DIMMer: A case for turning off DIMMs in clouds.

Dongli Zhang, Moussa Ehsan, Michael Ferdman, Radu Sion
Stony Brook University
{dozhang, mehsan, mferdman, sion}@cs.stonybrook.edu

Abstract

Lack of energy proportionality in server systems results in significant waste of energy when operating at low utilization, a common scenario in today's data centers. We propose DIMMer, an approach to eliminate the idle power consumption of unused system components, motivated by two key observations. First, even in their lowest-power states, the power consumption of server components remains significant. Second, unused components can be powered off entirely without sacrificing server availability. We demonstrate that unused memory capacity can be powered off, eliminating the energy waste of self-refresh for unallocated memory, while still allowing for all capacity to be available on a moment's notice. Similarly, only one CPU socket must remain powered on, allowing unused CPUs and attached memory to be powered off entirely. The DIMMer vision can improve energy proportionality and achieve energy savings. Using a Google cluster trace as well as in-house experiments, we estimate up to 50% savings on DRAM and 18.8% on CPU background energy. At $0.10/kWh, this corresponds to 0.6% of total data center cost.

Categories and Subject Descriptors K.6.0 [*Management of Computing and Information Systems*]: General

General Terms Economics, Management, Measurement

Keywords Cloud Computing, Energy, DRAM

1. Introduction

To handle hundreds of millions of users and their associated transactions, companies such as Amazon, Facebook, and Google run immense data centers with until-recently unimaginable computation and storage capacities. As online services become pervasive, projections indicate that electricity consumed in global data centers worldwide in 2010 was more than 200B KWh, between 1.1% and 1.5% of worldwide electricity use [28]. Three years ago, Google announced that their facilities have a continuous electricity usage equivalent to powering 200,000 homes [19].

Surprisingly, despite energy being one of the top three data center operating costs [20], much of the data center energy is wasted because data centers cannot modulate capacity according to demand. Even when experiencing frequent periods of complete inactivity (idle periods upwards of one second [34] during times of low utilization), servers are kept operating at full capacity. As a result, a report by the New York Times found energy waste upwards of 90% as the facilities are operated at full capacity regardless of demand [18].

Industry has a number of energy saving principles and mechanisms, such as consolidation, virtualization (for increased utilization), decommissioning of unused servers, and energy-efficient hardware [2]. Such mechanisms show promise and research demonstrates that job consolidation and server power-off strategies can result in up to 50% energy savings [44]. Nevertheless, despite their theoretical promise, these techniques are rarely used due to the need for fast response times to instantaneous demand and the increased failure rates of mechanical components such as hard disks and fans due to frequent power cycling [36].

Motivated by the fact that complete server power-off strategies are not appropriate in many data centers, we propose an alternative that can modulate energy use based on capacity demand by powering off independent hardware components. In this paper, we envision DIMMer, a system to provide an agile framework for workload-driven scalability and power reduction in data centers. DIMMer would power off all idle DRAM ranks (physical subdivisions of memory capacity) in data center servers during low resource utilization to save DRAM background power. Prior work either applies dynamic voltage and frequency scaling (DVFS) to DRAM [15, 16], or maximizes the time that DRAMs spend in low-power modes [25, 30, 43] (i.e., self-refresh mode). We observe that dynamic control of memory *capacity* presents an opportunity to reduce energy consumption and promote power proportionality of server systems. While prior work has reduced the time that DRAMs spend in high-power modes, we find that dynamically reducing the mem-

SoCC '14, November 03 - 05 2014, Seattle, WA, USA.
Copyright 2014 ACM 978-1-4503-3252-1/14/11$15.00.
http://dx.doi.org/10.1145/2670979.2670990

ory capacity available to the system can yield greater benefits: energy costs for additional DRAM capacity are paid only when this capacity is requested by the system. Current systems waste this energy, because when the DRAM is in self-refresh mode, the power consumption of a 4GB DIMM is approximately 1W (20% of the precharge-standby power [15]). Furthermore, because self-refresh power is proportional to DRAM capacity, the savings of DIMMer are likely to be higher in future systems. In this paper, we use publicly available traces from a production Google data center to demonstrate the effectiveness of DIMMer.

A further benefit of DIMMer over switching DRAM to self-refresh mode [43] is that freeing DIMM contents and powering them off permits also powering down unused CPUs to which these DIMMs are attached, whereas using self-refresh modes and retaining memory contents forces keeping the CPUs powered up, even if all cores in those CPUs are unused. However, to power off DRAM ranks, DIMMer requires disabling DRAM channel interleaving at boot, migrating memory pages among DRAM ranks, and reducing the memory available for disk cache, which may have an impact on system performance. Although DIMMer requires modification to the OS kernel and hardware configuration, there are no modifications to the applications [12, 32]. Finally, many of DIMMer's prerequisites are already implemented in prior work [12, 27, 32, 43].

Applying DIMMer to Google cluster traces [37] demonstrates that background DRAM and CPU energy consumption can be reduced by up to 50% and 18.8%, compared to switching DRAM into self-refresh mode. At $0.10/kWh, this corresponds to 0.6% of total data center cost.

2. Motivation for Powering off Components

2.1 DRAM Power Consumption

Figure 1 shows the typical power consumption of server DIMMs under nominal use. Measured DIMMs (Samsung 1600MHz Dual-Rank ECC) were physically isolated in a dedicated socket – the illustrated data is for one of the 8GB DIMMs, installed alone at the second CPU of a PowerEdge M620 server.[1] Power consumption was measured with increasing throughput up to 2GB/s, a reasonable upper bound for modern data center workloads [33]. According to [33], many workloads, such as web search, memcached, bzip, and gcc, use at most 2GB/s. We find that simply keeping a DIMM powered on with near-zero memory traffic has a constant power consumption of 2.3W, which constitutes more than 50% of each DIMM's power consumption observed at peak throughput for cloud workloads.

2.2 CPU Power Consumption

Our experience shows that individual core power consumption varies significantly across different CPUs within the same server and is difficult to measure precisely and consistently (it varies with the number of per-node DIMMs, channels, utilization, etc.). For the purpose of this evaluation, we chose to be conservative and consider power consumption measured across CPUs. The measured power consumptions of our test system while completely idle are as follows: for 1 CPU (Intel® Xeon® CPU E5-2650 2.00GHz) and 4 DIMMs: 32W; for 2 CPUs and 4 DIMMs: 48W; and for 2 CPUs and 8 DIMMs: 52W. We find that keeping an idle CPU powered up only to maintain contents of the 4 DIMMs attached to it consumes 16W, indicating significant opportunity to save energy by powering down the CPU in addition to memory at times of low utilization.

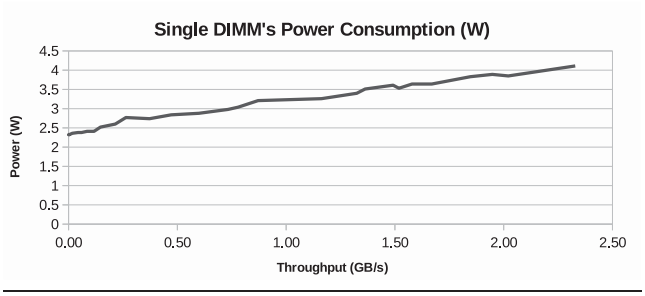


Figure 1. 8GB server DIMM power consumption as a function of throughput. Power at near-zero throughput is more than half of the power at peak utilization.

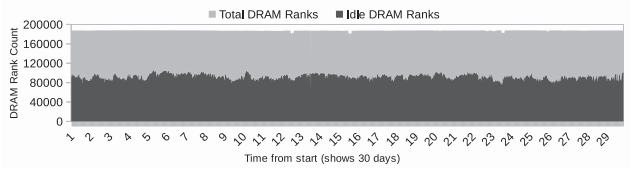
3. DIMMer Benefits

In an ideal world, servers can be powered on or off at zero latency and cost. In that world, one would strive to do exactly that; components would be powered off as soon as a server's utilization can be reduced to zero through migrating applications and consolidation.

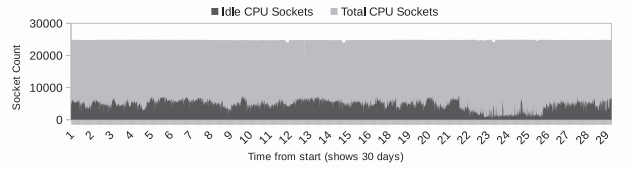
The real world however, imposes a set of rigid latency-related constraints on this vision that cannot be ignored. Migrating jobs and powering servers off and back on are high latency and high cost operations. Additionally, often (as in the case of the Google dataset [37] discussed below) individual servers may participate in distributed services (e.g., file serving) that preclude powering off machines because they must be able to serve file contents on a moment's notice. As a result, clouds end up operating hundreds of thousands of servers at full capacity even at periods of low load, in anticipation of spontaneous demand spikes [11].

In this paper, we show that DIMMer, which would dynamically provision memory capacity at the granularity of DRAM ranks, represents an opportunity to reduce energy consumption of server systems while having little to no impact on performance. To demonstrate that DIMMer is a viable design in practical settings, we analyze the main tradeoffs and show that: (i) idle power consumption of DRAM ranks is high and warrants full power-off, (ii) existing clouds feature numerous- and long-enough (per-server, per-rank) idle periods to justify the DIMMer overheads and latencies, (iii) the resulting cloud power savings are significant.

[1] DRAM power was measured with Intel® PCM [5].



(a) Idle server memory, measured at rank granularity



(b) Idle CPUs, measured at CPU socket granularity

Figure 2. Total number of idle DRAM ranks and CPUs cluster-wide in each 5-minute time slot.

3.1 Idle Resources in the Cloud

Google Cluster Traces (GCT). We analyzed a Google cluster usage dataset [37]. The dataset consists of workload traces for over 12,000 servers collected at 5-minute granularity over the course of more than one month. The traces include detailed information about the servers and the workload jobs and tasks, including CPU, memory, and storage per task, and machine resource utilization. This dataset has spawned a number of seminal results [3]. DRAM details and server counts used in this study are listed in Table 1.

We first compute the number of idle DRAM ranks and CPU sockets of each server in every 5-minute interval. Based on this, we derive the total cluster-wide number of idle memory ranks and CPUs for each 5-minute time slot. Note that, because OS kernel resource usage and job dependencies are not available in the traces used, our approximation of savings is optimistic. Figure 2(a) illustrates the idle DRAM results. We find that, assuming memory can be consolidated on a subset of ranks within each server, on average 50% of the cluster DRAM ranks would be unused and could be powered off.

Figure 2(b) illustrates the number of CPUs that can be powered off in each 5-minute time slot. Unlike DRAM ranks that can be powered off independently, a CPU can be powered off only when all of its cores *and* all DRAM ranks attached to its socket are idle. We note that powering off a CPU may have other system implications, such as rendering some PCIe devices unavailable, which may place additional constraints on DIMMer. This provides opportunity for future research, for example, in powering off some of a system's NICs for further energy reduction. In the analyzed trace, on average, 20% of the CPUs can be powered off across Google's cluster, while the aggregate CPU utilization is 50%. Changes to the cluster resource management framework can mitigate this inconsistency and maximize the number CPUs that can be powered off. However, for the remainder of this study, we use the exact data available in GCT, showing DIMMer's effectiveness with the existing job placement policies.

Figure 3 shows that, in one month, DIMMer would save 30MWh and 52MWh on DRAM and CPU, respectively.[2] The savings achieved by powering off CPUs is actually greater, but can be achieved only together with powering off the associated DRAM. Using the cost model from [21], we estimate the corresponding cost saving over the Total Cost of Ownership (TCO), including data center construction, IT equipment, and operating cost at 0.6% (over total cost), 1.4% (over total power cost) and 3.1% (over total power cost, excluding power for cooling).[3] These energy savings also translate to a significant reduction in environmental pollution. According to the EPA Emissions & Generation Resource Integrated Database (eGRID) [1], this corresponds to a U.S. annual non-baseload CO_2 output emission reduction of over 51 metric tons of CO_2. If the price of electricity is lower, the savings (in terms of dollars) are also lower, due to the dominating costs of server hardware and other data center infrastructure [21]. At \$0.05/kWh, the savings of DIMMer over total data center cost would change from 0.6% to 0.3%.

Normalized Memory Capacity	Frequency in Dataset	Capacity	Ranks	Channels
0.03	5	-	Ignored	-
0.06	1	-	Ignored	-
0.12	54	8G	Ignored	-
0.25	3990	16G	8	4
0.5	6732	32G	16	4
0.75	1002	48G	24	6
1	799	64G	32	8
Total Machines:	12583			

Table 1. The dataset used for our study provides relative memory capacities normalized to the maximum-capacity machine present in the cluster. We estimate per-machine DRAM capacity and number of DIMMs based on the distribution of the machine counts in the dataset, the approximate date of cluster deployment, and the fact that Google populated all server DRAM slots at that time [38]. Machines with unusual memory capacities (likely due to partial memory failures) were ignored.

3.2 Power-off vs. Self-refresh

Prior work proposed maximizing the time DRAM ranks spend in low-power self-refresh mode [25, 30, 43]. Although these techniques reduce DRAM power, the background power consumption of an 8GB DIMM in self-refresh mode with a typical cloud workload is actually higher than

[2] Using DRAM power estimates from [15].

[3] We assume \$0.10/kWh, \$50M facility cost, and 1.2 PUE for 12,583 servers. Facility and IT capital costs are amortized over 15 and 3 years, respectively.

Case	Percentage of Time in ACT_STBY/PRE_STBY					CPU	Self Refresh	STBY	Total
	Node	Rank 0/1	Rank 2/3	Rank 4/5	Rank 6/7	(W)	(W)	(W)	(W)
Self-Refresh [43]	Node 1	100	100	100	100	16	0	21.44	66.00
(2 idle ranks)	Node 2	100	100	0	0	16	1.84	10.72	
DIMMer	Node 1	100	100	100	100	16	0	21.44	64.16
(2 idle ranks)	Node 2	100	100	0	0	16	0	10.72	

Table 2. Sample system with 2 CPU sockets, each having two channels with 8 ranks. In an ideal case, the system would consume 66W when consolidating hot memory pages on "hot" ranks and switching the "cold" ranks to self-refresh mode (using [43]'s approach). Using DIMMer, if we instead power off the "self-refresh"ed ranks we can save an additional 3% of the **background** power consumption.

Case	Percentage of Time in ACT_STBY/PRE_STBY					CPU	Self Refresh	STBY	Total
	Node	Rank 0/1	Rank 2/3	Rank 4/5	Rank 6/7	(W)	(W)	(W)	(W)
Self-Refresh [43]	Node 1	100	100	100	100	16	0	21.44	57.12
(1 idle node)	Node 2	0	0	0	0	16	3.68	0	
DIMMer	Node 1	100	100	100	100	16	0	21.44	37.44
(1 idle node)	Node 2	0	0	0	0	0	0	0	

Table 3. The main opportunity arises when powering off all of the ranks – this corresponds to 20% memory nodes in Google cluster as in Figure 2(b). If all hot memory pages are migrated to the "hot" memory node and the "cold" node's ranks are placed in self-refresh mode, 57.12W is consumed. If "cold" ranks are powered off completely, the entire CPU socket can be powered off, resulting in an additional saving of 35% of the total **background** power consumption.

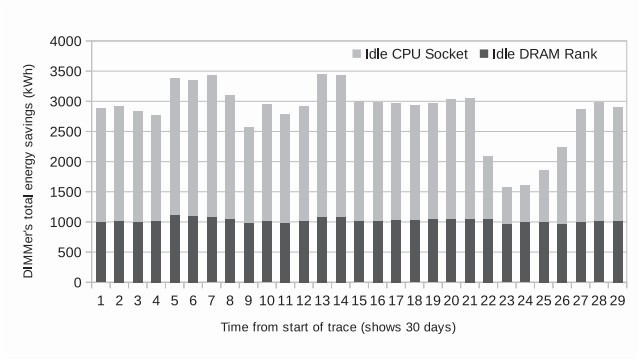


Figure 3. Energy saved by powering off the idle DRAM ranks and CPUs, respectively.

the savings achieved by switching DRAM into self-refresh mode, as shown in section 2.

Table 2 shows that DIMMer would significantly reduce power consumption by powering off idle memory and CPUs. For example, a dual-socket server where each CPU is connected to eight memory ranks (four dual-rank DIMMs) across two memory channels, will consume 66W with techniques that arrange memory into "cold" and "hot" ranks, saving energy on "cold" ranks by putting them into self-refresh mode [43]. Powering off "cold" ranks entirely, DIMMer would additionally save 5.4% of the DRAM power.

Furthermore, the main opportunity for power savings arises when all of the ranks attached to a CPU can be powered off (e.g., as would be the case in the Google cluster, where DRAM utilization of many servers often falls below 50%). In this case, DIMMer would further reduce background power consumption by powering off idle memory ranks *and* their corresponding CPUs. As shown in Table 3, this results in an additional 35% reduction in the background power consumption when compared to the self-refresh approach. Because hardware manufacturers do not currently support fully powered-off DRAM, the exact latency of this operation is unavailable. However, it is expected to be in the same order of magnitude as powering DRAM ranks on or off, which is several hundreds of nano seconds [4, 8]. This latency is imperceptible from the perspective of a sudden server load spike.

4. Vision for Implementation

This paper presents DIMMer as a vision. However, actual implementation is not complex. DIMMer requires modifications to the memory management subsystem of the OS kernel. Unlike the traditional Linux kernel which maintains a memory free list for each memory zone (ZONE_DMA, ZONE_DMA32 and ZONE_NORMAL), DIMMer's **allocator** creates a free list for each DRAM rank. Similar functionality has already been implemented in [12, 27]. The difference in page allocation between DIMMer and standard Linux lies in the total number of free lists. Similar to [24], a page **migrator** running as a kernel thread would be responsible for on-demand memory page migration, moving cold pages from "cold" ranks. Unlike prior work [12, 32], no libraries or application would need to be modified.

For reliable deployment, DIMMer *may* also require hardware changes. Flikker [32] has already proposed to reduce the memory refresh power consumption by decreasing DRAM refresh rate. Theoretically, we can remove most self-refresh power by setting refresh rates to zero. However, to reduce the total self-refresh power to zero, we hope hardware manufacturers will expose registers that allow *full electrical power-off* for entire DRAM ranks in the next generation

DRAM controllers.[4] Support for full electrical power-off of CPU sockets may also be helpful.

5. DIMMer Costs

5.1 Cost of Page Migration

Before powering off unused ranks, DIMMer would migrate memory pages (generally of 4KB sizes) for consolidation onto the active ranks. To estimate the energy cost of page migration, we measured the energy consumption of migrating 8GB of memory (four DRAM ranks) from one NUMA node to another. The node-to-node migration is the most expensive migration that would happen in DIMMer. The integrity check of the migrated pages may need to be performed with checksum. The live migration of hot pages may also pose a significant problem when considering the impact in cache coherency and integrity. These are future works.

The average measured energy cost to migrate a single page is 102μJ. As Table 4 shows, if DIMMer would migrate 8GB every 30 minutes, the additional monthly energy cost of page migration for the cluster would be approximately 210KWh, a mere 0.26% of DIMMer's total savings. We note that, while the servers in our study have at most 64GB, server RAM capacities are increasing, which will increase the page migration costs incurred by DIMMer in future systems.

Migration Frequency	Energy(KWh)	Percentage over total saving
Every 5-min	1257.6	1.54%
Every 15-min	419.2	0.51%
Every 30-min	209.6	0.26%
Every 1-hour	104.8	0.13%

Table 4. Measured energy consumption of page migration, expressed as absolute energy and as the percentage of DIMMer's energy savings.

We also measured the performance penalty of page migration. It takes at most 13.5s to migrate 8GB of memory in our test system. Although the time is not trivial, it is an upper-bound. Further, it is important to note that this penalty occurs by design only at low CPU and memory utilization, when DIMMer is engaged to power off components. This is exactly the time when unused CPU and memory bandwidth are available. During high utilization, DIMMer can be designed to simply disable its allocator and migration thread.

Further, based on GCT, 30.2% of used pages are cache pages and 11.2% are cache pages not mapped into any processes. In many workloads [46], very few disk cache pages are hot; it is only useful for DIMMer to migrate hot cache pages and anonymous pages. During times of low utilization, DIMMer applies a smart page allocation policy that minimizes the total number of page migrations that will be needed before ranks can be powered down. To avoid perturbing live services, DIMMer would migrate pages in the background and at low priority [14]. If a system incurs frequent state transitions, DIMMer would likely have a negative performance impact. There also exist circumstances and load conditions that could make VMs performance suffer due to dynamic memory resizing [10]. According to the study in [31], the mean execution time of VM-based 1GB memory addition with coarse-grained hotplug is 0.43s on Dell PowerEdge1950 servers with two Intel® quad-core Xeon E5450 3GHz processors. The mean execution time of 1GB memory removal is 0.3s in a heavily-loaded VM running TPC-C [6]. If tasks run significantly longer due to DIMMer, the overall power efficiency may decrease.

[4] We suspect the functionality already exists in modern DRAM controllers, but the control registers to power off ranks are not publicly documented.

5.2 Cost of Reduced Cache Capacity

Modern OSes liberally use large amounts of idle memory as disk cache, following the mantra "free memory is wasted memory." Powering off DRAM reduces the disk cache capacity and may impact performance and energy by forcing re-read of disk contents.

Prior work suggests that cache miss rates follow "the 30% Rule" (i.e., doubling the cache size decreases the miss rate by 30% on average) [26, 39, 40]. Accordingly, cache benefits decrease with larger caches. Beyond a certain capacity that captures an active working set, disk caches do not noticeably affect system performance.

Experimental evidence with disk caches in cloud workloads support this estimate. Zhu et al. indicate that, for a web server, a large decrease in cache capacity leads to minimal changes in hit rate [46]. Sacrificing a few percent hit rate, the cache costs can be reduced by almost 90%. Although one cannot reliably infer the performance implications of reducing disk cache for other workloads, these web server results are promising.

Furthermore, prior work offers mechanisms to mitigate cache capacity concerns [45]. Cache entries can be dynamically classified as "hot" or "cold", and kept in separate ranks. As certain cache pages become hotter, and their "cold" rank host becomes a candidate to transition to low-power state, the hot cache pages can be migrated to a "hot" rank.

Finally, note that any and all performance impact of DIMMer mechanisms can be disabled on demand at high utilization and only employed when the load (memory and CPU) warrant their use.

5.3 Cost of Non-Interleaved Address Mapping

Rank-aware memory allocation can be achieved by disabling both rank and channel interleaving [7], which may degrade performance for some workloads. Channel interleaving is used to improve memory bandwidth by interleaving physical pages across DIMMs on multiple channels. Rank interleaving reduces memory latency by spreading each page across many ranks, enabling concurrent accesses by letting the controller open a row in one rank, while another rank is being accessed. However, for cloud workloads, the performance reduction from disabling interleaving is negligible.

Reduced memory bandwidth may degrade performance of some workloads. Replacing server memory with lower-bandwidth mobile DRAM results in between zero and 1.55x performance degradation of workloads such as SPEC-CPU, PARSEC, and SPEC-OMP [33]. However, most cloud workloads severely underutilize the available memory bandwidth [17, 33], even during peak times. Ferdman et al. show that the per-core off-chip bandwidth utilization of map-reduce, media streaming, web front end, and web search is at most 25% of the available bandwidth. As a result, the peak bandwidth reduction associated with disabling channel interleaving is not expected to impact the performance of cloud applications.

Similar to turning off channel interleaving, turning off rank interleaving will not incur an obvious performance reduction. For the niche of memory intensive workloads that may get impacted, VipZonE [12] shows that turning off rank interleaving results in only 1.03% execution time overhead. While this is the average overhead, it really depends on the application. A high-performance key-value store, for example, might care quite a bit about per-lookup (and thus DRAM) latency. Also, accessing data on a different NUMA node can have an unacceptable performance impact in some applications.

Further, DIMMer can be designed to reserve certain interleaving-enabled DRAM channels (e.g., on CPU socket 0, which will always remain powered on) to service memory-intensive workloads. Finally, it is also possible to retain the benefits of channel interleaving by only powering off parallel ranks across channels.

6. Related Work

A number of works address energy proportionality at server granularity. In [44], Zhang et al. dynamically change the number of active machines in the cloud to save energy by finding the best trade-off between the cost of reconfiguration and the amount of energy consumed across the entire data center. Analyzing GCT shows that this solution could have saved 18.5% to 50% of the consumed energy. [13, 29] propose power-proportional provisioning approaches for internet services. However, these approaches cannot be used if cloud servers provide background services and cannot be powered off, providing an opportunity for DIMMer to achieve energy proportionality without losing availability. Moreover, instantaneous demand spikes and the increased failure rates of frequently power-cycled components such as power supplies, disks, and fans further discourage such power-off approaches [36].

Servers reduce power consumption during times of low utilization by putting components into low-power states. CPU cores and caches use dynamic voltage and frequency scaling (DVFS) to adaptively tune the CPU frequency to reduce power consumption [22, 23]. When completely inactive, CPU cores are power gated and memory DIMMs are placed into self-refresh states. DIMMer will take this concept a step further, completely powering down entire CPUs when all cores and attached memory are unused.

Similar to the approaches that increase inactivity time on disks, a number of proposals modify the OS page allocation policy and DRAM controller logic to maximize the time DRAM ranks spend in low-power states [9, 30]. Sparsh Mittal has surveyed techniques that efficiently manage DRAM power consumption [35] (e.g., by reducing the power consumption of memory activation, memory read/write, transition among power modes, and by the utilization of low-power self-refresh mode). DIMMer would extend these approaches to allow powering off unused DRAM ranks.

Noting that data center workloads need high memory capacity, but under-utilize bandwidth, several proposals improve energy proportionality by reducing memory bandwidth. Using power-efficient DRAM reduces server energy costs [33, 42]. MemScale [16] proposes a scheme to apply DVFS to the memory controller and dynamic frequency scaling (DFS) to the memory channels and DRAM devices. Unlike DIMMer, these approaches assume that all memory contains useful data, resulting in an energy cost to refresh the entire memory capacity, even if much of it is unused.

Due to the bursty nature of data center workloads, "PowerNap" proposes introducing low-power idle states into all server components [34]. When work arrives, servers must quickly transition to an operational state, perform work, and return to the low-power idle mode. This work is complimentary to ours, as we propose identifying and powering off entirely unused cores and memory, further increasing the potential energy savings.

7. Conclusions

It is long-recognized that most server hardware exhibits disproportionately high energy consumption when operating at low utilization [41]. To mitigate this effect, low-power operating modes have been introduced. Moreover, techniques have been developed to maximize the time spent in low-power states. DIMMer builds on this work and observes that dynamic control of memory capacity presents an additional opportunity to reduce energy consumption of server systems. Using a Google cluster trace as well as in-house experiments, we estimate up to 50% savings on DRAM and 18.8% on CPU background energy. At $0.10/kWh, this corresponds to 0.6% of total data center cost.

Acknowledgments

We thank our shepherd, Dan Tsafrir, and the anonymous reviewers for their insightful comments on earlier versions of this paper. This research was supported in part by NSF grants NSF CNS-1161541, NSF CNS-1318572, NSF CNS-1223239, NSF CCF-0937833, by US ARMY award W911NF-13-1-0142, and by gifts from Northrop Grumman Corporation, Parc/Xerox, and Microsoft Research.

References

[1] EPA Emissions & Generation Resource Integrated Database (eGRID). http://www.epa.gov/cleanenergy/energy-resources/refs.html.

[2] Top Twelve Ways to Decrease the Energy Consumption of Your Data Center. Online at http://goo.gl/jYdNP4.

[3] Publications based on Google cluster trace. https://code.google.com/p/googleclusterdata/wiki/Bibliography.

[4] Kingston Memory Module Specification. http://goo.gl/FKl3Qm.

[5] Intel Performance Counter Monitor - A better way to measure CPU utilization. http://goo.gl/1tbm0V.

[6] http://www.tpc.org/tpcc.

[7] *BIOS and Kernel Developer Guide (BKDG) for AMD Family 15h Models 00h-0Fh Processors*. 2012.

[8] Hynix 1GB DDR2-SDRAM, Rev. 0.4 . http://goo.gl/teTFh6, November 2008.

[9] A. M. Amin and Z. A. Chishti. Rank-aware cache replacement and write buffering to improve dram energy efficiency. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '10)*, 2010.

[10] N. Amit, D. Tsafrir, and A. Schuster. Vswapper: A memory swapper for virtualized environments. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2014.

[11] L. A. Barroso and U. Hlzle. The case for energy-proportional computing. *IEEE Computer*, 40, 2007.

[12] L. A. D. Bathen, M. Gottscho, N. Dutt, A. Nicolau, and P. Gupta. Vipzone: Os-level memory variability-driven physical address zoning for energy savings. CODES+ISSS '12, 2012.

[13] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, 2008.

[14] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation*, 2005.

[15] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu. Memory power management via dynamic voltage/frequency scaling. ICAC, 2011.

[16] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini. Memscale: Active low-power modes for main memory. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, 2011.

[17] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Quantifying the mismatch between emerging scale-out applications and modern processors. *ACM Trans. Comput. Syst.*, 2012.

[18] J. Glanz. The Cloud Factories Power, Pollution and the Internet. The New York Times, online at http://goo.gl/WsLWni, .

[19] J. Glanz. Google Details, and Defends, Its Use of Electricity. The New York Times, online at http://goo.gl/J31LBT, .

[20] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM Computer Communication Review*, 2008.

[21] J. Hamilton. http://perspectives.mvdirona.com.

[22] J. Hopper. Reduce Linux power consumption, Part 1: Tuning results. Online at https://www.ibm.com/developerworks/library/l-cpufreq-1, 2009.

[23] J. Hopper. Reduce Linux power consumption, Part 3: Tuning results. Online at https://www.ibm.com/developerworks/library/l-cpufreq-3, 2009.

[24] H. Huang, P. Pillai, and K. G. Shin. Design and implementation of power-aware virtual memory. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, 2003.

[25] H. Huang, K. G. Shin, C. Lefurgy, and T. Keller. Improving energy efficiency by making dram less randomly accessed. ISLPED, 2005.

[26] B. Jacob, S. Ng, and D. Wang. *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007. ISBN 0123797519, 9780123797513.

[27] G. Jia, X. Li, J. Wan, L. Shi, and C. Wang. Coordinate page allocation and thread group for improving main memory power efficiency. HotPower, 2013.

[28] J. G. Koomey. My new study of data center electricity use in 2010. www.koomey.com/post/8323374335, 2011.

[29] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. H. Katz. Napsac: Design and implementation of a power-proportional web cluster. In *Proceedings of the First ACM SIGCOMM Workshop on Green Networking*, Green Networking '10, 2010.

[30] A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis. Power aware page allocation. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*, 2000.

[31] H. Liu, H. Jin, X. Liao, W. Deng, B. He, and C.-z. Xu. Hotplug or ballooning: A comparative study on dynamic memory management techniques for virtual machines. *Parallel and Distributed Systems, IEEE Transactions on*, 2014.

[32] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn. Flikker: Saving dram refresh-power through critical data partitioning. ASPLOS, 2011.

[33] K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz. Towards energy-proportional datacenter memory with mobile dram. In *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA '12)*, 2012.

[34] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: Eliminating server idle power. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XIV)*, 2009.

[35] S. Mittal. A survey of architectural techniques for dram power management. *International Journal of High Performance Systems Architecture*, 2012.

[36] S. Mueller. *Upgrading and Repairing PCs*. Pearson Education, 2011.

[37] C. Reiss, J. Wilkes, and J. L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., 2011.

[38] S. Shankland. Google uncloaks once-secret server. CNET, online at `http://news.cnet.com/8301-1001_3-10209580-92.html`.

[39] A. J. Smith. Cache memories. *ACM Comput. Surv.*, 1982.

[40] A. J. Smith. Disk cache—miss ratio analysis and design considerations. *ACM Trans. Comput. Syst.*, 1985.

[41] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the energy efficiency of a database server. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*.

[42] T. Vogelsang. Understanding the energy consumption of dynamic random access memories. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*.

[43] D. Wu, B. He, X. Tang, J. Xu, and M. Guo. Ramzzz: Rank-aware dram power management with dynamic migrations and demotions. SC, 2012.

[44] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein. Dynamic energy-aware capacity provisioning for cloud computing environments. In *The 9th International Conference on Autonomic Computing*, 2012.

[45] P. Zhou, V. Pandey, J. Sundaresan, A. Raghuraman, Y. Zhou, and S. Kumar. Dynamic tracking of page miss ratio curve for memory management. In *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2004.

[46] T. Zhu, A. Gandhi, M. Harchol-Balter, and M. A. Kozuch. Saving cash by using less cache. HotCloud, 2012.