DR. Swap: Energy-Efficient Paging for Smartphones

Kan Zhong Xiao Zhu Tianzheng Wang[†] Dan Zhang
Xianlu Luo Duo Liu[*] Weichen Liu Edwin H.-M. Sha

College of Computer Science, Chongqing University, liuduo@cqu.edu.cn
Key Lab. of Dependable Service Computing in Cyber Physical Society (Chongqing Univ.), Ministry of Education
[†]Department of Computer Science, University of Toronto, tzwang@cs.toronto.edu

ABSTRACT

Smartphones are becoming increasingly energy-hungry to support feature-rich applications, posing a lot of pressure on battery lifetime and making energy consumption a non-negligible issue. In particular, DRAM is among the most demanding components in energy consumption. In this paper, we propose DR. Swap, an energy-efficient paging design to reduce energy consumption in smartphones. We adopt emerging energy-efficient non-volatile memory (NVM) and use it as the swap area. Utilizing NVM's byte-addressability, we propose direct read which guarantees zero-copy for read-only pages in the swap area. Experimental results based on the Google Nexus 5 smartphone show that our technique can effectively reduce energy consumption.

Categories and Subject Descriptors

D.4.2 [**Operating Systems**]: Storage Management—*Main memory, storage hierarchies*

Keywords

Swapping; paging; energy; non-volatile memory; smartphone

1. INTRODUCTION

Thanks to the advances in mobile microprocessors and operating systems, smartphones nowadays integrate more functionality than they ever had, such as the ability to install third-party applications, multi-tasking and gaming. These functionalities, on the one hand bring great user experiences; on the other hand they accelerate the depletion of the limited energy that could be carried by a smartphone in the form of batteries with a capacity of around 1000–2000mAh. Such resource-constrained nature of smartphones in turn affects user experience. For example, in most smartphone OSes, applications are not terminated (thus resources not released) when they are switched to backend to allow faster switch-back. Various daemons also keep running all the time to pull useful information for the user (e.g., notifications for new instant messages). As a result, a lot of energy is consumed by the DRAM-based main memory to maintain these runtime data, leading to high energy consumption.

[*]Duo Liu is the corresponding author.

Table 1: Comparing PCM, DRAM and NAND flash [5, 21].

Attributes	DRAM	PCM	NAND
Non-volatility	No	Yes	Yes
Idle power	~100mW/GB	~1mW/GB	~10mW/GB
Bandwidth	~GB/s	50-100MB/s	5-40MB/s
Write latency	20-50ns	~1us	~500us
Erase cycles	∞	$10^6 - 10^7$	$10^4 - 10^5$

What makes the situation worse is the trend of adopting large main memories to support feature-rich applications. For example, Google Nexus 5 has as much as 2GB main memory.[1] Larger main memory improves system performance, but inevitably leads to higher energy consumption [4, 18]. It is reported that smartphone's main memory can consume more than 30% of the overall energy [4]. Reducing the energy consumption of main memory becomes critical in smartphones. Most existing work [8, 13] suggests turning off inactive DRAM banks or reducing memory usage. However, these approaches may degrade performance as they essentially reduce usable system memory.

We argue that smartphones should re-adopt swapping with the help of emerging byte-addressable, non-volatile memory (NVM). Swapping is an effective way of extending memory using storage spaces [17]. It has long been a standard feature in modern OSes, but smartphones seldom use it because of the sub-optimal storage (NAND flash) performance. Though flash memory has much better energy consumption parameters than DRAM, it could not be used as the swap area while maintaining acceptable performance. Compared to flash, byte-addressable NVMs such as phase change memory (PCM) [26] and memristor [23] offer not only faster (near-DRAM) performance, but also lower energy consumption. As shown in Table 1, PCM exhibits much better energy parameters when compared to both DRAM and NAND flash. It also exhibits much shorter write latency when compared to NAND flash [27]. A plethora amount of work have been proposed to further achieve near-DRAM performance and better endurance for PCM [2, 7, 9, 12, 16, 19, 28]. Other NVMs such as STT-RAM [6] could promise even faster performance and better endurance than DRAM [10]. Thus, we do not specifically consider endurance or latency issues

[1]`http://www.google.com/nexus/5`

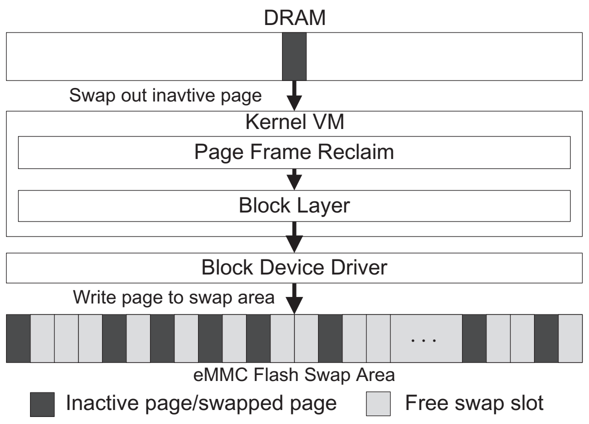

Figure 1: Traditional NAND flash backed swapping.

and focus on energy consumption in this paper. Unlike flash, these NVMs are byte-addressable and can be placed on the memory bus, available to `load` and `store` instructions. Such combination of high performance and low energy consumption makes NVM an ideal candidate for swapping.

In this paper, we propose an in-memory paging architecture called *DR. Swap*, to re-adopt swapping in smartphones by replacing part of the DRAM with NVM, and using NVM as a swap area. With less DRAM, we reduce energy consumption, while the NVM based swap area extends memory capacity to still allow feature-rich applications to run. In addition, utilizing the NVM's byte-addressability, we allow direct read (DR) for read requests directly from the swap area, guaranteeing zero-copy for read-only pages. With DR, read requests are satisfied by mapping the virtual address to the physical page in the NVM-based swap area, instead of by copying the memory page from the swap area to user space. DR is made possible because of the byte-addressability of NVM. In DR. Swap, the NVM-based swap area is attached to the memory bus, eliminating I/O and the whole storage stack overhead. With the traditional swap approach which has to go through the whole storage stack to access a page, we avoid unnecessary memory copying to DRAM, thus reducing energy consumption.

In summary, we make the following contributions:

- We explore the feasibility of re-adopting swapping for better energy consumption behaviors in smartphones;
- Based on byte-addressable NVMs, we propose an in-memory paging architecture to reduce energy consumption while maintaining high performance;
- We propose direct read (DR) to further avoid unnecessary memory copying induced by read-only requests, thus reducing energy consumption.

In Section 2 we first give related backgrounds. Section 3 details the design of DR. Swap. Evaluation results are shown in Section 4. We summarize related work and conclude in Sections 5 and 6, respectively.

2. BACKGROUND

We give background on swapping and energy-related issues in smartphones. We use Google Android as an example as it is the most widely adopted smartphone OS. Note that this work can also be extended to support other platforms.

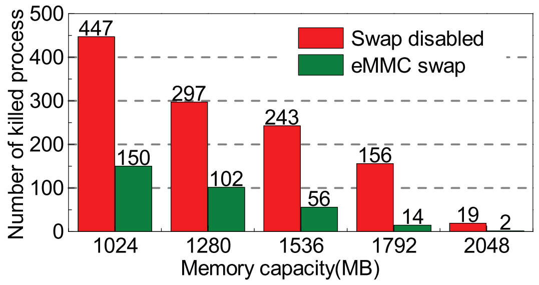

Figure 2: The number of processes killed with and without a swap area under different DRAM sizes.

2.1 Swapping and Paging in Smartphones

Swapping is an effective way to extend memory space by borrowing space backed by storage devices (e.g., NAND flash) in modern OSes [17]. With paging, swapping becomes more flexible as processes could be swapped in and out in units of non-contiguous pages. However, usually swapping is not enabled by smartphones due to the sub-optimal storage (NAND flash) performance [11]. As shown in Figure 1, a traditional swap area is backed by storage, such as flash memory. The swap area is divided into *slots*, each of which is precisely the size of a page. As shown in Figure 1, when memory is under pressure, the kernel will start to swap inactive pages out to the I/O device via the block layer to make room for incoming memory allocations.

To avoid poor performance, mainstream mobile OSes such Android disables swapping and implements a low memory killer (LMK) to reclaim memory by terminating certain processes when the system is under pressure. Despite the poor performance, we find that a swap area can significantly reduce the number of killed processes and improve user experience should we have high performance storage. We plot the number of killed processes by LMK (y-axis) with varying memory capacity (x-axis) in Figure 2. With a flash backed swap area, the number of killed processes could be significantly reduced (e.g., from 447 to 150 with 1G memory). Figure 3 highlights the amount of block I/O induced by a flash backed swap. Swapping greatly increases I/O operations. Recent research has shown that storage plays a significant role in application performance [11]. In particular, when pages are swapped out from main memory to the on-board eMMC flash, a significant portion of bandwidth is occupied, leading to sub-optimal overall performance. Moreover, the erase count of eMMC flash is limited to 10^5 [3]; frequently writing to the swap area further reduces the lifetime of NAND flash.

2.2 Energy Consumption in Smartphones

Due to size, weight and heat dissipation constraints, smartphones nowadays usually can only be equipped with batteries of very limited capacity (e.g., 1000–2000mAh). This implies that energy becomes a first-class citizen in smartphones. In particular, the energy consumed by DRAM is non-negligible [1]. DRAM could account for as much as 34.5% of the overall energy consumption of a smartphone [20]. What makes the situation worse is the trend of adopting large main memories to support feature-rich applications. For example, the Google Nexus 5 smartphone has as much as 2 GB memory. In most smartphone OSes, applications are not fully closed (thus resources not released) when they

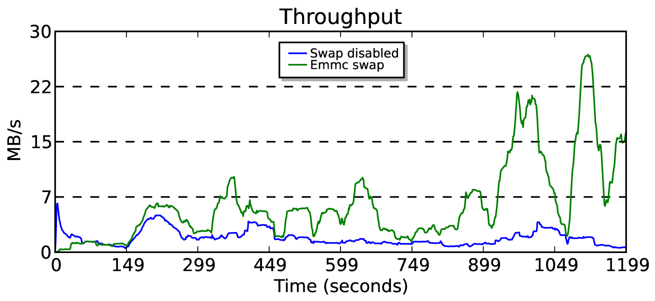

Figure 3: Comparison I/O between eMMC swap and native Android OS with swap disabled.

are switched to backend to allow faster switch-back. Various daemons also keep running all the time to pull useful information for the user (e.g., notifications for new instant messages). As a result, excessive energy is consumed by the DRAM-based main memory to maintain these run-time data, leading to high energy consumption.

3. ENERGY-EFFICIENT PAGING

We first give details on DR. Swap, which consists of our energy-efficient in-memory paging (IMP) architecture and the direct read optimization. In the end, we discuss IMP's energy efficiency.

3.1 In-Memory Paging Architecture

Utilizing NVM's energy-efficiency and byte-addressability, DR. Swap consists of an in-memory paging architecture and the direct read (DR) optimization. Our in-memory paging architecture attaches NVM to the memory bus, side by side with DRAM to make it directly accessible by the `load` and `store` instructions. Different from hybrid memory approaches which treat NVM as part of main memory, we dedicate the NVM region as the swap area, which is usually backed by some I/O device (e.g., NAND flash) in existing systems. Compared to hybrid memories, swapping effectively re-uses the infrastructure that is already existed in mobile OSes and much less intrusive to implement. With IMP, swapping requests become pure memory copying, instead of I/O requests, thus eliminating the need to go through all the storage stack to access data in the swap area, and allowing better utilization of NVM's high performance.

Since we attach NVM to the memory bus, the OS sees an NVM area that shares part of the physical address space with DRAM. We focus on the software side in this paper, but expect the memory controller to provide information on which part of the whole address space belongs to NVM (e.g., through the E820 table in the x86 architecture). The OS can then manage the NVM area by reading such information at boot time. We still use DRAM as main memory and eMMC flash as secondary storage for system and user data. On top of the OS kernel, all system libraries and user space applications work as usual.

Figure 4 shows the details of adopting IMP in existing OSes. When the system is under pressure (i.e., no enough memory for satisfying allocation requests), the memory management subsystem will try to reclaim page frames from running applications and swap them out to the swap area. We replace the traditional swap subsystem with our NVM-based swap subsystem, which accesses NVM directly without going

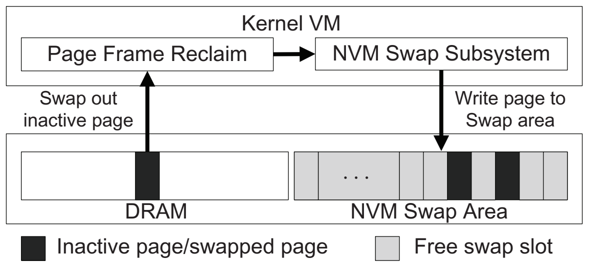

Figure 4: In-memory paging architecture. We replace the traditional storage-based swap area with memory-attached NVM. The memory management subsystem interacts with memory, instead of I/O devices (e.g., flash) to swap in/out pages.

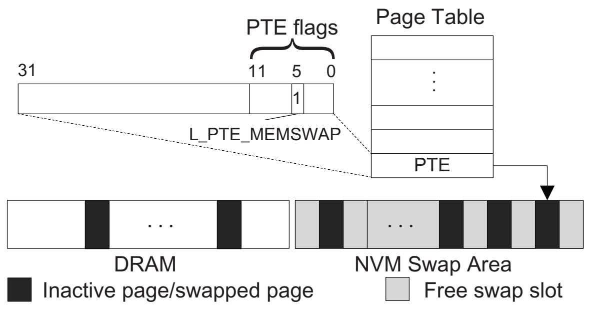

Figure 5: Overview of direct read. DR directly maps the NVM page to the user space from the swap area, saving extra memory copying for page reads.

through the storage stack. Victim pages selected by the kernel's page frame reclaim routine are directly written to the swap area through simple `memcpy` calls. Compared to NAND flash, though NVM could have similar read/write power, it exhibits lower idle power and much faster read/write speed than NAND flash. Compare to the traditional I/O based swap architecture shown in Figure 1, IMP achieves both high performance and energy efficiency.

3.2 Direct Read

In a traditional paging system based on I/O devices (e.g., NAND flash), victim memory pages will be copied first to the swap area and then copied back to main memory (i.e., DRAM) when the page is requested again from the user space. The kernel handles such requests through the page fault handler, which reads the I/O device to fetch the requested page, set up new page table mappings and return to the user application. The whole operation will involve at least one I/O device read, one DRAM page write and one page table entry (PTE) write. It fits nicely with its target architecture. In the IMP architecture, the whole operation now will involve one memory read, one memory write and one PTE write. However, this approach incurs unnecessary memory copying, especially for page reads, since the requested memory page already resides in memory – the NVM – though in a different region.

To remove unnecessary memory operations between NVM and DRAM, as shown in Figure 5, DR directly sets up the PTE mappings from the user space virtual address to the physical address of the NVM page in the swap area, instead of first reading and then copying the page from NVM to

DRAM. In this way, we remove the need of both reading and writing of NVM and DRAM, respectively. Compared to the traditional approach, we save the energy for reading and writing a whole page. The only overhead left is for the PTE write, which only involves writing a 32-bit entry.

DR naturally utilizes the fast read performance of most NVM technologies. We do not allow "direct write" for write requests, due to the asymmetric nature of most NVM's latency (e.g., PCM has much faster reads than writes), wear leveling concerns, and the complication brought by intrusive changes to support writes.

3.3 Energy Model

We analyze the energy efficiency of different paging architectures with the following model. Table 2 lists the power consumption mnemonics and values of read, write, idle and refresh operations for DRAM, NVM, and NAND flash. Energy parameters used in our model are shown in Table 3. We compare the energy consumed by DR. Swap, NAND flash backed swap and `ramdisk` (DRAM-backed) swap. By comparing the energy consumption of DR. Swap with `ramdisk` swap, we show how much energy is saved by our IMP architecture and the DR optimization.

Note that idle power is not included in our model for simplicity. The idle power of different memory technologies have obvious larger-than relationships, and will only add a constant to each pair of comparison result. Thus, the omission will not affect the accuracy of our model when comparing different architectures.

DR. Swap. Swapping pages out involves copying them from DRAM to NVM and then setting up PTEs to indicate that these pages are not present in memory (see details in Section 2). Thus, the energy consumed by W swap-outs is:

$$E_{WN} = W \times S \times (P_{RD} + P_{WN}) + W \times P \times (P_{RD} + P_{WD}) \quad (1)$$

For each page read, no memory copy is required with DR. The only overheads are one NVM read for the processor to access the page in NVM, the PTE write to setup new memory mapping and one DRAM read to read the mapping by the MMU. The energy consumed by R swap-ins is:

$$E_{RN} = R \times S \times P_{RN} + R \times P \times (P_{WD} + P_{RD}) \quad (2)$$

NAND flash backed swap. Swap-out is similar to DR. Swap, except that accessing NAND flash is slower:

$$E_{WF} = W \times S \times (P_{RD} + P_{WF}) + W \times P \times (P_{RD} + P_{WD}) \quad (3)$$

Direct read cannot be applied in this case, thus the energy consumed by R swap-ins is:

$$E_{RF} = R \times S \times (P_{RF} + P_{WD}) + R \times P \times (P_{WD} + P_{RD}) \quad (4)$$

DRAM backed swap. The only difference with NAND flash backed swap is the latency values. For swap-outs:

$$E_{WD} = W \times S \times (P_{RD} + P_{WD}) + W \times P \times (P_{RD} + P_{WD}) \quad (5)$$

Swap-in is similar, with DRAM's latency:

$$E_{RD} = R \times S \times (P_{RD} + P_{WD}) + R \times P \times (P_{WD} + P_{RD}) \quad (6)$$

Comparison. Based on Equations (2) and (4), the difference on swap-ins with DR. Swap and NAND flash is:

$$D = R \times S \times (P_{RF} + P_{WD} - P_{RN}) \quad (7)$$

Similarly, we can derive the difference between DR. Swap and DRAM swap by replacing P_{RF} in Equation (7) with P_{RD}. As we expect that P_{RN} is smaller than both P_{RF} and P_{RD}. When taking idle power and refresh power(for DRAM) into consideration, we conclude that DR. Swap is capable of reducing energy consumption when compared to existing swap architectures. In Section 4 we verify these projections by running various smartphone applications.

Table 2: Power consumption mnemonics and values for DRAM, NVM and NAND flash in our energy model. Values are shown in the corresponding parentheses.

	Read (nJ/Byte)	Write (nJ/Byte)	Idle (mW/GB)	Refresh (mW/GB)
DRAM	P_{RD} (0.8)	P_{WD} (0.8)	P_{ID} (100)	P_{FD} (1.35)
NVM	P_{RN} (0.8)	P_{WN} (8)	P_{IN} (1)	P_{FN} (0)
eMMC	P_{RF} (1)	P_{WF} (1.3)	P_{IF} (10)	P_{FF} (0)

Table 3: Energy model parameters.

Parameter	Explanation
R	Number of swap-ins
W	Number of swap-outs
S	Page size, default 4KB
P	PTE length, default 4 bytes

4. EVALUATION

We implement and evaluate DR. Swap based on Google Nexus 5 with Android 4.4 (Linux kernel version 3.4). The Nexus 5 smartphone features a Qualcomm Snapdragon 8974 processor clocked at 2.3GHz, 2GB DRAM and 16GB eMMC NAND flash. We focus on energy issues in this paper, though it is obvious that a fast NVM-based swap area will definitely improve performance. We left performance evaluation as future work. The rest of this section first gives our experimental setup. We then present and discuss the results.

4.1 Experimental Setup

We use the Android Debug Bridge (ADB) provided by the Android SDK to communicate with the Nexus 5 smartphone which is connected to a desktop PC. To understand the effect of eMMC flash backed swap, we use blktrace to collect block layer I/O activities. Before each test, we reboot the phone and set aside for few minutes to ensure the device is roughly in the same state (e.g. number of background process). For all the experiments, we connect the phone to a charger and make sure the phone is working in its full performance capability.

We use PCM as the NVM in our experiments. Note that we do not specifically emulate the latency values of PCM, though it is slower than DRAM. As we mentioned in Section 1, a lot of work on improving the endurance and performance of PCM has been proposed. In our future work, we will study how to improve the endurance for NVM backed swap area. Moreover, our system does not rely on any specific type of NVM and can be easily adopted by different NVM-based systems. In our work, we do not focus on which NVM can be served as the swap area, and instead, we focus on the how to design and implement the NVM backed swap area. The energy consumption values for DRAM, PCM, and eMMC flash is shown in Table 2. Table 4 lists the applications we used in the experiments. We classify them into seven categories, including browser, social network, multimedia, office,

Table 4: Workload applications.

Category	Application
Internet	Android Browser, Firefox Browser for Android, Google Chrome, Opera
Social networking	Facebook, Google+, Pinterest, QQ, Sina Weibo, Skype, Twitter, WhatsApp
Multimedia	Google Play Music, MX Player, TTpod Player, Youtube
Office	Evernote, Gmail, Google Drive, Google Maps, Office Mobile
Gaming	Angry Brid, Asphalt 8 , Temple Run 2
Shopping	Amazon, Ebay, Fancy, Google Play, TaoBao
News	BBC News, Engadget, Flipboard, Google Newsstand, NBC News, NetEase News, Netflix, TED, Zaker

games, shopping and news. Those categories have covered the applications we daily used and the application we choice are worldwide popularity. Therefore we use those applications to evaluate our work. By running these applications, we collect data for the following metrics:

Memory copy reduction. To evaluate the effectiveness of direct read, we run all the applications in each category for 15 minutes shown in Table 4 and count the number of reduced memory copy. Accessing a non-present page in DRAM could cause a page fault, we modified the page fault handler to support read a swapped out page in NVM swap area directly. A read counter is used to counts the number of reduced memory copy and a write counter is used to counts the actually swap-ins.

Energy consumption. To evaluate the energy consumption of DR.Swap, we run all the applications in each category to compare the energy consumption under different swap implementations including DRAM backed, eMMC flash backed and DR.Swap. The energy consumption by each swap implementations is computed using the model proposed in section 3 according to the swap-ins and swap-ous.

4.2 Results

Figure 6 compares the number of direct reads and the "real" swap-ins. With an eMMC flash backed swap architecture, the total number of swap-ins can be count as the sum of DR. Swap's direct reads and DR. Swap's "real" swap-ins. With DR. Swap, we reduce the number of required memory copy by around 50% for browsers, office and gaming applications. Moreover, for other applications such as shopping, DR. Swap reduces more than 70% "real" swap-ins.

Figure 7 shows that DR. Swap consumes much less energy when compared to eMMC flash backed swap and DRAM backed swap architectures. The paging architecture we proposed is energy efficient. The energy consumption is computed under the energy model proposed in Section 3. The total energy consists of three parts: swap-in/swap-out energy, idle energy and refresh energy. For eMMC flash swap and DR. Swap, the refresh energy is zero as neither PCM nor eMMC flash requires constant voltage to maintain its data. Because of the limited swap-ins and swap-outs during the 15 minutes, the energy is dominated by the idle energy and refresh energy. Therefore, our results in Figure 7 could hardly show the difference among different categories of ap-

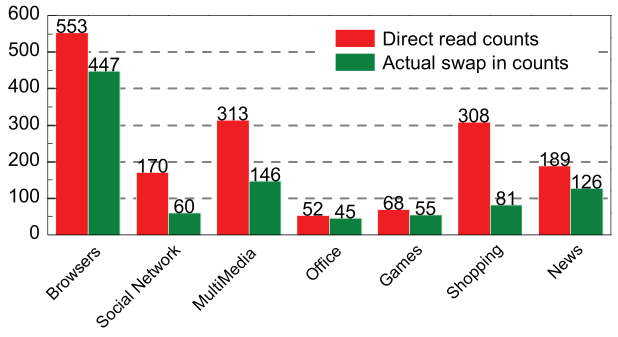

Figure 6: The number of direct reads in 30 minutes.

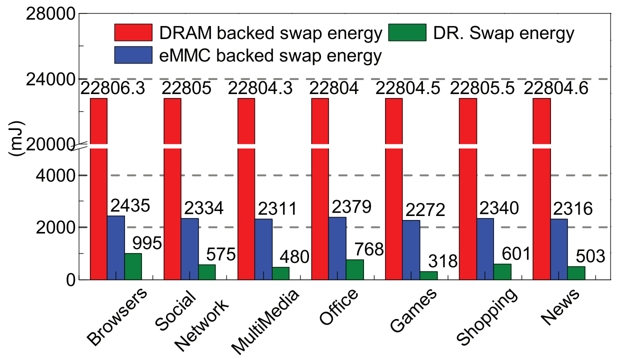

Figure 7: Comparing energy consumption between DR. Swap, eMMC swap and DRAM swap.

plications when using DRAM backed swap. However, we observe uniformly much lower energy consumption for DR. Swap compared to eMMC flash backed swap. Though eMMC flash backed swap also reduces a considerable amount of energy consumption, it greatly degrades performance due to the sub-optimal I/O design. Therefore, we conclude that an in-memory paging architecture with the help of emerging byte-addressable NVM is the ultimate solution for effective and efficient swapping for smartphones.

5. RELATED WORK

Reducing energy consumption in smartphones has been a focus in the research community. Wang et al. [24] uses profile-based battery traces to estimate the power consumption of mobile applications. To better understand energy consumption in smartphones, Perrucci et al. [18] measured and compared the energy consumed by different components in mobile devices. Shen et al. [22] proposed an energy-efficient caching and prefetching by considering the characteristics of mobile systems such as data update and user request patterns. Lee et al. [14] focused on the optimization of the power delivery network(PDN) in smartphones.

Due to its low standby power, high density and byte addressability, PCM is considered as a promising DRAM alternative [12, 21]. To improve PCM's performance and lifetime, various techniques and systems have been proposed [2, 7, 9, 12, 16, 19, 25, 28]. Hybrid approaches are also used, such as hybrid cache [15] and using mobile RAM and NVM together [4]. Though we do not target at any specific NVM products, we expect them to be energy-efficient, fast, and cheap as predicted. Our system could be easily ported to work with different future NVM technologies.

6. CONCLUSIONS

Reducing energy consumed by DRAM is critical for saving battery lifetime in smartphones. Emerging NVM's energy-efficiency and byte-addressability make it attractive for swapping in smartphones. In this paper, we have proposed *DR. Swap*, an energy-efficient in-memory paging (IMP) architecture to reduce energy consumption in smartphones. We re-adopt swapping in smartphones by replacing part of the DRAM with NVM, and using it as a swap area. We also propose direct read which guarantees zero-copy for read-only pages in the swap area. Experimental results based on Google Nexus 5 show that on average DR. Swap can reduce more than 60–80% energy consumption when compared to flash and DRAM based swap architectures, respectively.

ACKNOWLEDGEMENTS

This work is partially supported by the National Natural Science Foundation of China (61309004), National 863 Program (2013AA013202), Research Fund for the Doctoral Program of Higher Education of China (20130191120030), Chongqing cstc2012ggC40005 and cstc2013jcyjA40025, Fundamental Research Funds for the Central Universities (CDJZR14185501).

REFERENCES

[1] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. *USENIX ATC*, 2010.

[2] S. Cho and H. Lee. Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance. *MICRO*, pages 347–357, 2009.

[3] J. Cooke. Flash memory technology direction. *Micron Applications Engineering Document*, 2007.

[4] R. Duan, M. Bi, and C. Gniady. Exploring memory energy optimizations in smartphones. *IGCC*, pages 1–8, 2011.

[5] S. Eilert, M. Leinwander, and G. Crisenza. Phase change memory: A new memory enables new memory usage models. *IMW*, pages 1–2, 2009.

[6] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane, H. Yamada, M. Shoji, H. Hachino, C. Fukumoto, H. Nagao, and H. Kano. A novel nonvolatile memory with spin torque transfer magnetization switching: spin-ram. *IEDM*, pages 459–462, 2005.

[7] J. Hu, C. J. Xue, Q. Zhuge, W.-C. Tseng, and E. H.-M. Sha. Write activity reduction on non-volatile main memories for embedded chip multiprocessors. *ACM TECS*, pages 77:1–77:27, 2013.

[8] H. Huang, P. Pillai, and K. G. Shin. Design and implementation of power-aware virtual memory. *ATEC*, 2003.

[9] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. Childers. Improving write operations in MLC phase change memory. *HPCA*, pages 1–10, 2012.

[10] A. Jog, A. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer, and C. Das. Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs. *DAC*, pages 243–252, 2012.

[11] H. Kim, N. Agrawal, and C. Ungureanu. Revisiting storage for smartphones. *FAST*, 2012.

[12] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. Architecting phase change memory as a scalable DRAM alternative. *ISCA*, pages 2–13, 2009.

[13] M. Lee, E. Seo, J. Lee, and J.-S. Kim. PABC: Power-aware buffer cache management for low power consumption. *IEEE TC*, 56(4):488–501, 2007.

[14] W. Lee, Y. Wang, D. Shin, N. Chang, and M. Pedram. Optimizing the power delivery network in a smartphone platform. *IEEE TCAD*, pages 36–49, 2014.

[15] J. Li, L. Shi, C. Xue, C. Yang, and Y. Xu. Exploiting set-level write non-uniformity for energy-efficient nvm-based hybrid cache. *ESTIMedia*, pages 19–28, 2011.

[16] D. Liu, T. Wang, Y. Wang, Z. Qin, and Z. Shao. PCM-FTL: A write-activity-aware NAND flash memory management scheme for PCM-based embedded systems. *RTSS*, pages 357–366, 2011.

[17] J. Park, H. Han, and S. Cho. Extending main memory with flash – the optimized SWAP approach. *NVMW*, 2014.

[18] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer. Survey on energy consumption entities on the smartphone platform. *VTC*, pages 1–6, 2011.

[19] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali. Enhancing lifetime and security of PCM-based main memory with Start-gap wear leveling. *MICRO*, pages 14–23, 2009.

[20] A. Rice and S. Hay. Decomposing power measurements for mobile devices. *PerCom*, pages 70–78, 2010.

[21] Z. Shao, Y. Liu, Y. Chen, and T. Li. Utilizing PCM for energy optimization in embedded systems. *ISVLSI*, pages 398–403, 2012.

[22] H. Shen, M. Kumar, S. K. Das, and Z. Wang. Energy-efficient data caching and prefetching for mobile devices based on utility. *Mob. Netw. Appl. 2005*, 10(4):475–486.

[23] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. The missing memristor found. *Nature*, 2008.

[24] C. Wang, F. Yan, Y. Guo, and X. Chen. Power estimation for mobile applications with profile-driven battery traces. *ISLPED*, pages 120–125, 2013.

[25] J. Wang, X. Dong, Y. Xie, and N. Jouppi. i2WAP: Improving non-volatile cache lifetime by reducing inter- and intra-set write variations. *HPCA*, pages 234–245, 2013.

[26] H. S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson. Phase change memory. *Proceedings of the IEEE*, 98(12):2201–2227, 2010.

[27] C. Xue, G. Sun, Y. Zhang, J. J. Yang, Y. Chen, and H. Li. Emerging non-volatile memories: Opportunities and challenges. *CODES+ISSS*, pages 325–334, 2011.

[28] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. A durable and energy efficient main memory using phase change memory technology. *ISCA*, pages 14–23, 2009.