5, 15, 22, 24, 25, 28] have used either live migration of full VMs [6] or partial migration of VMs.

Live VM migration refers to migration of VMs with minimal downtime. Live migration is implemented with one of two approaches: pre-copy live migration and post-copy live migration. Pre-copy live migration iteratively copies pages from source to destination while the VM runs at the source. The first iteration copies all pages to the destination. In subsequent iterations only pages dirtied by the VM's execution during the previous iteration are copied. Once the set of dirty pages is small or the limit of iterations exceeded, the VM is suspended and all pages and execution context transferred to the destination. The VM's execution starts at the destination and its resources are released from the source. Post-copy live migration [11] starts by suspending the VM at the source and transferring its execution context to the destination host, where the VM resumes execution. Memory is actively pushed from the source while the VM executes on the destination. When the VM accesses pages that have not yet arrived at the destination, pages are faulted in from the source.

Both methods migrate the VMs in full, which requires the destination to have enough resource capacity and thus limits consolidation density. Full VM migration is also slow and restricts the cluster controller's ability to consolidate VMs over short idle intervals.

Partial VM migration consolidates only the VMs' idle working sets. It takes advantage of the observation that idle VMs access only a small fraction of their full memory allocation. For example, Figure 1 shows the aggregate memory accesses of three VMs that were allowed to become idle after an initial warm-up period. Two of the VMs are respectively configured as a Web server and a database server to run the popular RUBiS [2] benchmark, which emulates an online auction site. The third VM runs a remote desktop environment with Linux, a mix of multiple LibreOffice applications, and a Web browser with multiple open tabs. Each VM was configured with 4 GiB of memory and a 12 GiB disk image. Over the course of an idle period of 1 hour, the Web and database VMs accessed 37.6 MiB and 30.6 MiB out of the 4 GiB memory allocation, respectively. By comparison, the desktop VM accessed 188.2 MiB. This corresponds to less than 5% of their nominal memory allocation.

Partial VM migration operates as follows. When VMs are active they run on their home hosts where their full memory footprint resides in DRAM. When the VMs becomes idle, their idle mode working sets (pages that are accessed during the idle time) are migrated on-demand to consolidation hosts where the VMs then run. Migration to the consolidation host starts by suspending the VM at its home and transferring to the consolidation host only the execution context and VM meta-data needed to create and initiate execution of a partial VM. This VM lacks most of its memory and its execution causes it to access missing pages. Similar to post-copy migrations, these pages are faulted in from the home host. However, unlike the post-copy migration, partial VM migration does not actively push all VM pages to the destination. When the home server is idle, it is suspended into sleep mode. When the partial VM faults on a page, the home server is awakened to service the page request and kept awake only for the duration of received requests.

Partial VM migration was originally applied to saving energy in office environments by consolidating idle desktops. In this scenario, when the VM becomes active again (because the user started to interact with it), it is migrated back to its home host (i.e., the user's desktop). Migration away from the consolidation host is fast because only the pages modified during VM execution on the consolidation server are transferred. This approach ensures that active VMs can deliver full performance while idle VMs run their applications with minimal resources.

While this approach yields substantial sleep opportunities for a host that is home to a single VM, sleep opportunities disappear when multiple VMs are co-located on the same home. Figure 2 contrasts the sleep opportunities available to a host serving page requests for a single database VM and one serving requests for ten VMs, five consisting

[2] http://rubis.ow2.org/

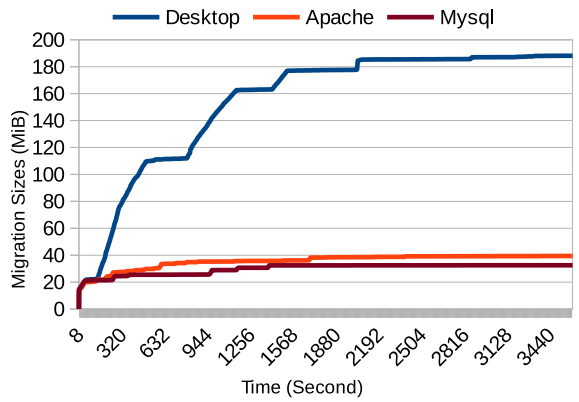

Figure 1. Memory access pattern for an idle desktop, a Web server and a database VM.

of databases and five consisting of Web servers. The average page request inter-arrival time goes from 3.9 minutes in the case of a single VM to 5.8 seconds in the case of 10 co-located VMs. This is nearly the same time it takes for a commercial server to transition between low- and full-power modes [3], effectively preventing the server from taking advantage of any sleep opportunities. This result shows that using servers that transition between low-power and full-power modes to service page requests will have limited opportunities to sleep and we will show in §§ 3.3 how low-power memory servers can support host sleep in these environments.

3. Hybrid Server Consolidation

Oasis is a new approach for energy-oriented cluster management and achieves high consolidation ratios. Our approach combines traditional full VM migration with partial VM migration. Each host is augmented with a low-power memory page server that can efficiently serve VM memory state while the host sleeps. Oasis is predicated on the following assumptions:

1. Consolidation ratio (the number of VMs per host) is limited by the memory demands of VMs as opposed to other resources, e.g., CPU. Modern hypervisors have better over-subscription support for CPU than memory. Over-committing CPU by a factor of 3 is regarded as a safe practice [17]. On the other hand, sophisticated memory sharing techniques, such as ballooning and deduplication, enable memory over-commitment by only a factor of 1.5 [2].
2. The virtual disks of VMs are network hosted.
3. An active VM requires all its memory state to be present on the host's memory in order to achieve good performance.

[3] Our prototype server takes 3.1s to suspend to RAM and 2.3s to resume.

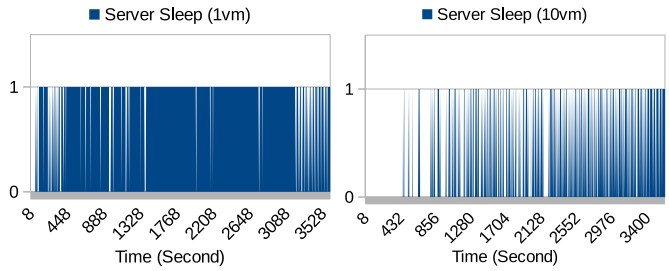

Figure 2. Server sleeping opportunities with 1VM vs. 10 VMs

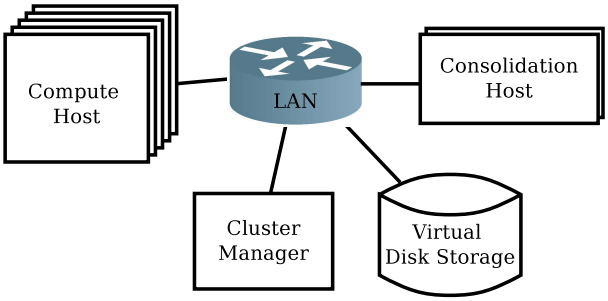

Figure 3. An Oasis cluster with compute and consolidation hosts

4. An idle VM requires only a small fraction of its memory state to be present on the host's memory (see § 2).
5. Servers include support for low-power sleep mode (e.g., ACPI S3).

The rest of this section describes the approach we use for VM consolidation, including placement decisions made when VMs change between active and idle state, and when consolidation hosts exhaust their memory capacity, and discuss the design of a low-power memory server that is necessary to support energy-oriented server consolidation.

3.1 Consolidation Approach

We consider a VM to be in one of two states: active or idle. A VM is active any time it needs to access a large fraction of the assigned system resources (e.g. memory, CPU) in order to process the ongoing workloads (e.g. an Elasticsearch cluster member processing large volumes of incoming queries). Conversely, A VM is idle if it accesses a small fraction of assigned system resources (e.g. elasticsearch cluster member sending and receiving periodic ping messages to maintain membership in the cluster). To determine a VM's idleness, we can monitor its resource usage. For example, one metric for memory usage is VM page dirtying rate which can be monitored from the hypervisor [9].

A cluster consists of compute hosts and consolidation hosts (Figure 3). A new VM is created in a compute host which becomes the VM's current home. At any time, each

host may run a mix of active and idle VMs. Compute hosts and consolidation hosts can be in one of several power modes:

- *powered* – the host is fully powered and running VMs;
- *low-power/sleep* – the host is using minimal power to maintain the context and cannot run VMs;
- *in-transit* – the host is transitioning between low-power and powered modes.

Next, we discuss the consolidation and power management policies of the cluster manager. The cluster manager determines when to migrate a VM, where to migrate the VM, how to migrate the VM, and when to place hosts in either powered or low-power modes. The cluster manager makes migration plans at periodic intervals. The size of an interval is a configurable parameter.

When to migrate: The cluster manager consolidates VMs only when it determines that doing so can save energy. Consolidation decisions are made periodically. At the beginning of each interval, the manager searches for an alternative VM placement plan that minimizes the number of powered hosts. If a better plan than the current one is found, the manager initiates VM migrations to realize the new plan.

How to migrate: The cluster manager aims to minimize the application performance degradation caused by VM migrations. Because partial VMs request memory pages from remote hosts, their applications can suffer from visibly degraded performance that is unacceptable for an active user (Figure 6). As such, partial VM migration is used only on idle VMs. Active VMs are migrated in full to consolidation hosts. We use pre-copy live migration [6] because it offers minimal performance degradation to active workloads during migration.

Where to migrate: Because VMs states and their resource demands vary over time, the search of an optimal VM placement is an NP-hard problem. In this paper, we take a simple greedy approach. First, we sort the compute hosts by their total VM memory demand (or by their migration costs) in ascending order and form a queue of hosts to vacate. We find a plan that vacates the maximum number of compute hosts from the queue.

The destination for each migrating VM is selected at random from the consolidation hosts list. Whether a host can become a destination of a VM depends on whether that host has enough memory capacity. More sophisticated placement algorithms that optimize specific goals, such as reducing memory fragmentation, is not the focus of this paper. Note that migration decisions could also be based on resources other than memory, e.g. network and storage.

When to sleep: For a compute host, if all its VMs are migrated out, it enters low-power sleep mode. Hosts with active VMs running on them should never sleep. A consolidation host is in sleep mode by default and is awakened only to accommodate incoming VMs.

3.2 Changes to Consolidated VM State

The state of a VM in a consolidation host can change over time. An active VM that was consolidated in full, can become idle. Conversely, an idle VM that was only partially consolidated can become active. When VM state changes occur, we use one of the following policies:

1. *Default* – consolidated VMs remain on the consolidation host until the VMs exhaust the host's capacity. The host capacity can be exhausted when partial VMs become active and require the full memory commitment or when they request additional resources as their idle working sets grow. When the consolidation host's capacity is exhausted, the cluster manager wakes up the requesting VM's home host and returns all of its VMs. This strategy is based on the observation that, once a host is awake there is little benefit in leaving its partial VMs on the consolidation hosts. In fact, doing so is wasteful because the partial VMs utilize the memory of their home (equal to the VM's full footprint) as well as the memory on the consolidation host (equal to the idle VM's working set). Migrating back all full VMs that were originally homed on the awake host creates additional space on the consolidation hosts.

   When a partial VM becomes active and the consolidation host has the sufficient resources, the full memory footprint of the VM is transferred from its previous home into its current host which becomes the VM's new home.

2. *FulltoPartial* – is a refinement of the *Default* policy above. When a full VM becomes idle in a consolidation host, it is fully migrated to its home host. The home host is awakened temporarily to accommodate the incoming full VM. The VM is then partially migrated back to the same consolidation host and its home returns to sleep mode. Essentially, the consolidation host exchanges a full idle VM for a partial VM. This step is used to free memory on the consolidation host for future incoming VMs and, as we will show in § 5, it leads to significant energy savings in our evaluation.

3. *NewHome* – is a refinement on *FulltoPartial*. When a partial VM becomes active and exhausts the consolidation host's capacity, it migrates to any other compute or consolidation host that is currently powered and is capable of accommodating it. If no free host is available, we use the same strategy as *Default*, i.e., wake up the VM's home host and migrate back all its VMs. The results of § 5 show that, contrary to our intuition, this optimization turned out to have little additional benefit over *FulltoPartial*.

3.3 Low-Power Memory Page Server

In § 2 we showed that consolidating multiple partial VMs from the same home host causes enough page requests to prevent the host from sleeping. To ensure that the home host is able to sleep, we design a server architecture that embeds a

low-power memory server on the host hardware. This design enables compute hosts to remain asleep while continuing to service page requests from their partial VMs.

We considered two design alternatives for the memory server. In the first, each host implements its own low-power memory server and uses an internal bus for control. In the second, hosts share a network accessible memory server. When each host is about to sleep, it must transfer the full memory of its consolidated partial VMs to the memory server (full VM migrations). As shown previously [4], doing full VM migrations saturates the network and does not scale. As such, we implement per-host memory servers, which improve agility in realizing VM placement plans.

A low-power memory server serves memory pages while the host is in sleep mode. Such a server must meet the following requirements: i) it only consumes a fraction of the host's power, ii) it has access to the memory pages of the host's VMs, iii) it has access to the network. The processor and memory demands of the memory server itself are modest since it does not run VMs and only needs to keep up with the page request rate.

There are several options to implement the memory server. One option is to extend the service processor that is built into many servers, e.g., HP iLO [13], Dell DRAC [12]. The service processor is powered independently and is network reachable. However, current architectures must be extended to support direct access to the host's memory.

An alternative is to use a programmable network interface(NIC) with RDMA capabilities. Unfortunately, existing RDMA cards require the hosts to be fully powered to read their memory. This is partly because RDMA targets high performance applications where high bandwidth and low latency, as opposed to energy efficiency, are the priority.

Any of the above approaches could keep the host's memory in low-power self-refresh mode, only switching individual DIMMs into high power mode momentarily when serving requests for pages stored on them.

A commercial implementation of a low-power memory server requires modifications on the host motherboard. Instead, we build our prototype using existing hardware by augmenting a standard host with a low-power computing platform and a dual mounted SAS drive. We discuss the prototype in detail in § 4. Our results show that even with such a sub-optimal implementation, our approach can yield significant energy savings(§ 5).

4. Prototype

Our prototype consists of a cluster manager, virtual machine hosts, and network storage for the VM images and configurations. Each host contains an agent, a hypervisor and a memory server. Figure 4 provides an overview. We implemented partial VM migration and reintegration on the Xen hypervisor.

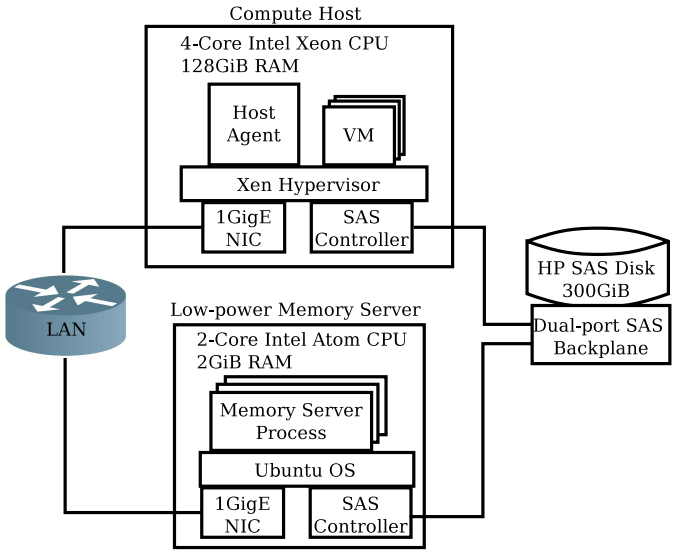

Figure 4. Oasis Prototype Overview

4.1 The Cluster Manager

The cluster manager is responsible for VM creation, migration and shutdown, and switching the hosts between power modes. It provides an RPC interface that clients use to create and manage VMs. Clients create VMs by issuing a request which includes the path of a VM configuration file in the network storage. Each VM configuration file contains a unique four digit $vmid$ used to identify the VM, the path to the VM's disk image, memory allocation, number of virtual CPUs, and device configuration such as network and virtual frame buffer. The manager parses the VM configurations, identifies a host with sufficient resources to accommodate the VM, and issues a VM creation call to the agent of the selected host which, in turn, starts the VM. The agent becomes the owner of the VM.

The cluster manager receives periodic statistics about the host and VM performances from the host agents. Each agent reports the host's memory, CPU, and I/O utilization. It also reports per VM statistics, including memory allocation and resource utilization. The manager uses the policy discussed in § 3 to determine whether to migrate virtual machines. When the manager detects an opportunity for consolidation, it sends a list of tuples to the agent consisting of $< vmid, migration\_type, destination >$, where $migration\_type$ is either partial or full migration and $destination$ is the host identified to receive the VM. Once the agent completes the migration tasks, the manager notifies the agent to suspend the host into sleep mode if it has no running VMs. When the manager determines to place a VM on a host that is currently in sleep mode(either because a partial VM has become active and requires extra resources, or because a new VM is created by a client), the manager wakes up the corresponding host with a network

Wake-on-LAN before issuing the migration or creation call to the agent.

4.2 The Host Agent

The host agent is a user level process that runs on the administrative domain of the host ($dom0$). It performs host level power management operations using the host's ACPI [14] interface, performs host-to-host VM migration, and collects host and VM performance statistics using Xen's xenstat interface.

Partial VM migration. When an agent receives a call to partial migrate a VM to another host, it suspends the VM, uploads the VM's memory to its memory server and pushes the VM's descriptor (including its page tables, configuration and execution context) to the destination host. The receiving agent creates a partial VM with only the frames needed for the received page tables, initializes vCPUs and schedules the VM. When setting up the page tables of a partial VM, the hypervisor marks its page entries as absent which causes page faults whenever the VM attempts to access the pages.

For each partial VM, the host agent creates a *memtap* user level process that is responsible for handling VM page faults and retrieving pages from the corresponding memory server. The memtap is configured with the host and the port number of the memory server containing the pages belonging to the VM. Page fault handling in Xen was extended to allocate frames on-demand and, via an event channel, notify the corresponding memtap process of the pseudo-physical page number of the faulting page and the machine address of the frame on which to store the page. The hypervisor allocates frames at the granularity of a chunk consisting of 2 MiB in order to reduce fragmentation of the host's heap. Once a page is fetched, memtap updates the local frame and notifies the hypervisor to re-schedule the suspended vCPU. While a partial VM runs on the destination host, the VM's ownership remains with the agent of the source host, which controls the memory server responsible for the VM's memory image.

Full VM migration. When the agent receives a full migration request from the manager, it initiates live VM migration [6] to the destination. Once live VM migration is completed the agent frees all resources previously allocated to the VM, including any memory state uploaded to the memory server. The destination becomes the owner of the VM.

VM reintegration. When migrating a partial VM, the consolidation host's agent suspends the VM's execution and pushes the partial VM's memory to the destination. If the destination is the owner of the VM, where its full memory is resident in DRAM, only the dirty state is pushed. The consolidation host uses shadow page tables to track dirty pages of each partial VM. When migrating a partial VM to its owner, the destination reintegrates the dirty state with the full VM memory and returns the VM into execution rapidly. The source then releases the resources that were allocated to the partial VM.

4.3 The Memory Server

We prototype the memory server by pairing a low-power ASUS AT5IONT-I computing platform with an x86 rack server. The memory server is connected to the host via a shared hot-swappable Serial Attached (SAS) hard drive. Before entering low-power mode, the host attaches the drive, writes out all of its VMs' memory pages, detaches the drive and notifies the low-power processor. The low-power processor, attaches the drive and activates its server daemon which services network page requests by their guest pseudo frame numbers. When the host wakes up and its VMs are returned, it notifies the memory server's daemon to stop serving pages and detach the shared drive. The SAS interface provides fast write speeds necessary for the host to push VM memory and to enter sleep mode with minimal delay. It also ensures that memory transfer traffic from the host to the memory server does not reach the datacenter network. In our experiments, the interface was capable of sustaining throughput of 128 MiB/s of sequential writes.

The memory server is equipped with a 1.8 GHz dual-core Intel Atom D525 processor, 2 GiB RAM, a Gigabit NIC, and a 320 GiB internal disk. A CS Electronics ADP-4000 hot-swappable SAS backplane adapter was used to support dual connections from the host and memory server to a shared HP Entry 516814B21 SAS drive with 300 GiB capacity. The host and the memory server use HighPoint RocketRAID SAS controllers (models 2720SGL and 2640SGL) to connect to the SAS adapter. To ensure data consistency, only one device mounted the shared disk at a time.

Memory upload optimizations. When uploading VM memory images to the shared disk, our prototype employs two strategies to reduce the upload latency. First, it uses per-page compression to reduce the page sizes. Each page is compressed using the Lempel-Ziv-Oberhumer [21] real-time compression library before it is written to the memory image and only the *memtap* process of the partial VM decompresses it when servicing a page fault. The memory server accesses and transmits the compressed pages. Second, it performs differential upload, a refinement that uses dirty page tracking to identify and send only the pages that have been dirtied by their VMs since the previous upload to the memory server.

Security. Because the memory server exposes the contents of VMs memory to the network, it is important to ensure that only authorized memtap processes are able to access each VM's memory. Without any security mechanism in place, local area hosts can access VM memory by requesting pages from the memory server, or by eavesdropping on packets transmitted between the server and memtap process. To prevent these attacks the page server and *memtap* client should implement authentication and and encryption using Transport Layer Security (TLS) [8]. The establishment of connections between a client and server using TLS follows a handshake process that establishes the authenticity of

the server and client, and the parameters for encryption of the data to be transferred. Authentication can be established through the use of certificates issued by the enterprise's IT administrator.

4.4 Micro Benchmarks

In this section we use a micro benchmark to characterize the performance of our prototype. We compare the performance of full and partial migration in terms of migration latency and network bandwidth. We also measure the latency for reintegrating a partial migrated VM back to its home server and the performance degradation that a user experiences if their partial-migrated VM is left to run on the consolidation host once it becomes active.

4.4.1 Experimental Setup

We use two enterprise-class servers, and a low-power platform to act as a memory server. The first server is built from custom components to ensure that it is capable of using S3 low-power mode. Few off-the-shelf servers are known to support low-power modes. The server is built with components including a Supermicro X9DAi motherboard, two 2.4 GHz quad-core Intel Xeon(R) E5-2609 CPUs, 128 GiB of DRAM, 1 GigE NIC, and a 1 TB SATA hard drive.

The memory server consists of the ASUS AT5IONT-I platform and the independently powered SAS drive that is shared with the host using the architecture described in §§ 4.3.

The second server is an off-the-shelf HP ProLiant DL560 Gen8 with a 2.20GHz 8-core Intel(R) Xeon(R) CPU E5-4620 and 512 GiB of DRAM, 1 GigE NIC, and a 300 GiB drive. Because this server lacks support for S3, it is designated as the consolidation host and always remains powered. Only the custom host is suspended into low-power when its VMs are consolidated. We envision production deployments of Oasis to consist largely of servers that support low-power sleep mode. The hosts and the memory server are connected over a Gigabit Ethernet network.

Device	State	Time (s)	Power (W)
Custom host	Idle	N/A	102.2
	20 VMs	N/A	137.9
	Suspend	3.1	138.2
	Resume	2.3	149.2
	Sleep (S3)	N/A	12.9
Memory server	Idle	N/A	27.8
SAS drive	Idle	N/A	14.4

Table 1. Energy profiles and S3 transition times.

Table 1 shows the energy profile of our custom host and memory server components. We measure the power of the host system when it is fully idle, hosting 20 active VMs, in sleep mode, and transitions between full and sleep. When the host is in sleep mode, the combined power use of the host and memory server components (55.1 W) does not exceed

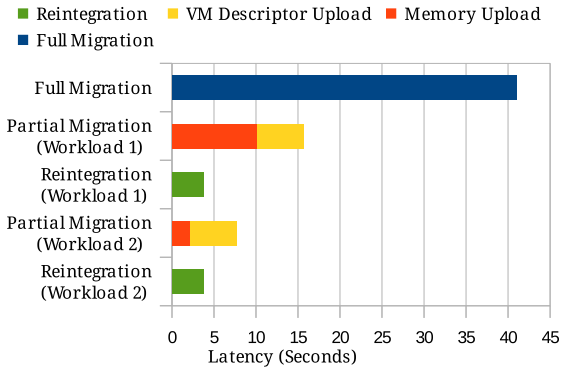

Figure 5. Consolidation latencies for one VM.

the power of an idle host (102.2 W), which indicates an opportunity to save power through consolidation. Arguably, a production memory server implementation would have a much lower power footprint by forgoing the SAS drive and using a more power-efficient processor.

The experiments consisted of a deployment of remote desktop VMs on the custom server, which migrate between the two servers. Each desktop VM was configured with 4 GiB of RAM, 12 GiB disk image hosted on an NFS share, and ran the GNOME desktop environment. After initial deployment on the custom host, the VM's were primed using a script that loads the applications of Workload 1 described in Table 2. The workload mimics a heavily multitasking user who concurrently runs multiple applications. Once all applications in Workload 1 were loaded, the VM became idle for five minutes, after which it was partial migrated to the consolidation server. Partial migration includes uploading the VM pages to the memory host and the VM descriptors to the consolidation host. When VMs ran on the consolidation host, page faults were serviced by on-demand page transfers from the memory server. The partial VMs ran on the consolidation host for twenty minutes, after which they were reintegrated to the custom server. During VM reintegrations, only dirty pages that were modified by the partial VM were transferred back to the custom host to update the old VM memory image. When VMs started running again on the custom host, they performed the tasks listed in Workload 2. This step was used to emulate the event in which users become active and interact with their VMs. Once Workload 2 operations were completed, the VMs remained idle for another five minutes and were partial migrated to the consolidation host for the second time. Note that while the VMs were idle, they continued to run background tasks with low activity (e.g., e-mail client fetches messages periodically, IM client sends keep alive messages). Repeated consolidations were used to evaluate the improvements gained by differential memory upload optimization discussed in §§ 4.3.

Workload	Applications
Workload 1	Thunderbird mail, Pidgin IM, LibreOffice with three documents, Evince with an open PDF, and Firefox with five open sites (CNN.com, Slashdot.com, Maps.Google.com, the SunSpider JavaScript benchmark [26], as well as Acid3 Web standard compliance test [10]).
Workload 2	Adds: Shopping.HP.com, CDW.com, BBC.co.uk/news, and TheGlobeAndMail.com to Firefox; three office documents to LibreOffice; a PDF document to Evince.

Table 2. Desktop workloads.

4.4.2 Consolidation Latencies

Figure 5 shows consolidation latencies for a single VM using both full and two iterations of partial VM migration for the workloads describes above. Results are averages over 3 runs. The full VM migration experiment accounts for the time it takes to live migrate the VM's complete memory image to the consolidation host. The partial migration results include the time to upload the VM memory to the memory server as well as the time to upload the VM descriptor to the consolidation host. The plot also shows the reintegration times for the two partial migration iterations.

As was expected, partial migration is faster than full migration. While it takes an average of 41 s to fully migrate our VM, it only takes 15.7 s and 7.2 s to partial migrate after executing the first and second workloads, respectively. The second partial migration benefits from the differential memory transfer optimizations, which for this use case manages to reduce the time to upload memory to the memory server from 10.2 s to 2.2 s. An alternative implementation of the memory server with access to the host's memory could theoretically reduce the partial migration latency even further to the time it takes to upload the VM descriptor to the consolidation host, which is about 5.2 s on our platform. Finally, reintegration latency (the time it takes to migrate a partial VM back to its host of origin) is also low (3.7s on average for our two scenarios).

4.4.3 Network Traffic

Partial migration generates much less network traffic compared to full migration. Whereas fully migrating our VM required sending its 4 GiB memory images over the network, partial migration transmits only 16.0±0.5 MiB to create the VM on the consolidation host, and 56.9±7.9 MiB to fetch memory on-demand to support its execution on the consolidation host. VM reintegration required transferring 175.3±49.3 MiB of dirty memory. The amount of dirty state that needs to be reintegrated exceeds the total state migrated to the consolidation host because Oasis implements an optimization that obviates the transmission of memory pages that will be completely overwritten, e.g., new memory allocations, recycled file buffers [4].

4.4.4 Idle to Active Transition

Since memory is fetched on demand, applications run in partial VMs suffer from performance degradation. Figure 6 compares the latencies for starting a number of applications

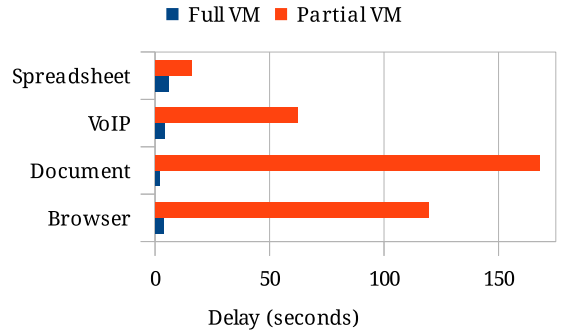

Figure 6. Application start-up latency.

that are commonly used in virtual desktop (VDI) deployments. The figure shows that there is little latency for applications running on full VMs. In contrast, desktop applications take up to 111 times longer to start in partial VMs. For example, starting the LibreOffice document takes 168 seconds. In contrast, pre-fetching all the VM's remaining state takes only 41 seconds on our testing environment. Given these results, when a partial VM becomes active, our policy is to convert it into a full VM by either bringing the remaining VM pages to the consolidation host (*Default* policy for consolidation host with spare capacity), reintegrating the VM into its home host (*Default* policy for saturated consolidation host), or migrating the VM in its entirety into a new available host (*NewHome* policy).

4.5 Discussion

Our memory server prototype is built with a full-fledged PC that can run the SAS adaptor and a shared SAS disk. This setup is not optimal in terms of cost, performance and energy consumption. Nevertheless, our evaluation shows that we can achieve significant energy-savings with this sub-optimal setup(§ 5).

We expect that a commercial implementation would reuse the host memory, which eliminates memory transfer cost and allows the host transition to sleep faster. Also, the full fledge PC can be replaced with an embedded implementation that uses a more energy-efficient processor.

5. Evaluation

In this section, we use trace-driven simulation to evaluate the potential for Oasis to save energy in a cluster deployment.

While Oasis supports the consolidation of arbitrary server workloads, we evaluate its performance in the context of a virtual desktop infrastructure (VDI) server farm. We answer the following research questions:

1. How much idleness is there in a VDI deployment?
2. Does Oasis save energy in VDI clusters?
3. How effective are the *FulltoPartial* and *NewHome* migration policies?
4. How large are the network transfers of hybrid server consolidation?
5. What is the user perceived latency of consolidation?
6. How sensitive is Oasis performance to cluster size?
7. How much extra energy could be save with a more optimal memory server implementation?

5.1 Simulation Environment

We simulate an Oasis cluster consisting of a standard server rack with 42 identical 1U servers connected through a top-of-rack 10GigE switch. Each host has a low-power memory server as described in §§ 4.3. All hosts share the same energy profile shown in Table 1. Every host is designated to act as either a *home* host or as a *consolidation* host. When a home host is in S3 mode, its low-power memory server is turned on and consumes power. Low-power memory servers attached to consolidation hosts are not used and are therefore not powered at any time.

We configured 30 hosts to act as home hosts, and varied the number of consolidation hosts between 2 and 12. Each home host was assigned to host 30 VMs. We assigned each VM 4 GiB RAM and 1 vCPU. The VM's virtual disk is hosted by a separate well-provisioned network-attached storage system.

The simulations assume that a full VM requires all of its RAM, i.e., the host guarantees its 4 GiB RAM is present. For a partial VM, its memory consumption is randomly sampled from the distribution collected from [4] which shows that the mean working set of idle desktop VMs with 4 GiB RAM was only 165.63 ± 91.38 MiB, less than 4% of VM's allocated memory. Base on the data reported in [7], we assume that fully migrating a 4 GiB VM over 10 GigE takes 10s. For partial migrations, we use the conservative parameters from 4.4.2. Partially migrating an idle VM (including memory upload) takes 7.2s and resuming a partial VM takes 3.7s. Suspending a server to RAM takes 3.1s and resuming takes 2.3s(Table 1).

To drive the simulation, we use a trace consisting of the user desktop activity collected on the desktops and laptops of 22 researchers over a period of four months [4]. We used a Mac OS X tracker to record every 5 seconds whether the user was active or idle. User activity was detected by polling for keyboard or mouse activity. The trace encompass 2086 user days, of which 1542 days are weekdays and 544 weekends. In each simulation run, we randomly sample 900 user weekdays from traces, align them into one day and treat them as if there are 900 different users. We repeat the same process for weekends. Each user-day is then divided into 5-minute intervals. For each interval, if there is any keyboard or mouse activity, we mark that interval as active, or idle otherwise.

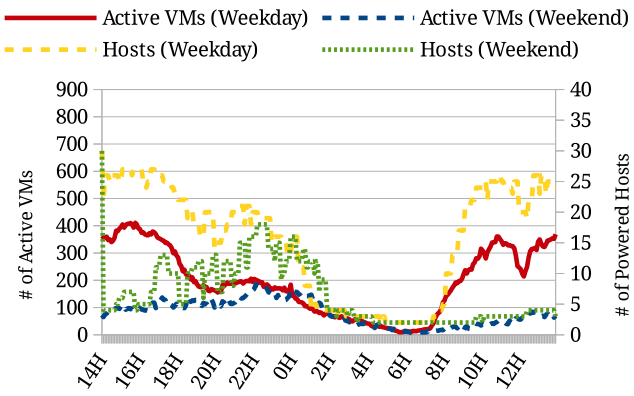

Figure 7. Number of active VMs and fully powered host over a simulation day for a cluster of 30 home and 4 consolidation hosts. We use the *FulltoPartial* policy in this simulation run.

5.2 VM Activity and Cluster Sizes

In this section, we show that our hybrid consolidation approach is able to adapt the size of the cluster to closely match the mix of active and idle VMs found in our traces throughout the course of the day. Figure 7 shows the variation in the number of active VMs over a sampled weekday and weekend. For the weekdays, there are diurnal activity patterns: The level of activity reaches its peak at around 2pm and keeps falling until it arrives at the bottom at 6.30am. Interestingly, there are never more than 411 (46%) active VMs simultaneously. Unsurprisingly, we see lower activity rates over the weekends.

Figure 7 also shows how well *FulltoPartial* policy adapts the size of the cluster with the number of active VMs. Other policies (not shown) show similar ability to adapt the size of the cluster. The figure plots the total number of fully powered hosts (both home and consolidation) over a simulation day. The number of fully powered hosts goes down when the VMs are consolidated and the hosts transition into sleep mode, and goes up when the level of VM activity resumes. At one point, all 900 VMs get consolidated into just three consolidation hosts when the number of active VMs reaches its lowest level.

5.3 Energy Savings

Figure 8 shows the energy savings for different Oasis policies on a cluster consisting of 30 home hosts as we vary the

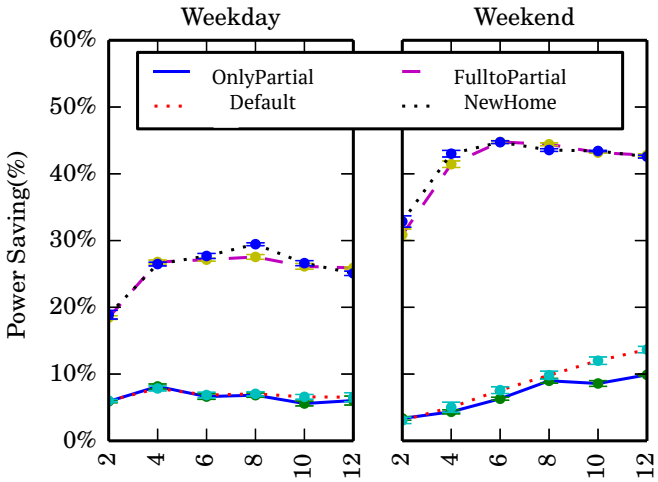

Figure 8. Energy savings for a simulation day for a cluster of 30 home hosts.

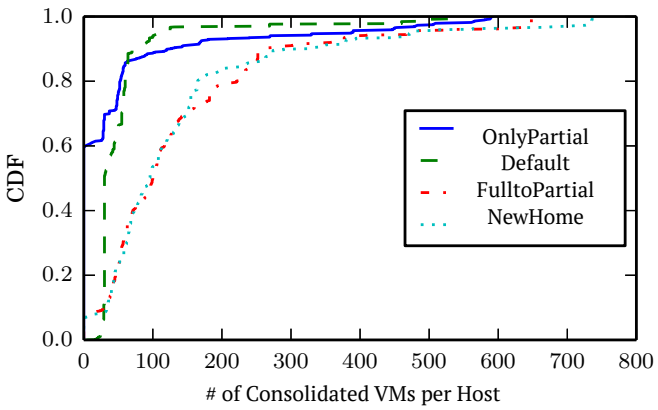

Figure 9. CDF of consolidation ratio

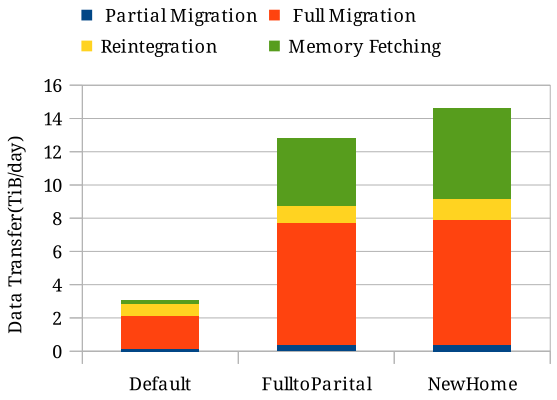

Figure 10. Weekday data transfer breakdown

number of consolidation hosts. Results are the averages of five different runs. The plot also shows small standard deviations, represented with error bars on each data point.. The savings are normalized over the energy consumed by the home hosts if left powered for the duration of the simulation.

Energy savings increase initially as we add more consolidation hosts until we have sufficient capacity to host all idle (and a few active) VMs and then level off. Across all policies, the best energy savings gain with the minimal number of consolidation hosts is achieved with four consolidation hosts. We use this configuration for the rest of the evaluation unless stated otherwise.

The results show that the approach that makes exclusive use of partial VM migration (*OnlyPartial*) achieves very limited energy savings (about 6%). This is not surprising because on average all of the VMs assigned to a home host are simultaneously idle only 13% of the time. The basic approach that simply combines full migration with partial migration (*Default*) when consolidating host performs only marginally better. This approach ends up running too many full VMs on the consolidation hosts and, as a result, achieves limited consolidation ratios. In contrast, the *FulltoPartial* approach, which migrates consolidated full VMs that become idle back to their home hosts and re-consolidates them back as partial VMs, increases power savings to 28% on weekdays and 43% on weekends. *FulltoPartial* takes advantage of the fact that consolidated full VMs often become idle after a short period of time, and that re-consolidating them as partial VMs can free more memory on the consolidation hosts, which can be re-used to accommodate additional VMs. The increase in the consolidation ratio can be seen in Figure 9. For example, the median number of VMs running on a consolidation host (the 50% on the CDF plot) increases from 60 with the pure *Default* approach to 93 with the *FulltoPartial* policy.

Somewhat surprisingly, the more complex *NewHome* policy does not achieve additional saving beyond the *FulltoPartial* policy. This is also evident in Figure 9 in the overlap of plot lines for the two policies. Therefore, we forego the *NewHome* policy and use the *FulltoPartial* policy in the rest of our evaluation.

5.4 Network Traffic

Figure 10 shows a breakdown of the network transfer volumes of various policies. The figure shows that the *FulltoPartial* policy leads to an increase in both partial and full migration traffic. The extra traffic results from the migrations of the consolidated full VMs that become idle back to their home(s) and then re-consolidating them as partial VMs, as well as the migrations of any additional full and partial VMs that can now be accommodated in the newly freed space. In effect, the *FulltoPartial* policy trades energy for network traffic. We argue that this is an acceptable trade-off for the deployments where the home and consolidation hosts are co-

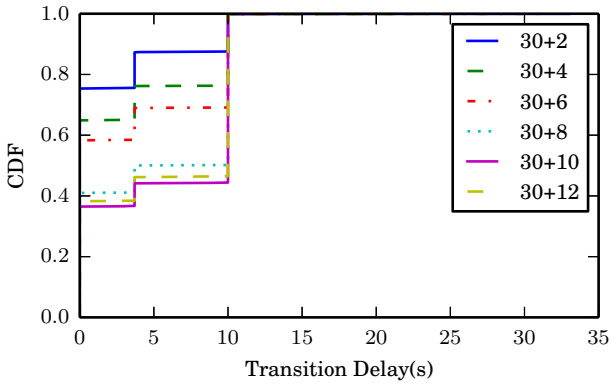

Figure 11. Idle→Active transition delay distribution for different combinations of home and consolidation host numbers.

located on the same rack and the bandwidth among hosts is abundant.

5.5 User Perceived Latency

When the users resume their activity, their VMs are expected to possess all their assigned resources and deliver fast responses. Any delay of the VMs acquiring resources impacts the user experience. We collected the latency distribution of all idle→active delays in our simulation(Figure 11). If the transition happens in a full VM, the latency is zero since the VM has already acquired all its assigned resources. This type of transitions takes the majority of the cases. Interestingly, as we increase the number of consolidation hosts from 2 to 12, the probability of zero latency drops from 75% to 38% because it is more likely for a transitioning VM to reside in a consolidated host which incurs the VM re-integration latency. More interestingly, when transitions happen in a partial VM, the impact of VM re-integrations is small. Users can expect to experience the delay of less than four seconds. Even in the worst case when there is a VM resume storm, the reintegration latency only reaches up to 19s (99.99 percentile). We believe that the low probability and the small magnitude of VM re-integration latencies have limited impact on user productivity.

Memory Server Power(Watt)	Energy Saving	
	Weekday	Weekend
Current Prototype 42.2	28%	43%
16	34%	59%
8	37%	65%
4	39%	66%
2	41%	67%
1	41%	68%

Table 3. Alternative memory server implementations

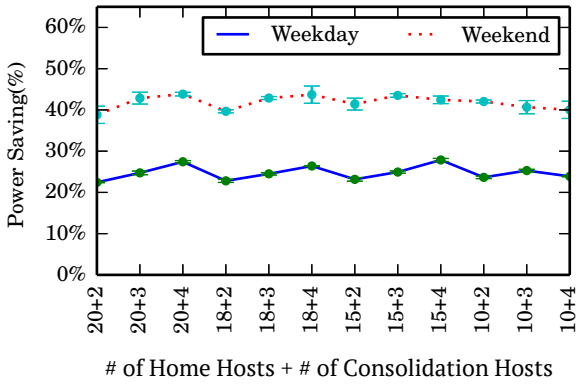

Figure 12. Sensitivity analysis with different cluster sizes.

5.6 Results Sensitivity and Generality

Table 3 explores the benefits of alternative implementations of the memory server with power budgets between 1 and 16 watts. It is clear that a more efficient implementation can significantly improve Oasis' effectiveness and reaches up to 41% and 68% for weekdays and weekends, respectively.

We also evaluate the sensitivity of our results to the number of VMs assigned to a home host as well as the number of home and consolidation hosts used. We keep the total of 900 VMs unchanged, and then vary the server capacity to host 45, 50, 60 and 90 VMs. Figure 12 shows that the power saving are similar, independent of the number of VMs assigned to a home host.

Finally, while this evaluation is based on traces that emulate a VDI server farm deployment, we argue that other server workloads are likely to exhibit similar performance. We base this argument on the observation made in § 2 that idle desktop VMs are usually more demanding in their memory access than idle Web server or database VMs. We postulate that this is due to the nature of desktop VMs, which run a wide range of applications and background services, whereas Web and database server VMs tend to be single-purposed.

6. Related Work

Previous work has relied on full VM migration to reduce energy consumption by consolidating VMs and switching hosts to low-power mode [5, 9, 15, 22, 28]. Unfortunately, full VM migration requires the target host to have enough resource slack to accommodate the incoming VMs, resulting in low consolidation ratios. In contrast, Oasis implements a hybrid approach that uses partial and full VM migration to achieve very high consolidation ratios and save energy.

Jettison [4] introduces the use of partial VM migration to save energy in office environments by consolidating idle desktops. In contrast, this paper uses a combination of partial and full migration to save energy in the data center. The

latter is a more challenging environment as the co-location of multiple VMs in a single host results in frequent memory requests that would prevent the original Jettison implementation (which relies on waking up the host to serve memory requests) from saving any energy. Instead, Oasis offloads the tasks of serving memory requests to a low-power page server.

Oasis differs from hierarchical power management systems, such as Turducken [23] and Somniloquy [1], where the application functionality migrates to and executes on the low-power component. Oasis only requires an energy efficient memory server without the need of application modifications or protocol aware proxies.

PowerNap [18] describes the mechanisms to eliminate idle power waste by letting servers quickly transition between high and low-power modes in response to workloads. Isci et al. [15] describe a virtual server platform that supports low-latency low-power modes. This work is complementary to ours, as faster power mode transitions could be leveraged to reduce the server reintegration latency.

Oasis is similar to KnightShift [27], a server-level heterogeneous server architecture. But unlike KnightShift, The low-power memory server in Oasis is not a general purpose computing device. It can be made simple because it serves one function: serving memory pages over the network.

Finally, Oasis makes use of remote memory as an energy-efficient swap for VM state. Systems, such as DLM [19] and Nswap [20], also use the cluster memory to replace disk swapping, but their focus is on supporting large working sets and improving the page swapping latency, as opposed to reducing the overall cluster energy utilization.

7. Conclusion

In this paper, we propose Oasis, a new approach for energy-oriented cluster management. Oasis achieves high consolidation ratios by combining traditional full VM migration with partial VM migration, a technique that migrates only the limited working set of an idle VM, which is typically an order of magnitude smaller than the VM's memory allocation. Partial VM migration operates by creating a partial replica of the VM on the consolidation host and transferring memory pages on-demand as the VM attempts to access them. Traditional partial VM migration wakes up the sleeping host to serve memory requests, which yields little energy savings when applied in servers that host multiple VMs and experience frequent page requests. Oasis overcomes this challenge by augmenting the host with a low-power memory server that can efficiently serve VM memory state without interrupting the host's sleep mode.

We implemented Oasis by extending the Xen hypervisor. We built a prototype of a low-power memory server using existing hardware by augmenting a standard host with a low power computing platform and a shared SAS drive. Whereas this prototype is suboptimal in terms of cost, performance and energy consumption, it nevertheless demonstrates the benefits of the approach. We evaluated our implementation using a combination of micro benchmarks, and a simulated virtual desktop (VDI) server farm. The results show that Oasis reduces energy usage by up to 28% on weekdays and 43% on weekends with minimal impact on the user experience.

Acknowledgments

We would like to thank the anonymous EuroSys'16 reviewers for providing helpful comments and suggestions. We also thank our shepherd, Mahesh Balakrishnan, for additional help improving the quality of the paper.

References

[1] Y. Agarwal, S. Hodges, J. Scott, R. Chandra, P. Bahl, and R. Gupta. Somniloquy: Augmenting network interfaces to reduce PC energy usage. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI '09)*, Boston, MA, Apr 2009.

[2] I. Banerjee, F. Guo, K. Tati, and R. Venkatasubramanian. Memory overcommitment in the ESX server. *VMware Technical Journal*, 2, 2013.

[3] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *IEEE Computer*, 40(12):33–37, 2007.

[4] N. Bila, E. de Lara, K. Joshi, H. A. Lagar-Cavilla, M. Hiltunen, and M. Satyanarayanan. Jettison: Efficient idle desktop consolidation with partial VM migration. In *ACM European Conference on Computer Systems (EuroSys '12)*, Bern, Switzerland, Apr 2012. doi: 2168836.2168858.

[5] N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing SLA violations. In *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, pages 119–128. IEEE, 2007.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *2nd Conference on Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, MA, May 2005.

[7] U. Deshpande, U. Kulkarni, and K. Gopalan. Inter-rack live migration of multiple virtual machines. In *Proceedings of the 6th international workshop on Virtualization Technologies in Distributed Computing Date*, pages 19–26. ACM, 2012.

[8] T. Dierks and E. Rescorla. The TLS protocol: Version 1.2. `https://tools.ietf.org/html/rfc5246`, Aug 2008.

[9] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu. Vmware distributed resource management: Design, implementation, and lessons learned. *VMware Technical Journal*, 1(1):45–64, 2012.

[10] I. Hickson. The Acid3 test. `http://acid3.acidtests.org/`, Jul 2012.

[11] M. R. Hines and K. Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)*, Washington, DC, Mar 2009.

[12] D. Inc. Dell DRAC. http://www.dell.com/content/topics/global.aspx/power/en/ps2q02_bell?c=us, Jan 2015.

[13] H. Inc. HP iLo. http://www8.hp.com/us/en/products/servers/ilo, Jan 2015.

[14] T. Intel, Microsoft. Advanced configuration and power interface specification http://www.acpi.info/DOWNLOADS/ACPIspec10b.pdf, Feb 1999. [Last Accessed: 2015-01-27].

[15] C. Isci, S. McIntosh, J. Kephart, R. Das, J. Hanson, S. Piper, R. Wolford, T. Brey, R. Kantner, A. Ng, et al. Agile, efficient virtualization power management with low-latency server power states. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA'13)*, pages 96–107. ACM, 2013.

[16] H. Liu. A measurement study of server utilization in public clouds. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, 2011.

[17] S. D. Lowe. Best practices for oversubscription of CPU, memory and storage in vsphere virtual environments. *VMware's White paper, available at: https://communities. vmware. com/servlet/JiveServlet/previewBody/21181-102-1-28328/vsphereoversubscription-best-practices [1]. pdf*, 2013.

[18] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: eliminating server idle power. *ACM SIGARCH Computer Architecture News*, 37(1):205–216, 2009.

[19] H. Midorikawa, M. Kurokawa, R. Himeno, and M. Sato. DLM: A distributed large memory system using remote memory swapping over cluster nodes. In *Cluster Computing, 2008 IEEE International Conference on*, pages 268–273. IEEE, 2008.

[20] T. Newhall, S. Finney, K. Ganchev, and M. Spiegel. Nswap: A network swapping module for linux clusters. In *Euro-Par 2003 Parallel Processing*, pages 1160–1169. Springer, 2003.

[21] M. F. Oberhumer. LZO real-time data compression library. http://www.oberhumer.com/opensource/lzo/, Aug 2011.

[22] A. Singh, M. Korupolu, and D. Mohapatra. Server-storage virtualization: integration and load balancing in data centers. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, page 53. IEEE Press, 2008.

[23] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins. Turducken: Hierarchical power management for mobile devices. In *3rd International Conference on Mobile Systems, Applications and Services (Mobisys 2005)*, Seattle, WA, USA, Jun 2005.

[24] H. Takahiro, H. Nakada, S. Itoh, and S. Sekiguchi. Making VM consolidation more energy-efficient by postcopy live migration. In *The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, Rome, Italy, 2011.

[25] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu. Delivering energy proportionality with non energy-proportional systems-optimizing the ensemble. *HotPower*, 8: 2–2, 2008.

[26] webkit. SunSpider 0.9.1 JavaScript Benchmark. http://www.webkit.org/perf/sunspider-0.9.1/sunspider-0.9.1/driver.html, Jul 2012.

[27] D. Wong and M. Annavaram. Knightshift: Scaling the energy proportionality wall through server-level heterogeneity. In *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*, pages 119–130. IEEE, 2012.

[28] Z. Xiao, W. Song, and Q. Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *Parallel and Distributed Systems, IEEE Transactions on*, 24 (6):1107–1117, 2013.