DIMMer: A case of 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. However, even during times of low utilization, servers cannot be powered off because of unpredictable spikes in instantaneous demand on some of the resources (e.g., file storage). To mitigate energy waste, unused processor cores transition into low-power sleep states, power-gating non-critical components; memory ranks (physical subdivisions of capacity) transition into self-refresh modes, where energy is spent only to maintain memory contents.

We propose DIMMer, an approach to entirely eliminate the idle power consumption of unused system components. DIMMer is motivated by two key observations. First, that even in their lowest-power states, the power consumption of today's server components remains significant. Second, that unused components can be powered off entirely without sacrificing availability. We demonstrate that unused memory capacity can be turned 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. Leaving a single CPU powered on allows the server to remain fully operational and capable of rapidly ramping up to peak capacity if needed. In this work, we demonstrate the potential for DIMMer to improve energy proportionality and achieve energy savings. Using power measurements of a modern server system and data from a Google data center, we show up to 50% savings on DRAM and 18.8% on CPU background energy.

Acknowledgement: 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.

Submission to SoCC '14, October, 2014, Seattle, WA, USA.

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 is more than 200B KWh, between 1.1% and 1.5% of worldwide electricity use [20]. Three years ago, Google announced that their facilities have a continuous electricity usage equivalent to powering 200,000 homes [11].

Surprisingly, despite energy being one of the top three data center operating costs [13], 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 [25] during times of low utilization), servers are kept operating at full capacity. Across data centers, hundreds of thousands of servers remain idle or underutilized in anticipation of spontaneous demand spikes [6]. As a result, a report by New York Times found energy waste upwards of 90% as the facilities are operated at full capacity regardless of the demand [12].

Industry has a number of energy saving principles and mechanisms, such as consolidation, virtualization (for increased utilization), decommissioning of unused servers, and the purchase of energy-efficient hardware [3]. Such mechanisms show promise and research demonstrates that job consolidation and server power-off strategies can result in up to 50% savings [34]. 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 [27].

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 turning 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 turns 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 [9, 10], or maximizes the time that DRAMs spend in low-power modes [17, 22, 33] (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 reducing the memory *space* available to the system can yield even 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 even when the DRAM is in self-refresh mode, the power consumption of a 4GB DIMM is approximately 1W (20% of the precharge-standby power [9]). Furthermore, because self-refresh power is proportional to DRAM capacity, the savings of DIMMer are likely to be even higher in future systems. In this paper, we use publicly available traces from production Google data centers to demonstrate the effectiveness of DIMMer.

A further benefit of DIMMer over switching DRAM to self-refresh mode [33] is that freeing DIMM contents and turning 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. Although DIMMer requires disabling DRAM channel interleaving, migrating memory pages among DRAM ranks, and reducing the memory available for disk cache, we find that these requirements do not impact the effectiveness of DIMMer. Moreover, while DIMMer requires modification to the OS kernel and hardware, there are no modifications to the applications [7, 23]. Finally, many of DIMMer's prerequisites are already implemented in prior work [7, 19, 23, 33].

Applying DIMMer to Google cluster traces [28] demonstrates that background DRAM and CPU energy consumption can be reduced by up to 50% and 18.8%, respectively.

2. Motivation for Powering off Components

2.1 DRAM Power Consumption

Figure 1 depicts our measurements of typical power consumption of active DIMMs under nominal use in a server rack. 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. Power consumption was measured with increasing throughput up to 2GB/s, a reasonable upper bound for modern data center workloads [24]. 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.

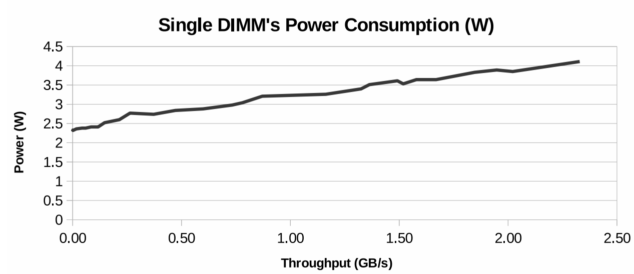


Figure 1. 8GB server DIMM power consumption as a function of throughput. Power at near-zero throughput is more than half of the power at peak utilization.

2.2 CPU Power Consumption

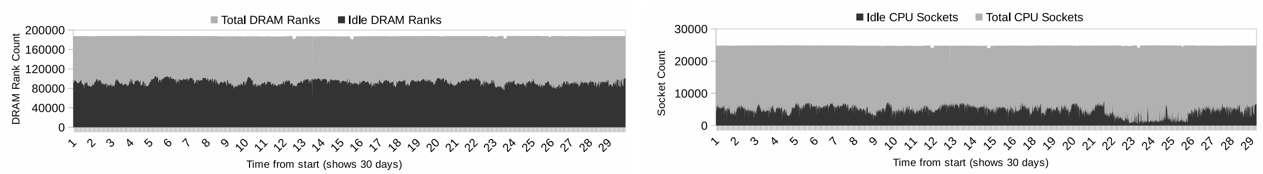
Our experience shows that individual core power consumption varies significantly across different CPUs even within the same server and is difficult to measure precisely and consistently (it varies with the number of per-node DIMMs, channels, utilization, etc.). For the purpose of this evaluation, we chose to be conservative and consider power consumption measured across CPUs rather than individual cores. The measured power consumptions of our test system while completely idle are as follows: for 1 CPU (Intel(R) Xeon(R) 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.

3. DIMMer Benefits

In an ideal world, servers can be turned 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 turning servers off and back on are high latency and high cost operations. Additionally, often (as in the case of the Google dataset [28] discussed below) individual servers may participate in distributed services (e.g., file serving) that preclude turning 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 [6].

In this paper, we show that DIMMer, which dynamically provisions 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 trade-offs and



(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. DRAM capacities, rank, CPU, and server counts are shown in Table 1.

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.

3.1 Idle Resources in the Cloud

To justify the overheads and latencies incurred by DIMMer, we must establish that real clouds feature numerous- and long-enough (per-server, per-rank) idle periods.

Google Cluster Traces. We analyzed a cluster usage dataset released by Google [28]. 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 [2].

We first compute the DRAM and CPU utilization 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. Figure 2(a) illustrates the idle DRAM results. We find that up to 50% of the cluster DRAM ranks are unused and can be powered off.

Figure 2(b) illustrates the number of CPUs that can be turned off in each 5-minute time slot. Unlike DRAM ranks that can be turned off independently, a CPU can be powered off only when all of its cores *and* all DRAM ranks attached to its socket are idle. In the existing 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 Google traces, showing DIMMer's effectiveness with the existing job placement policies.

Figure 3 shows that DIMMer can save 30MWh (DRAM) and 52MWh (CPU) for this cluster running for one month.[1] Using the cost model from [14], we estimate the corresponding cost saving over the Total Cost of Ownership (TCO),

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 [29]. Machines with unusual memory capacities (likely due to partial memory failures) were ignored.

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), respectively.[2] 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.

3.2 Power-off vs. Self-refresh

Prior work proposed to maximize the time DRAM ranks spend in low-power self-refresh mode [17, 22, 33]. Although these techniques effectively reduce DRAM power consumption, as shown in section 2, the background power of an 8GB DIMM in self-refresh mode is actually higher than the savings achieved by self-refresh compared to the peak power consumption of DRAM for a typical cloud workload.

Table 2 shows that DIMMer can significantly reduce wasted power by turning off idle memory and CPUs. For

[1] Using DRAM power estimates from [9].

[2] We assume $0.1/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 [33] (2 idle ranks)	Node 1	100	100	100	100	16	0	21.44	66.00
	Node 2	100	100	0	0	16	1.84	10.72	
DIMMer (2 idle ranks)	Node 1	100	100	100	100	16	0	21.44	64.16
	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 [33]'s approach). Using DIMMer, if we instead turn 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 [33] (1 idle node)	Node 1	100	100	100	100	16	0	21.44	57.12
	Node 2	0	0	0	0	16	3.68	0	
DIMMer (1 idle node)	Node 1	100	100	100	100	16	0	21.44	37.44
	Node 2	0	0	0	0	0	0	0	

Table 3. The main opportunity arises when turning 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 turned off completely, the entire CPU socket can be turned off, resulting in an additional saving of 35% of the total **background** power consumption.

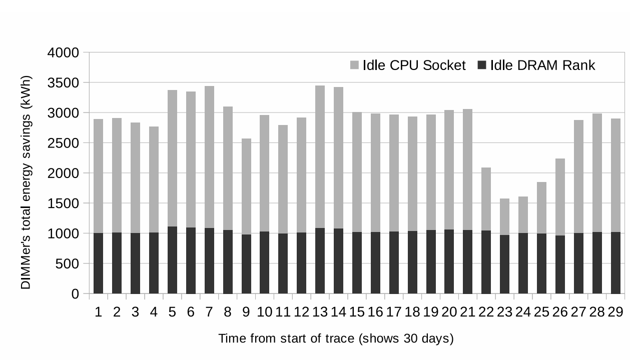


Figure 3. Energy saved by turning off the idle DRAM ranks and CPUs, respectively.

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 when using techniques that arrange memory into "cold" and "hot" ranks, saving energy on "cold" ranks by putting them into self-refresh mode [33]. If we turn off "cold" ranks entirely, DIMMer can save additionally 3% of the DRAM power consumption.

Furthermore, the main opportunity for power savings arises when all of the ranks attached to a CPU can be turned 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 can further reduce background power consumption by turning 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.

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 [7, 19]. The difference in page allocation between DIMMer and standard Linux lies in the total number of free lists. Besides an allocator, 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 [7, 23], no libraries or application would need to be modified.

For reliable deployment, DIMMer *may* also requires hardware changes. Flikker [23] 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.[3] Support for full electrical power-off of CPU sockets may also be helpful.

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

5. DIMMer Costs

5.1 Cost of Page Migration

Before powering off unused ranks, DIMMer migrates 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 (4 DRAM ranks) from one NUMA node to another. The node-to-node migration is the most expensive migration that would happen in DIMMer.

The average measured energy cost to migrate a single page is 102 μJ. As Table 4 shows, if DIMMer migrates 8GB every 30 minutes, the additional monthly energy cost of page migration for the cluster is approximately 210KWh, a mere 0.26% of DIMMer's total savings.

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 Google cluster traces, 30.2% of used pages are cache pages and 11.2% are cache pages not mapped into any processes. In many workloads [36], 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 migrates pages in the background and at low priority.

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." Turning off DRAM reduces the disk cache capacity and may impact performance and energy by forcing re-read of disk contents.

To estimate the impact of reduced cache capacity, [18, 30, 31] suggest that cache miss rates follows "the 30% Rule" (i.e., doubling the cache size decreases the miss rate by 30% on average). Accordingly, hit cache benefits decrease with larger cache size. 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 [36]. Sacrificing a few percent hit rate, the cache costs can be reduced by almost 90%, without any noticeable effects on users' experience.

Furthermore, prior work offers mechanisms to mitigate cache capacity concerns [35]. 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 [4], 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.

The main insight comes from the fact that most cloud workloads severely underutilize the available memory bandwidth [24], even during peak times. Ferdman et al. show that the per-core off-chip bandwidth utilization of MapReduce, 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 will not 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 [7] shows that turning off rank interleaving results in only 1.03% execution time overhead.

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 turning off parallel ranks across channels.

6. Related Work

A number of works address energy proportionality at server granularity. In [34], Zhang et al. dynamically change the number of active machines in the cloud to save energy by solving an optimization problem, finding the best trade-off between the cost of reconfiguration and the amount of energy consumed across the entire data center. Analyzing Google cluster traces shows that this solution could have saved 18.5% to 50% of the consumed energy. Krioukov et al. [21] propose NapSAC, a power-proportional web cluster. NapSAC provisions the number of machines needed for the current workload and the observed response latency. Chen et al. [8] also proposed an energy-aware server provisioning approach for internet services by adaptively changing the number of powered on servers. Unlike NapSAC, which concentrates on workloads with short-lived requests, Chen et al. [8] handled workloads with long-lived connections (e.g., Windows Live Messenger). 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 [27].

Servers reduce power consumption during times of low utilization by putting components into low-power states. CPU cores and private caches use dynamic voltage and frequency scaling (DVFS) to adaptively tune the CPU frequency to reduce its power consumption [15, 16]. When completely inactive, CPU cores are power gated and memory DIMMs are placed into self-refresh states. DIMMer takes this concept a step further, proposing to completely power down entire CPUs when all cores and attached memories 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 that DRAM ranks spend in low-power states [5, 22]. Sparsh Mittal has surveyed several techniques that efficiently manage DRAM power consumption [26] (e.g., by reducing the power consumption of memory activation, memory read/write, transition among different power modes, and by the utilization of low-power self-refresh mode). DIMMer extends these approaches to allow powering off unused DRAM ranks.

Noting that data center workloads need high memory capacity, but under-utilize bandwidth, a number of approaches improve memory energy proportionality by reducing performance. Using power-efficient mobile DRAM devices reduces server energy costs [24]. MemScale [10] proposes a scheme to apply DVFS to the memory controller and dynamic frequency scaling (DFS) to the memory channels and DRAM devices. Unlike our work, these approaches assume that all memory capacity 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 [25]. 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 [32]. 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. Just as servers experience times of low CPU utilization, they also experience times of low memory capacity demand. While prior work has concentrated on reducing the *time* that components such as CPU and memory spend in high-power modes, we find that reducing the memory *space* available to the system can yield even greater benefits.

By dynamically reducing the memory capacity available to the OS, we are able to consolidate unused memory on DRAM ranks that can be *entirely powered off* rather than simply being placed into a low-power state. Controlling capacity in this way enables true power proportionality for the memory system, where the energy costs for memory capacity are paid only when this capacity is requested by the system. Moreover, turning off unused memory capacity enables even greater energy savings on CPUs, as unused CPUs can be powered off when all memory capacity attached to their sockets is unused. DIMMer allows the server to remain fully operational and capable of rapidly ramping up to peak capacity if needed, but saves significant energy by entirely powering off components rather than placing them into a low-power state. Using publicly available Google cluster traces and power measurements on a modern server system, we demonstrated that applying the DIMMer approach can reduce DRAM and CPU background energy consumption up to 50% and 18.8%, respectively. This corresponds to a U.S. annual non-baseload CO2 output emission reduction of over 51 metric tons of CO2.

References

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

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

[3] Top Twelve Ways to Decrease the Energy Consumption of Your Data Center. Online at http://www.energystar.gov/index.cfm?c=power_mgt.datacenter_efficiency.

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

[5] Ahmed M. Amin and Zeshan 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.

[6] Luiz Andr Barroso and Urs Hlzle. The case for energy-proportional computing. *IEEE Computer*, 40, 2007.

[7] Luis Angel D. Bathen, Mark Gottscho, Nikil Dutt, Alex Nicolau, and Puneet Gupta. Vipzone: Os-level memory variability-driven physical address zoning for energy savings. CODES+ISSS '12, 2012.

[8] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng 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.

[9] Howard David, Chris Fallin, Eugene Gorbatov, Ulf R. Hanebutte, and Onur Mutlu. Memory power management via dynamic voltage/frequency scaling. ICAC, 2011.

[10] Qingyuan Deng, David Meisner, Luiz Ramos, Thomas F. Wenisch, and Ricardo 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.

[11] James Glanz. Google Details, and Defends, Its Use of Electricity. The New York Times, online at http://www.nytimes.com/2011/09/09/technology/google-details-and-defends-its-use-of-electricity.html?_r=0.

[12] James Glanz. The Cloud Factories Power, Pollution and the Internet. The New York Times, online at http://www.nytimes.com/2012/09/23/technology/data-centers-.waste-vast-amounts-of-energy-belying-industry-image.html.

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

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

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

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

[17] Hai Huang, Kang G. Shin, Charles Lefurgy, and Tom Keller. Improving energy efficiency by making dram less randomly accessed. ISLPED, 2005.

[18] Bruce Jacob, Spencer Ng, and David Wang. *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[19] Gangyong Jia, Xi Li, Jian Wan, Liang Shi, and Chao Wang. Coordinate page allocation and thread group for improving main memory power efficiency. HotPower, 2013.

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

[21] Andrew Krioukov, Prashanth Mohan, Sara Alspaugh, Laura Keys, David Culler, and Randy 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.

[22] Alvin R. Lebeck, Xiaobo Fan, Heng Zeng, and Carla Ellis. Power aware page allocation. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*, 2000.

[23] Song Liu, Karthik Pattabiraman, Thomas Moscibroda, and Benjamin G. Zorn. Flikker: Saving dram refresh-power through critical data partitioning. ASPLOS, 2011.

[24] Krishna T. Malladi, Benjamin C. Lee, Frank A. Nothaft, Christos Kozyrakis, Karthika Periyathambi, and Mark Horowitz. Towards energy-proportional datacenter memory with mobile dram. In *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA '12)*, 2012.

[25] David Meisner, Brian T. Gold, and Thomas 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.

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

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

[28] Charles Reiss, John Wilkes, and Joseph L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., 2011.

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

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

[31] Alan Jay Smith. Cache memories. *ACM Comput. Surv.*, 1982.

[32] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A. Shah. Analyzing the energy efficiency of a database server. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 231–242, New York, NY, USA, 2010. ACM.

[33] Donghong Wu, Bingsheng He, Xueyan Tang, Jianliang Xu, and Minyi Guo. Ramzzz: Rank-aware dram power management with dynamic migrations and demotions. SC, 2012.

[34] Qi Zhang, Mohamed Faten Zhani, Shuo Zhang, Quanyan Zhu, Raouf Boutaba, and Joseph L. Hellerstein. Dynamic energy-

aware capacity provisioning for cloud computing environments. In *The 9th International Conference on Autonomic Computing*, ICAC '12, 2012.

[35] Pin Zhou, Vivek Pandey, Jagadeesan Sundaresan, Anand Raghuraman, Yuanyuan Zhou, and Sanjeev 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.

[36] Timothy Zhu, Anshul Gandhi, Mor Harchol-Balter, and Michael A. Kozuch. Saving cash by using less cache. HotCloud, 2012.