Energy-Efficient In-Memory Paging for Smartphones

Kan Zhong, Duo Liu, Liang Liang, Xiao Zhu, Linbo Long, Yi Wang, and Edwin Hsing-Mean Sha

***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, dynamic random access memory (DRAM)-based main memory subsystem is a major contributor to the energy consumption of mobile devices. In this paper, we propose direct read (DR). Swap, an energy-efficient in-memory paging design to reduce energy consumption in smartphones. In DR. Swap, we adopt emerging energy-efficient nonvolatile memory (NVM) and use it as the swap area. Utilizing NVMs byte-addressability, we propose DR which guarantees zero memory copy for read-only requests when accessing a page in swap area. To better understand the energy consumption of swapping, we build an energy model to analyze the energy consumption of different paging architectures. We evaluate DR. Swap based on the Google Nexus 5 smartphone, experimental results show that our technique can reduce more than 50% energy consumption compared to DRAM backed swapping.**

***Index Terms*—Energy, in-memory paging (IMP), nonvolatile memory (NVM), smartphone, swapping.**

I. 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, multitasking 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 2000–3000 mAh. 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 run-time data, leading to high energy consumption.

TABLE I
COMPARING PCM, DRAM, AND NAND FLASH [8]

Attributes	DRAM	PCM	NAND
Non-Volatile	No	Yes	Yes
Idle power	$\sim$W/GB	$\ll$0.1 W	$\ll$0.1 W
Read latency	50 ns	50 - 100 ns	10-25 us
Write latency	$\sim$20 - 50 ns	$\sim$1 us	$\sim$100 us
Endurance	∞	10^8	$10^4 - 10^5$

Recent research has shown that the Samsung Galaxy S3's main memory, which is 1 GB DRAM can consume around 20% of the overall energy [2]. What makes the situation worse is the trend of adopting large main memories to support feature-rich applications. For example, Google Nexus 6 has as much as 3 GB main memory.[1] Larger main memory improves system performance, but inevitably leads to higher energy consumption [3], [4]. Moreover, the battery technology cannot catch up with the energy demands of smartphones [5], leading to rise more pressure on battery lifetime. Thus, reducing the energy consumption of main memory becomes critical in smartphones. Most existing work [6], [7] 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 readopt swapping with the help of emerging byte-addressable, nonvolatile memory (NVM). Swapping is an effective way of extending memory by writing inactive pages to storage spaces [9]. It has long been a standard feature in modern OSes, but smartphones seldom use it because of the suboptimal performance and limited endurance of storage (NAND flash). Though flash memory has much better energy consumption parameters than

Manuscript received May 20, 2015; revised August 6, 2015 and October 28, 2015; accepted November 27, 2015. Date of publication December 29, 2015; date of current version September 7, 2016. This work was supported in part by the National Natural Science Foundation of China under Project 61309004, in part by the National 863 Program under Grant 2015AA015304, in part by the Research Fund for the Doctoral Program of Higher Education of China under Grant 20130191120030, in part by the Chongqing High-Tech Research Program under Grant cstc2013jcyjA40025, in part by the Fundamental Research Funds for the Central Universities under Grant CDJZR14185501 and Grant 0214005207005. A preliminary version of this paper was presented at the ACM/IEEE 2014 International Symposium on Low Power Electronics and Design (ISLPED 2014) [1]. This paper was recommended by Associate Editor T. Mitra. *(Corresponding author: Duo Liu.)*

K. Zhong, D. Liu, X. Zhu, L. Long, and E. H.-M. Sha are with the College of Computer Science, Chongqing University, Chongqing 400044, China, and the Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, Chongqing 400044, China (e-mail: liuduo@cqu.edu.cn).

L. Liang is with the College of Communication Engineering, Chongqing University, Chongqing 400044, China.

Y. Wang is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2015.2512904

[1] http://www.google.com/nexus/6

DRAM, it could not be used as the swap area while maintaining acceptable performance. What is more, frequent swap in and out operations can also wear out certain flash memory blocks quickly. Compared to flash memory, byte-addressable NVMs such as phase change memory (PCM) [10] and memristor [11] offer not only better (near-DRAM) performance, but also lower energy consumption. As shown in Table I, PCM exhibits much better energy parameters when compared to DRAM. It also exhibits much shorter read and write latency when compared to NAND flash [12]. A plethora amount of work have been proposed to further achieve near-DRAM performance and better endurance for PCM [13]–[20]. Other NVMs such as spin-transfer torque random access memory (STT-RAM) [21]–[23] could promise even faster performance and better endurance than DRAM [24]. Therefore, we do not specifically consider the endurance or latency issues and focus on energy consumption in this paper. Unlike flash memories, these emerging 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 (IMP) architecture called direct read (DR). Swap, to readopt 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 NVMs byte-addressability, we allow DR for read requests directly from the swap area, guaranteeing zero-copy for read-only (RO) 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. With DR, we avoid unnecessary memory copying to DRAM, furthering reduce energy consumption. To better understand the energy behavior of swapping, we present a fine-gained energy model to analyze the energy consumption of different paging architectures, and we use the proposed energy model to evaluate the energy efficiency of DR. Swap.

In summary, we make the following contributions.

1) We revisiting swapping in smartphones and propose an in-memory architecture with the help of byte-addressable NVMs to reduce the energy consumption of smartphones while maintaining high performance.
2) We propose DR to avoid unnecessary memory copying induced by RO requests, furthering reduce energy consumption.
3) We present an data sheet-based energy model to analyze the energy behaviors of different paging architectures.

The reminder of this paper is organized as follows. In Section II, we give related backgrounds about swapping, energy-related issues in smartphones and emerging byte-addressable NVM. In Section III, we present the details of the design of DR. Swap, including the IMP architecture and the DR optimization. In Section IV, we discuss the energy model for different paging architectures. Evaluation results are shown in Section V. We summarize the related work and the conclusion in Sections VI and VII, respectively.

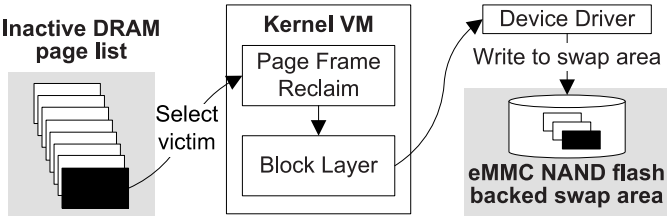


Fig. 1. Traditional NAND flash backed swapping.

II. Background

In this section, we first give the background on swapping, and then we introduce the energy-related issues in smartphones and the emerging byte-addressable NVM. We use Google Android as an example as it is the most widely adopted smartphone OS. Note that this paper can also be extended to support other platforms.

A. Swapping and Paging in Smartphones

Swapping is an effective way to extend memory space by borrowing space from secondary storage devices (e.g., NAND flash) in modern OSes [9]. With paging, swapping becomes more flexible as processes could be swapped in and out in units of noncontiguous pages. When the system is under pressure and finds itself difficult to satisfy memory allocation requests, the OS will write some inactive pages to swap area and allocate these page frames to the requesting applications. Because of the scarcity of DRAM and the abundance of disk and flash memory in capacity, a traditional swap area is usually backed by these two types of devices via I/O interface. The swap area is divide into slots, each of which is precisely the size of a DRAM page. Fig. 1 shows an embedded multi-media card (eMMC) NAND flash memory backed swapping for smartphones. As shown, when the memory is under pressure, the page frame reclaiming routine will start to select inactive pages and swap them to the eMMC NAND flash memory. Pages that being swapped out must go through all the storage stack to be written to the storage medium. However, due to the suboptimal storage (NAND flash memory) performance, swapping is usually not enabled by smartphones.

To avoid poor performance, mainstream mobile OSes such as Android disables swapping and implements a low memory killer (LMK) to reclaim memory by terminating certain processes when the system is under memory 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 Fig. 2. With a

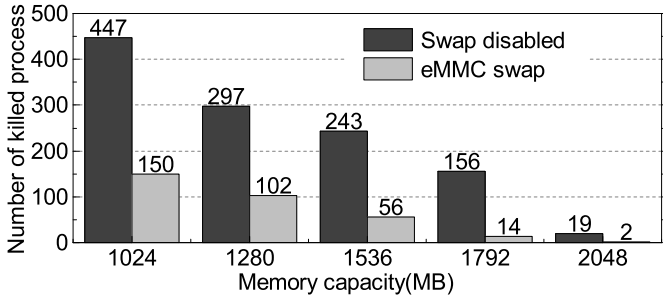


Fig. 2. Number of processes killed with and without a swap area under different DRAM sizes.

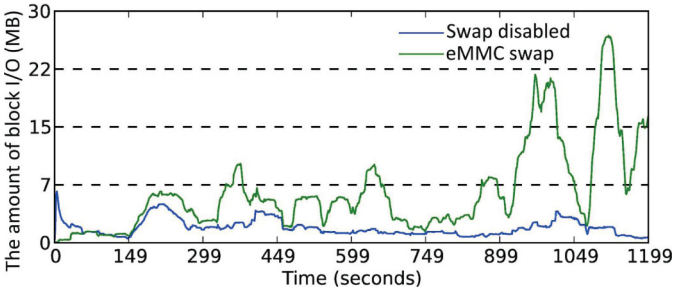


Fig. 3. Comparison I/O between eMMC swap and native Android OS with swap disabled.

128 MB flash backed swap area, the number of killed processes could be significantly reduced (e.g., from 447 to 150 with 1 GB memory). Fig. 3 highlights the amount of block I/O induced by a flash backed swap. As shown, swapping greatly increases I/O operations. Recent research has shown that storage plays a significant role in application performance [25]. 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 suboptimal overall performance. Moreover, the erase count of eMMC NAND flash is limited to 10^5 [26] cycles, frequently writing to the swap area further reduces the lifetime of NAND flash. Therefore, we argue that smartphones should readopt swapping with the help of emerging NVM to extend the main memory space.

B. 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., 2000–3000 mAh). This implies that energy becomes a first-class citizen in smartphones. In particular, the energy consumed by DRAM is non-negligible [27]. The DRAM-based main memory is a major contributor to the overall energy consumption of a smartphone. It is reported that Samsung Galaxy S3's 1 GB DRAM-based main memory accounts around for 20% of its overall energy consumption [2]. Recent mainstream phones have equipped with 3 GB DRAM, such as Google Nexus 6, or even 4 GB DRAM, such as ASUS Zenfone 2.[2] Since DRAM requires constant current to maintain its data, the trend of adopting large main memories to support feature-rich applications make the DRAM consumes more energy. Moreover, in most smartphone OSes, applications are not fully closed (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, excessive energy is consumed by the DRAM-based main memory to maintain these run-time data, leading to high energy consumption. Thus, reducing the energy consumption of DRAM-based main memory becomes critical in smartphones. In this paper, we reduce the energy consumption of smartphones by replacing part of the DRAM with emerging byte-addressable NVM backed swap area without sacrificing performance.

[2]http://www.asus.com/Phones/ZenFone_2_ZE551ML

C. Emerging Byte-Addressable NVM

Emerging byte-addressable NVMs such as PCM [10], STT-RAM [21], and memristor [11] has been extensively studied for replacing DRAM as main memory, static random access memory as on-chip cache, and even flash as secondary storage due to their attractive features, such as low power consumption, high density, and byte-addressability. Compared to DRAM, which needs constant voltage to maintain its data, NVM keeps data by changing the physical state of its underlying material, such as resistance level. One of the promising candidates is PCM, which uses the state changing between amorphous and crystalline of phase change materials (e.g., $Ge_2Sb_2Te_5$) to record logic zeros and ones.

However, NVM is an asymmetric read–write (RW) technology, the write latency is much higher compared to the read latency (e.g., PCM write latency is 4×–8× higher than read latency). In this paper, we do not regard the performance of NVM as a major problem, and we also do not concern the underlying hardware implementation, we assume that smartphones with both NVM and DRAM would become possible in the near future. The energy consumption of smartphones and the OS software design for NVM-based swap area are the main concerns in this paper. Particularly, we adopt PCM as the swap device in this paper.

III. Energy-Efficient Paging Design

In this section, we will present the details of DR. Swap, which consists of our energy-efficient IMP architecture and the DR optimization.

A. In-Memory Paging Architecture

Utilizing NVMs energy-efficiency and byte-addressability, DR. Swap consists of an IMP architecture and the DR optimization. Our IMP 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 memory) in existing systems. Compared to hybrid memories, swapping effectively reuses 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

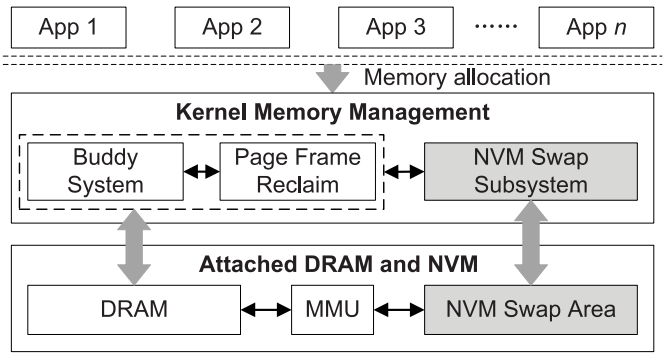


Fig. 4. IMP architecture. We replace the traditional storage-based swap area with memory-attached NVM. The memory management subsystem interacts with memory, instead of I/O devices (e.g., flash) to swap in/out pages.

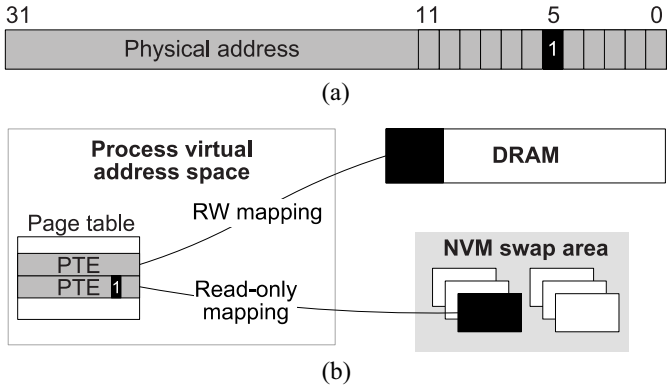


Fig. 5. Overview of DR. (a) PTE for DR. Bits 0 to 11 are PTE flags, bit 5 (i.e., the 6th bit) is used to identify whether the mapped page belongs to NVM swap area. The 6th bit is set to 1 when a page in NVM is linked to this PTE, otherwise, it is set to 0. (b) Direct read directly maps the NVM page to user space from the swap area, avoiding unnecessary memory copy operations for page reads.

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 NVMs high performance.

Since we attach NVM to the memory bus, the OS knows NVM shares part of the physical address space with DRAM. In this paper, we focus on the software design, and the proposed IMP architecture also do not need any changes to the memory controller. To let OS know which part of the whole address space belongs to NVM and manages it as a swap area, the e820 table need to be updated to provide information about the underlying DRAM and NVM layouts. 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.

Fig. 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 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 Fig. 1, IMP achieves both high performance and energy efficiency.

Note that when reclaiming memory space, only anonymous pages (i.e., pages do not correspond to any file) can be swapped to NVM swap area. On the contrary, for pages correspond to a portion of a file, if these pages are modified (i.e., dirty pages), they will be written back to their corresponding backup file(s) located on the external storage (e.g., NAND flash). Otherwise, these clean pages will be simply discarded when they are reclaimed. Moreover, when the NVM swap area is full, it cannot allocate space to store DRAM victim pages, and thus LMK has to start to reclaim memory space by terminating certain processes.

Currently, the IMP architecture is implemented in 32-bit architecture. However, the technique does not have any extra addition of complexity for the upcoming 64-bit architectures. Instead, our IMP architecture can benefit from the 64-bit architecture, since 64-bit architecture has a much larger address space than 32-bit architecture, and it is more flexible to arrange DRAM and NVM compared to 32-bit architecture, in which DRAM and NVM share a limited address space (4 GB). Therefore, in 64-bit architecture, the capacity of NVM swap area can be significantly extended.

B. 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.

Fig. 5 illustrates the overview of DR. We use the 6th bit of a PTE to distinguish DRAM page from NVM page in swap area, as shown in Fig. 5(a), if the 6th bit is set to 1, the mapped page is in NVM swap area, otherwise, the mapped page is in DRAM. To remove unnecessary memory copy operations between NVM and DRAM, as shown in Fig. 5(b), 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

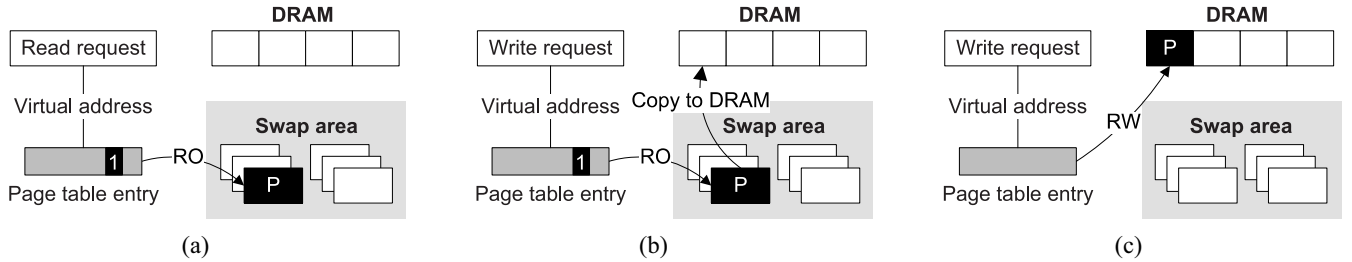


Fig. 6. Example of DR. (a) For read request, DR sets up RO mapping for the requested page directly without copying it back to DRAM. (b) Any attempt to write the page will trigger a page fault, in the page fault hander, DR copies the requested page to DRAM. (c) DR sets up the RW mapping for the page in DRAM.

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. Due to the asymmetric nature of most NVMs latency (e.g., PCM has much faster reads than writes) and wear leveling concerns, we do not allow "direct write" for write requests, since it may introduce a great hardware overhead. To achieve this, the hardware (i.e., memory controller) has to be modified to determine how many times the NVM page is written to and what is the write frequency. Different from copy-on-write (CoW), which is usually used in creating child process and aims to reduce the memory consumption and improve performance, our proposed DR aims to reduce the memory copy operations by using the byte-addressability of NVM. Moreover, CoW copies shared DRAM pages to new pages when they are written to. In DR, when NVM pages are written to, they are first copied to DRAM and then returned to the NVM swap subsystem.

Fig. 6 gives an example of DR. Assume *P* is a page in swap area and now a read request is issued from user space, instead of copying it back to DRAM and then updating the PTE, DR sets up a read only mapping for page *P* and sets the corresponding PTEs 6th bit to 1. After that, data in *P* can be read directly, as shown in Fig. 6(a). Due to the mapping of page *P* is read only, any write to *P* will trigger page fault. In the fault handler, we first check the PTEs 6th bit, if it is equal to 1, which means the mapped page is in swap area. In this case, we migrate page *P* from NVM swap area back to DRAM, remove the old mapping and set up the read and write mapping for page *P*, then *P* can be written to in DRAM, as shown in Fig. 6(b) and (c). As shown, for read only requests, DR can avoid the unnecessary memory copy effectively.

IV. Energy Model

In this section, we build energy model to analyze the energy consumption of different swapping schemes, including DRAM backed swapping, NAND flash backed swapping, and DR. Swap. Note that we dedicate `ramdisk` as swap area for DRAM backed swapping. Table II lists the power and timing notations used in the proposed energy model. Based on the model, we compare the energy consumption of different swapping schemes in Section V, to show how much energy is saved by our IMP architecture and the DR optimization.

TABLE II
Power and Timing Notations Used in the Energy Model

Specific power notations for DRAM and PCM			
Power state	Value (mW)		Description
	DRAM	PCM	
PRE_PDN	1.2	0.2	Background power of precharge power-down
PRE_STBY	6.8	3.5	Background power of idle standby
ACT_PDN	2.3	0.1	Background power of active power-down
ACT_STBY	9.3	4.8	Background power of active standby
REF	12.4	0	Background power to complete refreshes
ACT	76.7	156.0	Bank active power
RD	246.7	148.2	Burst read power
WR	246.0	232.7	Burst write power
DQ	33.8	20.3	Driving data output
Specific timing notations of DRAM and PCM			
Notation	Value		Description
	DRAM	PCM	
${}^tCK$	2.5 ns	5 ns	Average clock period
${}^tRCD$	42 ns	80 ns	Active to access delay
${}^tWR$	15 ns	15 ns	Write recovery time
RL	6 cycles	3 cycles	Data read access delay
WL	4 cycles	1 cycles	Data write access delay
BL	8	8	Burst read/write length
BW	32 bit	16 bit	Data bus width
Specific power and timing notations for eMMC flash			
Notation	Value		Description
f	26 MHz		Clock frequency
V_{DD}	3.3 V		Clock frequency
RL	2 cycles		Data read access delay
I_{DD1}	100 mA		Read access current
RD_BL	512 bytes		Read data block length
WL	32 cycles		Data write access delay
I_{DD2}	100 mA		Write access current
WR_BL	512 bytes		Write data block length
BW	8 bit		Data bus width
I_{DD3}	350 uA		Standby current

A. DRAM Backed Swapping

In DRAM backed swapping, swap-in and swap-out are implemented by memory copy operations—pages are copied from/to main memory (i.e., DRAM) to/from DRAM backed swap area (i.e., `ramdisk`) via `memcpy()`. Each memory copy operation involves a combination of DRAM operations, and the energy consumption of each DRAM operations constitute the overall energy dissipated within the DRAM backed swap area.

Basically, the overall energy consumption of DRAM backed swapping is comprised of four parts: 1) background power (i.e., PRE_PDN, PRE_STBY, ACT_PDN, ACT_STBY, and REF); 2) active power (i.e., ACT); 3) read/write power

(i.e., RD and WR); and 4) I/O power (i.e., DQ) [28], [29]. The background power is the power that a DRAM chip consumes all the time with or without operations, and it will increasing along with the DRAM chip size. There are five power states regarding the background power of DRAM as shown in Table II. Active power is the power used when DRAM is in active state, which performs banks activation and precharge. Read/write power is for data read/write when DRAM is in read/write state. I/O power is used for driving the data bus to transmit data when DRAM is in driving data state. Therefore, the overall energy consumption of DRAM backed swapping is determined by

$$\begin{aligned} E_{\text{DRAM}} = &\big(P_{\text{DRAM\_PRE\_PDN}} + P_{\text{DRAM\_PRE\_STBY}} \\ &+ P_{\text{DRAM\_ACT\_PDN}} + P_{\text{DRAM\_ACT\_STBY}} \\ &+ P_{\text{DRAM\_REF}}\big) \times t_{\text{TOTAL}} \\ &+ P_{\text{DRAM\_ACT}} \times t_{\text{ACT}} \\ &+ P_{\text{DRAM\_RD}} \times t_{\text{RD}} + P_{\text{DRAM\_WR}} \times t_{\text{WR}} \\ &+ P_{\text{DRAM\_DQ}} \times t_{\text{DQ}}. \end{aligned} \tag{1}$$

Table II gives the power states of a DRAM used in this model, whose values are calculated in terms of a 1 GB LPDDR2-SDRAM [30]. We will discuss how to calculate the DRAM active time (i.e., t_{ACT}), read/write time (i.e., t_{RD}), and driving data time (i.e., t_{DQ}) by breaking down the swap-in/swap-out operation in the following. Note that t_{TOTAL} in 1 denotes the up time of DRAM backed swap area.

In DRAM backed swapping, each swap-in/swap-out involves a sequence of DRAM operations. It must begin with an `active` command to active banks and select row, and the data is transferred from the selected row into sense amplifiers. Then `read/write` commands can be issued to read/write data from or into the sense amplifiers. After read/write data, a `precharge` command must be issued to restore the data to the cells in the selected row. We assume `read/write` can be issued consecutively after the banks are activated. The data length of each read/write is denoted by BL, the data bus width is denoted by BW and the DRAM page size (e.g., 4 kB in default) is denoted by P_{size}. Therefore, for N_{rd} swap-ins, we totally need N_{rd} `active` and `precharge` commands, and $((N_{\text{rd}} \times P_{\text{size}} \times 8)/(\text{BL} \times \text{BW}))$ `read` commands. For N_{wr} swap-outs, we totally need N_{wr} `active` and `precharge` commands, and $((N_{\text{wr}} \times P_{\text{size}} \times 8)/(\text{BL} \times \text{BW}))$ `write` commands. The time consumed by active banks, read/write and driving data for N_{rd} swap-ins and N_{wr} swap-outs will be discussed below.

For each swap-in, data become available at RLth cycle after the first `read` command is issued. Since the data are transmitted in double data rate (DDR), for each `read` command, it takes (BL/2) cycles to drive the data output. Therefore, the time consumed by reading N_{rd} pages from DRAM backed swap area is

$$t_{\text{RD}} = \left(\frac{N_{\text{rd}} \times P_{\text{size}} \times 8}{\text{BL} \times \text{BW}} \times \frac{\text{BL}}{2} + N_{\text{rd}} \times \text{RL}\right) \times {}^{t}\text{CK}. \tag{2}$$

For each swap-out, the data shall be driven at WLth cycle after the first `write` command is issued. For each `write` command, it also takes (BL/2) cycles to transmit the data to sense amplifiers and additionally ^{t}WR to restore the data to the cells in the array. Thus, the time consumed by writing N_{wr} pages to DRAM backed swap area is

$$t_{\text{WR}} = \left[\frac{N_{\text{wr}} \times P_{\text{size}} \times 8}{\text{BL} \times \text{BW}} \times \left(\frac{\text{BL}}{2} + {}^{t}\text{WR}\right) + N_{\text{wr}} \times \text{WL}\right] \times {}^{t}\text{CK}. \tag{3}$$

`Read` or `write` commands can be accepted at ^{t}RCD after the `active` command is issued. Thus, the time consumed by active operations for N_{rd} swap-ins and N_{wr} swap-outs is determined by

$$t_{\text{ACT}} = t_{\text{RD}} + t_{\text{WR}} + (N_{\text{rd}} + N_{\text{wr}}) \times {}^{t}\text{RCD}. \tag{4}$$

The time consumed by data driving for N_{rd} swap-ins and N_{wr} swap-outs is determined by

$$t_{\text{DQ}} = \frac{(N_{\text{rd}} + N_{\text{wr}}) \times P_{\text{size}} \times 8}{\text{BL} \times \text{BW}} \times \frac{\text{BL}}{2} \times {}^{t}\text{CK}. \tag{5}$$

The total energy consumed by DRAM backed swap area can be extracted from (1) after knowing the number of swap-ins and swap-outs.

B. NAND *Flash Backed Swapping*

In mobile devices, eMMC, which comprises both controller and NAND flash chips is employed as the storage system. Therefore, we assume the NAND flash backed swap area is built in an eMMC device, and pages are copied from/to main memory (i.e., DRAM) to/from NAND flash backed swap area by reading/writing NAND flash memory.

Unlike DRAMs, flash memory do not require modeling all possible states since the operation of flash memory is much simpler than that of DRAM. The power consumption of flash memory is comprised of three parts: 1) read power; 2) write power; and 3) standby power. Note that the standby power is the power consumed by standby state, in which the flash memory is without any operation, and it will go to sleep automatically. Therefore, the overall energy dissipated in NAND flash backed swap area is determined by

$$\begin{aligned} E_{\text{FLASH}} = &P_{\text{FLASH\_RD}} \times t_{\text{RD}} \\ &+ P_{\text{FLASH\_WR}} \times t_{\text{WR}} \\ &+ P_{\text{FLASH\_STBY}} \times t_{\text{STBY}}. \end{aligned} \tag{6}$$

The power of each state can be extracted from the corresponding data sheet. Table II lists the power and timing parameters of SanDisk iNAND Ultra eMMC [31]. According to Table II, the power of each state can be expressed as

$$P_{\text{FLASH\_RD}} = V_{\text{DD}} \times I_{\text{DD1}} \tag{7}$$

$$P_{\text{FLASH\_WR}} = V_{\text{DD}} \times I_{\text{DD2}} \tag{8}$$

$$P_{\text{FLASH\_STBY}} = V_{\text{DD}} \times I_{\text{DD3}}. \tag{9}$$

For read operation, data become available at the RLth cycle after the `read` command is issued. For write operation, data become available at the WLth cycle after the `write` command is issued. We assume the eMMC device transmits data in DDR. Thus, the time consumed by each read operation

is $(1/f) \times (((\text{RD\_BL} \times 8)/(2 \times \text{BW})) + \text{RL})$ and the time consumed by each write operation is $(1/f) \times (((\text{WR\_BL} \times 8)/(2 \times \text{BW})) + \text{WL})$.

Thus, the total time consumed by reading N_{rd} pages from flash backed swap area is

$$t_{\text{RD}} = \frac{1}{f} \times \left(\frac{\text{RD\_BL} \times 8}{2 \times \text{BW}} + \text{RL} \right) \times \frac{N_{\text{rd}} \times P_{\text{size}}}{\text{RD\_BL}}. \quad (10)$$

The time consumed by writing N_{wr} pages to flash backed swap area is

$$t_{\text{WR}} = \frac{1}{f} \times \left(\frac{\text{WR\_BL} \times 8}{2 \times \text{BW}} + \text{WL} \right) \times \frac{N_{\text{wr}} \times P_{\text{size}}}{\text{WR\_BL}} \quad (11)$$

and the eMMC device total standby time is

$$t_{\text{STBY}} = t_{\text{TOTAL}} - t_{\text{RD}} - t_{\text{WR}}. \quad (12)$$

Similar to DRAM backed swapping energy model, the energy consumed by flash backed swap area can be extracted from (6) after knowing the total number of swap-ins and swap-outs.

C. DR. Swap

Low power DDR (LPDDR) interface is widely adopted in mobile devices due to its low power consumption and high data transfer rate. Therefore, to model the energy consumption of DR. Swap, we assume PCM with LPDDR interface is adopted to build a PCM-based swap area for smartphones. Due to the same memory interface, the power states of PCM in smartphones are the same as that of DRAM.

Since NVM is attached to the memory bus and use memory copy to swap in/out pages, the operations involved by swap-in and swap-out of DR. Swap are the same as that of DRAM backed swapping. However, compared to DRAM, PCM does not require any refresh current to maintain its data. Thus, the overall energy dissipated within the PCM backed swap area is determined by

$$\begin{aligned} E_{\text{PCM}} = {} & (P_{\text{PCM\_PRE\_PDN}} + P_{\text{PCM\_PRE\_STBY}} \\ & + P_{\text{PCM\_ACT\_PDN}} + P_{\text{PCM\_ACT\_STBY}}) \\ & \times t_{\text{TOTAL}} + P_{\text{PCM\_ACT}} \times t_{\text{ACT}} \\ & + P_{\text{PCM\_RD}} \times t_{\text{RD}} + P_{\text{PCM\_WR}} \times t_{\text{WR}} \\ & + P_{\text{PCM\_DQ}} \times t_{\text{DQ}}. \end{aligned} \quad (13)$$

Table II lists the power states and timing parameters of a PCM with LPDDR2 memory interface, whose power values are calculated based on the LPDDR2-PCM [32] data sheet using the methods reported in [28] and [29]. Compared to DRAM backed swapping, the number of swap-ins in DR. Swap is reduced by DR. Therefore, for DR. Swap energy model, the calculation of t_{WR}, t_{ACT}, and t_{DQ} are the same as that of DRAM backed swapping energy model except for t_{RD}. Let N'_{rd} denotes the actual swap-ins (which copy pages from PCM to DRAM) in DR. Swap, and t_{RD} is determined by

$$t_{\text{RD}} = \left(\frac{N'_{\text{rd}} \times P_{\text{size}} \times 8}{\text{BL} \times \text{BW}} \times \frac{\text{BL}}{2} + N'_{\text{rd}} \times \text{RL} \right) \times {}^{t}\text{CK}. \quad (14)$$

Since both DRAM backed swapping and flash backed swapping do not support DR, in the above equation, $N'_{\text{rd}} < N_{\text{rd}}$. Therefore, DR reduce the energy consumption by reducing the number of swap-ins. However, as shown in the proposed energy model, the energy consumption of different swapping schemes not only relay on the number of swap-ins and swap-outs, but also relay on the underlying hardware. DRAM, flash, and PCM are three different memories, therefore exhibit very diverse energy dissipation. We give the energy consumption results of different swapping schemes in Section V.

V. Evaluation

In this section, we give the evaluation results of DR. Swap. We implement and evaluate DR. Swap based on Google Nexus 5. Besides DR. Swap, we also implement DRAM backed swapping (i.e., `ramdisk` backed swapping) and NAND flash backed swapping for comparison. In the rest of this section, we first describe the experimental setup, then give the experimental metrics and methodology. Finally, we present and discuss the experimental results.

A. Experimental Setup

We evaluate DR. Swap based on the Google Nexus 5, which is a smartphone with a 2.3 GHz Qualcomm 8974 processor, 2 GB DRAM (i.e., LPDDR3-SDRAM), and 16 GB eMMC-based NAND flash memory; the phone is running Android Kitkat 4.4.4 and Linux kernel 3.4.0 which has been modified to implement DR. Swap. In order to communicate with Nexus 5, we setup the Android debug bridge (ADB) on a Linux machine running Fedora 21. ADB is a command line tool provide by the Android software development kit, and it allows a host computer to communicate with the phone via universal serial bus (USB) in the USB debug mode. For 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, the phone is full charged to make sure it is working in its full performance capability. All the radio communication functionalities are disabled except the wireless network as most applications need Internet connection to work properly.

In the evaluation of DR. Swap, we use the PCM-based swap area as a case study. More specifically, we adopt Micron LPDDR2-PCM [32] as the swap area for DR. Swap. LPDDR2-PCM is a 45 nm technology-based PCM product from Micron with clock frequency up to 400 MHz and random read up to 400 MB/s. However, our system does not rely on any specific type of NVM product and can be easily adopted by different NVM-based systems. In this paper, we do not focus on which NVM can be served as the swap area, instead, we focus on the OS software design for NVM backed swapping. Moreover, based on our energy model, the core parameters are the number of swap-ins and swap-outs. Therefore, in our current implementation, we simply use a DRAM partition to simulate the PCM-based swap area.

Compared to DRAM, PCM is slower and has limited lifetime, so we need to read/write DRAM multiple times for each swap-in/out operation. To obtain more realistic simulation values, NVSim [33] is adopted to calculate the access latency of

TABLE III
WORKLOAD APPLICATIONS

Browser		**News**	
App name	Size	App name	Size
Google Chrome	84.53MB	BBC News	13.28MB
Firefox	38.93MB	Flipboard	12.37MB
Opera	59.84MB	Google Newsstand	18.18MB
Dolphin Browser	36.91MB	Feedly	13.29MB
Next Browser	24.81MB	Yahoo News Digest	16.30MB
Multimedia		**Social networking**	
App name	Size	App name	Size
KMPlayer	38.42MB	Facebook	99.14MB
MX Player	27.82MB	Twitter	24.38MB
Google Play Music	23.94MB	Google Plus	56.14MB
TTPod	18.93MB	Instagram	19.16MB
StormPlayer	31.77MB	Pinterest	22.41MB
Gaming		**Online shopping**	
App name	Size	App name	Size
Hill Climb Racing	34.84MB	Amazon	85.45MB
Temple Run	25.32MB	TaoBao	116.0MB
Boom Beach	107.0MB	eBay	26.02MB
Angry Birds	61.84MB	Fancy	20.54MB
Crazy Snowboard	26.93MB	Google Play Store	17.54MB
Mix1		**Mix2**	
Google Chrome, Flipboard, TTPod, Instagram, Angry Birds		Firefox, BBC News, Facebook, Temple Run, Amazon	

PCM and DRAM by feeding the corresponding parameters, such as data width and process node. According to the calculation results, the read and write performance of DRAM is around 2× and 12× faster than that of PCM, respectively. Therefore, for each swap-in operation (i.e., PCM read), we read DRAM two times, and for each swap-out operation (i.e., PCM write), we write DRAM 12 times. We believe that future mature PCM products will provide near-DRAM performance. The endurance issue of PCM is not considered as this paper mainly focuses on the design of NVM-based swapping and its energy consumption, we will study how to improve the endurance of NVM-based swap area in future work.

B. Application Benchmark

Table III lists the workload applications we adopted in the experiments. These applications are worldwide popular and daily used. Since different kinds of applications may have different memory access behaviors, we classify them into six categories, including browser, news, multimedia, social networking, gaming, and online shopping. To represent the realistic scenario, we also add two mixed categories, namely mix1 and mix2, by combining the applications selected from the above six categories, as shown in Table III. For a category, we run each application in foreground for 1 min, and we run all applications in round robin order for three times. Thus, each category needs 15 min to accomplish the evaluation. The detailed evaluation method is described as below.

1) *Browser:* For each run, we open the Google search home page, randomly type a keyword to search the results, and then we touch the screen to check the details of the first three search results.
2) *Social Networking:* Each social networking application is signed in before the test. For each run, we first drag down to refresh the posts, and then send a new post with both pictures and descriptions.
3) *Gaming:* We use popular games in this category. For each run, we play each game in this category for 1 min in default settings.
4) *Multimedia:* We chose both video players (i.e., MX Player, KMPlayer, and StormPlayer) and music players (i.e., Google Play Music and TTPod) in this category. For each video player, we play both 1080 p and 720 p video chips, for each Music player, we play 320 kbp music files.
5) *News:* For each run, we first refresh the news items, and then select three items to see the news content which both pictures and words.
6) *Online Shopping:* Each application is signed in before the test. For each run, we randomly type a keyword to search the products, and then click the first three items to see the products details.
7) *Mix1 and Mix2:* These applications are run use the methods described in their corresponding categories.

C. Metrics and Methodology

To evaluate the proposed technique, we run all the applications in each category to collect the following metrics. The corresponding methodology for each experimental metric is discussed as well.

1) Number of Memory Copy Operations: In DR. Swap, DR is designed to reduce the number of swap-ins, we use the number of memory copy operations to measure the effectiveness of DR. We run each category of applications shown in Table III for 15 min. All the applications are run in the variant with DR and without DR, respectively. We count the total number of swap-ins for each application category and compare the results between the two configures. For more accuracy, we run each application category in both configurations for five times and calculate the average value.

2) Energy Consumption: To evaluate the energy consumption of DR. Swap, we run all the applications in each application category to compare the energy consumption under different swapping schemes, including DRAM backed swapping, NAND flash backed swapping, and DR. Swap. The energy consumption of each swapping scheme is calculated using the energy model proposed in Section IV based on the number of swap-ins and swap-outs.

3) Application Switching Delay: Application switching delay is an very important performance metric for smartphone users. Application switching may cause inactive pages been swapping to swap area and requested pages been loading from swap area, especially when the system is under memory pressure, where application switching may trigger lots of swap-ins and swap-outs. Therefore, due to the performance differences, different swapping schemes can lead to different application switching delay. To collect this metric, we run all the application shown in Table III and compare the results between each swapping scheme. To measure the application switching delay, we use Android `am` tool to perform auto switching between applications and use the Linux `time` command to obtain the time consumed by application switching.

TABLE IV
MEMORY COPY OPERATION REDUCTION FOR EACH APPLICATION CATEGORY

Category	Swap size = 128MB			Swap size = 256MB			Swap size = 512MB		
	DR disabled	DR enabled	↓ (%)	DR disabled	DR enabled	↓ (%)	DR disabled	DR enabled	↓ (%)
Browser	6017	1771	70.57	5830	1224	79.01	3172	1102	65.26
News	1857	428	76.95	2200	449	79.59	1945	342	82.42
Multimedia	3038	1438	52.67	2172	807	62.85	1056	475	55.02
Social networking	4295	1150	73.22	2137	450	78.94	2086	583	72.05
Gaming	2174	711	67.30	1590	515	67.61	1795	592	67.02
Online shopping	2382	864	63.73	765	264	65.49	1136	385	66.11
Mix1	3280	1145	65.09	2685	1014	62.23	1836	796	56.64
Mix2	3343	996	70.21	2856	935	67.26	2166	673	68.93
Average	3298	1066	67.68	2529	707	72.04	1899	618	67.46

D. Number of Memory Copy Operations

Table IV compares the number of swap-ins between DR disabled and DR enabled in DR. Swap with different swap size. As shown, DR can reduce around 50%–75% swap-ins, which means a great number of memory copy operations are reduced. According to Table IV, we observe lots of memory copy operations reduction for each application category. In particular, news and social networking exhibit the highest memory copy reduction. According to memory management of Linux kernel, page fault is triggered when an accessed page is swapped out to the NVM swap area. Thus, in the page fault handler, the proposed DR sets up an RO mapping for the request page directly without copying it back to DRAM. Consequently, the number of memory copy operations is reduced between DRAM and NVM swap area. Note that the number of page faults cannot be reduced by DR. However, the number of page faults is related to the physical memory size, and thus it can be reduced by adding NVM swap area in the memory subsystem.

With DR, we reduce the number of memory copy operations by around 50% for multimedia applications and 65% for gaming and Online shopping applications, respectively. Moreover, for browser, news, and social networking applications, DR can reduce more than 70% of memory copy operations. In DR, the only overhead is updating the PTEs in the page fault handler when swapped out pages are requested. Compared to actual swap-in, which need at least one NVM page read, one DRAM page write, and one PTE update, the overhead of DR is negligible.

Besides, we observe that the average number of swap-ins decreases along with the increase of swap size. For example, the average number of swap-ins is decreased from 3289 to 1899 when the swap size is increased from 128 to 512 MB. This is mainly because that a larger swap area can store more inactive pages swapped from DRAM, leading to the DRAM have more free space to satisfy the memory requests of current running applications. Thus, pages belong to the current running application have less chance to be swapped out, resulting in less swap-ins for the current running applications. However, a larger swap area will increase the background power consumption. In the next section, we discuss the energy consumption of different swapping schemes.

E. Energy Consumption

Based on the number of swap-ins and swap-outs, the energy consumption of each swapping scheme can be obtained by using the energy model proposed in Section IV. For DRAM backed swapping and DR. Swap, the power consumption consists of background power, banks active power, data access power (i.e., read and write), and data driving power. For NAND flash backed swapping, the power consumption only consists of background power and data access power. Note that for DRAM backed swapping, the DRAM banks refresh power, which belongs to the background power is related to the swap size. Generally, the refresh power is proportional to the swap size, a larger swap area leads to more refresh energy consumption. As we have different swap size in our experiments, we need to scale the DRAM refresh power according to the swap size. Let S_{swap} denotes the DRAM backed swap area size and S_{chip} denotes the DRAM chip size which is 1 GB in our experiment. In the calculation of DRAM backed swapping energy consumption, the refresh energy is scaled with the factor $\text{scale}_s = (S_{\text{swap}}/S_{\text{chip}})$. For example, a 128 MB DRAM backed swap area is built on a 1 GB DRAM chip, the refresh power of DRAM backed swapping is scaled with $(128/1024) = 0.125$.

Fig. 7 shows the energy consumption of different swapping schemes with different size. The x-axis denotes the application category and the y-axis denotes the energy consumption in 15 min. Note that the DRAM refresh power has been scaled according to the swap area size. As shown in Fig. 7, DRAM backed swapping exhibits the highest energy consumption, and NAND flash backed swapping exhibits the lowest energy consumption. Along with the increasing of swap size, the energy consumed by DRAM backed swapping keeps increasing while the energy consumed by DR. Swap and NAND flash backed swapping almost stay the same. This is because for DRAM backed swapping, refresh power is proportional to the swap size, which will increasing along with the swap size. For DR. Swap and NAND flash backed swapping, the refresh power is zero as neither PCM nor NAND flash requires constant voltage to maintain its data. Because of the limited number of swap-ins and swap-outs during the 15 min, for DRAM backed swapping and DR. Swap, the energy consumption is dominated by the background energy consumption. Therefore, our results in Fig. 7 could hardly show the difference among different category of applications.

However, we observe uniformly much lower energy consumption for DR. Swap compared to DRAM backed swapping. DR. Swap can reduce more than 55%, 60%, and 65% energy consumption compared to DRAM backed swapping when the swap size is 128 MB, 256 MB, and 512 MB, respectively.

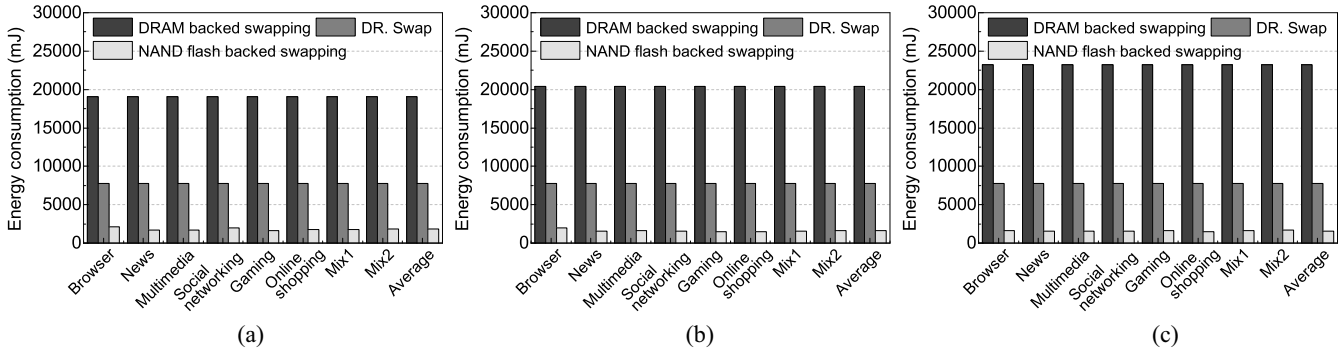


Fig. 7. Comparison of the energy consumption for each application category between different swapping schemes. (a) Swap size = 128 MB. (b) Swap size = 256 MB. (c) Swap size = 512 MB.

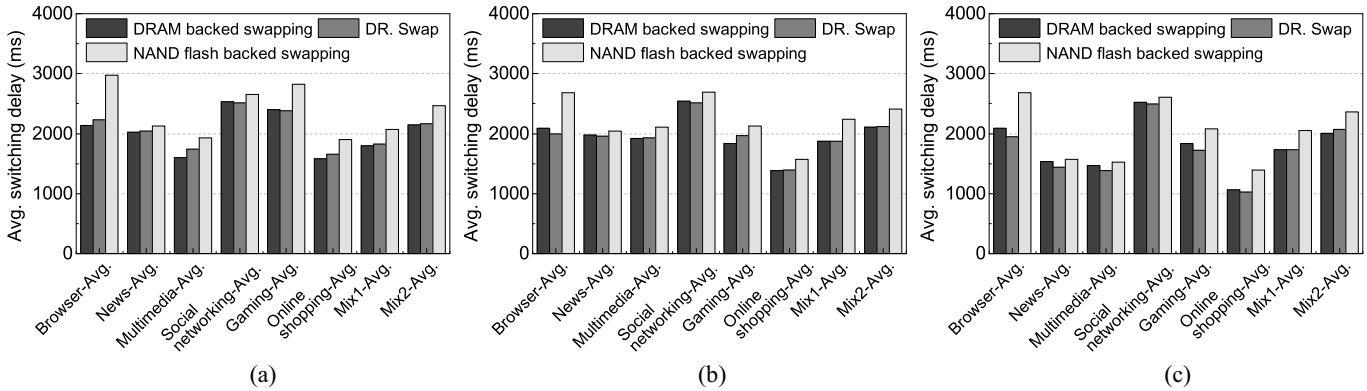


Fig. 8. Comparison of the average switching delay of each application category between different swapping schemes. (a) Swap size = 128 MB. (b) Swap size = 256 MB. (c) Swap size = 512 MB.

Since the background power of DRAM backed swapping increases along with the swap area size, DR. Swap can achieve more energy consumption reduction for a larger swap area. Though NAND flash backed swapping is more energy efficient compared to DRAM backed swapping and DR. Swap, it greatly degrades performance due to the suboptimal I/O design. Moreover, due to the limited P/E cycles of flash memory (e.g., 10^5 for SLC NAND flash and 10^4 for MLC NAND flash, respectively), NAND flash backed swapping can wear out the flash memory quickly. Therefore, we conclude that an IMP architecture with the help of emerging byte-addressable NVM is the ultimate solution for effective and efficient swapping for smartphones.

F. Application Switching Delay

Fig. 8 shows the average switching delay of each application category between different swapping schemes. As shown, DRAM backed swapping exhibits the lowest switching delay while NAND flash backed swapping exhibits the highest switching delay. Note that we already have emulated the access delay for DR. Swap as we use DRAM to simulate PCM. As shown, compared to DRAM backed swapping, DR. Swap only exhibits slightly higher switching delay than DRAM backed swapping. Table V lists the switching delay of each application under different swapping schemes. Note that the time unit is millisecond. As shown in Table V, most of the applications take 2 s to switch back. According to Fig. 8 and Table V, we observe that application switching speed of DR. Swap is uniformly faster than that of NAND flash backed swapping. For social networking applications in Fig. 8(a), and news, multimedia, and gaming applications in Fig. 8(c), the switching delay of DR. Swap even lower than that of DRAM backed swapping. For instance, as shown in Table V, the switching delay of Twitter in DR. Swap with a 128 MB size is 2093 ms, it is lower than that in DRAM backed swapping, which is 2150 ms. This is mainly because when swapping a page to DRAM backed swap area, which indeed is `ramdisk`-based, it still need to go through a "fake" I/O path. Though it finally call `memcpy` to copy inactive DRAM pages to `ramdisk`, the fake I/O path still leads to extra overheads. Therefore, DR. Swap sometimes even faster than DRAM backed swapping.

For NAND flash backed swapping, writing/reading a page to/from swap area needs to through the whole storage stack, which is much slower than writing/read a page to/from DRAM or PCM. As shown in Fig. 8, application switching speed of DR. Swap is around 10% faster than that of NAND flash backed swapping. For browser applications, the application switching speed of DR. Swap is even more than 25% faster than that of NAND flash backed swapping. Therefore, we conclude that application switching speed of DR. Swap is faster than that of NAND flash backed swapping, leading to a better user experience. We also observe that for news application, the switching speed only has a little improvement over the NAND flash backed swapping. The reason is that the size of news applications is smaller than that of other application categories.

TABLE V
APPLICATION SWITCHING DELAY OF EACH APPLICATION UNDER DIFFERENT SWAPPING SCHEME (UNIT: ms)

Category	Application	Swap size = 128MB			Improvement	Swap size = 256MB			Improvement	Swap size = 512MB			Improvement
		DRAM	DR. Swap	Flash	over Flash	DRAM	DR. Swap	Flash	over Flash	DRAM	DR. Swap	Flash	over Flash
Browser	Google Chrome	1286	1501	1643	8.64%	1399	1433	2717	47.26%	1399	1321	2717	51.38%
	Firefox	2051	2210	2274	2.81%	2170	2191	2408	9.01%	2170	2176	2408	9.63%
	Opera	2994	3033	5349	43.30%	2246	2134	3502	39.06%	2246	2087	3502	40.41%
	Dolphin Browser	1914	1881	2313	18.68%	2124	1877	2254	16.73%	2124	1884	2254	16.42%
	Next Browser	2426	2521	3300	23.61%	2525	2335	2554	8.57%	2525	2279	2554	10.77%
News	BBC News	2130	2085	2183	4.49%	2068	2041	2091	2.39%	1613	1475	1613	8.56%
	Flipboard	1400	1504	1569	4.14%	1424	1471	1509	2.52%	1090	1100	1133	2.91%
	Play Newsstand	2166	2178	2253	3.33%	2033	2080	2202	5.54%	1619	1488	1703	12.62%
	Feedly	2174	2151	2232	3.63%	2082	1999	2082	3.99%	1552	1543	1605	3.86%
	Yahoo News Digest	2271	2290	2411	5.02%	2266	2187	2305	5.12%	1796	1582	1796	11.92%
Multimedia	KMPlayer	2122	2215	2437	9.11%	2130	2150	2444	12.03%	1770	1601	1817	11.89%
	MX Player	1405	1481	1722	14.00%	1705	1666	1766	5.66%	1278	1255	1315	4.56%
	Google Play Music	725	825	975	15.38%	985	998	1117	10.65%	698	580	817	29.01%
	TTpod	1535	1717	1802	4.72%	2379	2402	2520	4.68%	1765	1633	1821	10.32%
	StormPlayer	2206	2450	2730	10.26%	2417	2440	2681	8.99%	1830	1830	1877	2.50%
Social networking	Facebook	4221	4200	4313	2.62%	4235	4163	4607	9.64%	4218	4122	4204	1.95%
	Twitter	2150	2093	2302	9.08%	2086	2121	2261	6.19%	2114	2111	2249	6.14%
	Google Plus	2155	2108	2338	9.84%	2163	2149	2225	3.42%	2125	2130	2246	5.16%
	Instagram	2084	2091	2161	3.24%	2111	2001	2154	7.10%	2098	1997	2167	7.84%
	Pinterest	2066	2050	2173	5.66%	2119	2112	2195	3.78%	2046	2091	2149	2.70%
Gaming	Hill Climb Racing	1928	1991	3197	37.72%	1464	1411	1560	9.55%	2160	1326	1600	17.13%
	Temple Run	1774	1745	1883	7.33%	1747	1700	1729	1.68%	1304	1421	1711	16.95%
	Boom Beach	2297	2259	2319	2.59%	1406	2165	2288	5.38%	1278	1830	2300	20.43%
	Angry Birds	3300	3208	3481	7.84%	2483	2477	2851	13.12%	2226	2111	2688	21.47%
	Crazy Snowboard	2709	2689	3244	17.11%	2057	2085	2210	5.66%	2212	1910	2103	9.18%
Online shopping	Amazon	901	1010	1339	24.57%	868	886	1011	12.36%	760	860	877	1.94%
	TaoBao	1851	1932	2133	9.42%	1564	1539	1711	10.05%	1350	1285	1593	19.33%
	eBay	1707	1800	1945	7.46%	1455	1460	1627	10.26%	1142	817	1468	44.35%
	Fancy	1689	1720	1995	13.78%	1458	1546	1631	5.21%	945	885	1484	40.36%
	Google Play Store	1765	1843	2110	12.65%	1552	1555	1869	16.80%	1131	1302	1554	16.22%
Mix1	Google Chrome	1320	1330	1609	17.34%	1311	1317	1721	23.47%	1302	1321	1647	19.79%
	Flipboard	1430	1460	1654	11.73%	1401	1442	1611	10.49%	1328	1336	1623	17.68%
	TTpod	1569	1630	1824	10.64%	2103	2119	2463	13.97%	1684	1701	1983	14.22%
	Instagram	2116	2142	2346	8.70%	2132	2009	2308	12.95%	2015	2000	2224	10.07%
	Angry Birds	2563	2589	2905	10.88%	2434	2486	3103	19.88%	2341	2318	2765	16.17%
Mix2	Firefox	2134	2155	2493	13.56%	2072	2088	2459	15.09%	2010	2102	2511	16.29%
	BBC News	1846	1822	2132	14.54%	1788	1799	2213	18.71%	1604	1642	1889	13.08%
	Facebook	4198	4173	4382	4.77%	4155	4168	4472	6.80%	4124	4132	4479	7.75%
	Temple Run	1664	1692	1997	15.27%	1658	1612	1831	11.96%	1452	1564	1784	12.33%
	Amazon	909	998	1310	23.82%	852	941	1092	13.83%	818	911	1154	21.06%

As shown in Table III, the average size of news applications, which is around 15 MB, is more than 3× smaller than that of browser applications, which is around 50 MB. Moreover, news applications require less memory space than browser application. The smaller application size and moderate memory consumption of news applications lead to less swap-outs and swap-ins. Therefore, he switching speed of news applications of DR. Swap only exhibit small improvement over that of NAND flash backed swapping.

As shown in Fig. 8, DRAM backed swapping also achieves faster switching speed than NAND flash backed swapping. However, as shown in Fig. 7, the energy consumption of DRAM backed swapping is about 3× higher than that of DR. Swap. According to the energy consumption analysis and application switching delay evaluation, DR. Swap can achieve near DRAM backed swapping performance while consumes 3× less energy than DRAM backed swapping. Therefore, DR. Swap can reduce the energy consumption and improve performance of smartphones.

VI. RELATED WORK

In this section, we discuss the related work of this paper from the following three aspects: 1) flash-based swapping system; 2) energy reduction for smartphones; and 3) hybrid PCM/DRAM main memory.

A. Flash-Based Swapping System

There are several newly proposed flash-based swapping systems. Jung *et al.* [34] designed and implemented flash-aware swap system, which is a raw flash memory-based swapping system without using a flash translation layer. FlashVM [35] is another flash backed swapping, which integrates flash memory with virtual memory and provides better garbage collection by batching writes. SSDAlloc [36] is an solid state drive (SSD)/DRAM hybrid system which extends DRAM with SSD and allows programmers to treat SSD as DRAM. Recently, Kim *et al.* [25] evaluated the impact of sub-optimal NAND flash-based storage in smartphones and report that storage plays a significant role in application performance. To eliminate the performance gap between main memory and flash-based swap area, we replace flash memory with emerging byte-addressable NVM and adopt it as swap area, swap in/out is through memory interface.

B. Energy Reduction for Smartphones

Reducing energy consumption in smartphones has been a focus in the research community. Wang *et al.* [37] used profile-based battery traces which are easy to acquire from any smartphones to estimate the power consumption of mobile applications. To better understand energy consumption in smartphones, Perrucci *et al.* [4] measured and compared the

energy consumed by different components in mobile devices, it allows the reader to understand what the energy hungry parts of a mobile device are and provides guidelines to design future mobile protocols and applications. Chen *et al.* [38] conducted comprehensive power measurements of the smartphone radio components on some representative applications. Li and John [39] profiled run-time OS characteristics and propose a routine based OS-aware microprocessor resource adaptation mechanism to reduce run-time OS power. Chen *et al.* [40] analyzed the energy consumption of active matrix organic light emitting diode screen. Nixon *et al.* [41] focussed on reduce the mobile graphics processing unit (GPU) power consumption through dynamic resolution and frame rate scaling. Shen *et al.* [42] 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.* [43] focussed on the optimization of the power delivery network (PDN) in smartphones. In this paper, we reduce the smartphone energy consumption by replacing part of DRAM with emerging NVM and using it as swap area.

C. Hybrid PCM/DRAM Main Memory

Due to its low standby power, high density, and byte addressability, PCM is considered as a promising DRAM alternative [17], [44]. In [45] and [46], a hybrid PCM/DRAM main memory system is proposed where pages can be transferred between PCM and DRAM for saving energy and improving PCM lifetime. Qureshi *et al.* [47] proposed a hybrid main memory organization with on-chip DRAM cache and PCM main memory, in which DRAM is not visible to OS and managed by the dedicated memory controller. Different from the architecture proposed in [47], in our architecture, DRAM is served as the main memory and NVM is used as the swap area, and both DRAM and NVM are visible to OS. In [48], a flat structure of hybrid DRAM/PCM for single-chip CPU/GPU is proposed considering the tight requirements of low latency from CPU and the relative tolerance to long latency from GPU. Park *et al.* [49] addressed the power management of hybrid DRAM/PCM main memory and utilize PCM to reduce the DRAM refresh energy. Khouzani *et al.* [50] aimed to improve the performance and lifetime of PCM/DRAM main memory through a proactive page allocation algorithm. Though we have the same or similar hardware architecture compared to these hybrid memories, the software or the management strategy is totally different. In the hybrid approaches, PCM is treated as main memory and the space is managed by the OS memory management component for allocation and deallocation. In our architecture, we adopt PCM as the swap area and the space is managed by the swap subsystem, which in charge of allocation swap space to store inactive pages swapped out from DRAM.

VII. Conclusion

Reducing energy consumed by DRAM is critical for saving battery lifetime in smartphones. Emerging NVMs energy-efficiency and byte-addressability make it an attractive swapping solution for swapping in smartphones. In this paper, we have proposed DR. Swap, an energy-efficient IMP architecture to reduce energy consumption in smartphones. We readopt swapping in smartphones by replacing part of the DRAM with NVM, and using it as a swap area. We also modified the Android Linux kernel to replace the traditional swap subsystem with our NVM swap subsystem. To avoid the unnecessary memory copy operations, we propose the DR optimization, instead of copying the request page from NVM to DRAM, we directly set up an RO mapping for the requested page and then process can read it directly. With DR, we guarantees zero memory copy operations for RO pages in swap area. To understanding the energy behavior of swapping, we present a fine-gained energy model to analyze the energy consumption of different swapping scheme.

DR. Swap does not target at any specific NVM products and any NVM can be adopted in our architecture. We implement the proposed techniques into Android's Linux kernel and use PCM as a case study. Experimental results based on Google nexus 5 show that DR optimization can reduce around 50%–75% swap-ins, which means a great number of memory copy operations are reduced. Compared to DRAM backed swapping, DR. Swap can reduce more than 60% energy consumption. Compared to NAND flash backed swapping, DR. Swap can reduce the application switching delay around 10% on average. We therefore conclude that swapping with the help of emerging byte-address NVM should be readopted to build both energy efficient and high-performance smartphones. We expect this paper can serve as a first step toward the full exploration of NVM-based swapping in smartphones.

References

[1] K. Zhong *et al.*, "DR. Swap: Energy-efficient paging for smartphones," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, San Francisco, CA, USA, 2014, pp. 81–86.

[2] A. Carroll and G. Heiser, "The systems hacker's guide to the galaxy energy usage in a modern smartphone," in *Proc. 4th Asia-Pac. Workshop Syst. (APSys)*, Singapore, 2013, Art. ID 5.

[3] R. Duan, M. Bi, and C. Gniady, "Exploring memory energy optimizations in smartphones," in *Proc. Int. Green Comput. Conf. Workshops (IGCC)*, Orlando, FL, USA, 2011, pp. 1–8.

[4] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer, "Survey on energy consumption entities on the smartphone platform," in *Proc. IEEE 73rd Veh. Technol. Conf. (VTC)*, Budapest, Hungary, 2011, pp. 1–6.

[5] A. Agrawal. (2010). *Trends in Wireless Communications*. [Online]. Available: http://www.ieeeinfocom.org/2010/docs/Infocom2010_keynote.pdf

[6] H. Huang, P. Pillai, and K. G. Shin, "Design and implementation of power-aware virtual memory," in *Proc. Annu. Conf. USENIX Annu. Tech. Conf. (ATEC)*, San Antonio, TX, USA, 2003, pp. 57–70.

[7] M. Lee, E. Seo, J. Lee, and J.-S. Kim, "PABC: Power-aware buffer cache management for low power consumption," *IEEE Trans. Comput.*, vol. 56, no. 4, pp. 488–501, Apr. 2007.

[8] S. Eilert, M. Leinwander, and G. Crisenza, "Phase change memory: A new memory enables new memory usage models," in *Proc. IEEE Int. Memory Workshop (IMW)*, Monterey, CA, USA, 2009, pp. 1–2.

[9] J. Park, H. Han, and S. Cho, "Extending main memory with flash—The optimized SWAP approach," in *Proc. Non-Volatile Memories Workshop (NVMW)*, San Diego, CA, USA, 2014.

[10] H.-S. P. Wong *et al.*, "Phase change memory," *Proc. IEEE*, vol. 98, no. 12, pp. 2201–2227, Dec. 2010.

[11] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, May 2008.

[12] C. J. Xue, "Emerging non-volatile memories: Opportunities and challenges," in *Proc. 9th Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Taipei, Taiwan, 2011, pp. 325–334.

[13] D. Liu *et al.*, "Application-specific wear leveling for extending lifetime of phase change memory in embedded systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 10, pp. 1450–1462, Oct. 2014.

[14] S. Cho and H. Lee, "Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarch. (MICRO)*, New York, NY, USA, 2009, pp. 347–357.

[15] 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 Trans. Embedded Comput. Syst.*, vol. 12, no. 3, 2013, Art. ID 77.

[16] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, "Improving write operations in MLC phase change memory," in *Proc. IEEE 18th Int. Symp. High Perform. Comput. Archit. (HPCA)*, New Orleans, LA, USA, 2012, pp. 1–10.

[17] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable DRAM alternative," in *Proc. 36th Annu. Int. Symp. Comput. Archit. (ISCA)*, Austin, TX, USA, 2009, pp. 2–13.

[18] 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," in *Proc. IEEE 32nd Real-Time Syst. Symp. (RTSS)*, Vienna, Austria, 2011, pp. 357–366.

[19] M. K. Qureshi *et al.*, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarch. (MICRO)*, New York, NY, USA, 2009, pp. 14–23.

[20] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Proc. 36th Annu. Int. Symp. Comput. Archit. (ISCA)*, Austin, TX, USA, 2009, pp. 14–23.

[21] M. Hosomi *et al.*, "A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-RAM," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Washington, DC, USA, 2005, pp. 459–462.

[22] F. Oboril, R. Bishnoi, M. Ebrahimi, and M. B. Tahoori, "Evaluation of hybrid memory technologies using SOT-MRAM for on-chip cache hierarchy," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 3, pp. 367–380, Mar. 2015.

[23] W. Wen, Y. Zhang, Y. Chen, Y. Wang, and Y. Xie, "PS3-RAM: A fast portable and scalable statistical STT-RAM reliability/energy analysis method," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 11, pp. 1644–1656, Nov. 2014.

[24] A. Jog *et al.*, "Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs," in *Proc. 49th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2012, pp. 243–252.

[25] H. Kim, N. Agrawal, and C. Ungureanu, "Revisiting storage for smartphones," *ACM Trans. Storage*, vol. 8, no. 4, 2012, Art. ID 14.

[26] J. Cooke, *Flash Memory Technology Direction*. Boise, ID, USA: Micron Technol., 2007.

[27] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf. (USENIX ATC)*, Boston, MA, USA, 2010, p. 21.

[28] Micron Inc. (2001). *TN-41-01: Calculating Memory System Power for DDR3*. [Online]. Available: http://www.micron.com/~/media/documents/products/technical-note/dram/tn41_01ddr3_power.pdf

[29] L. Wehmeyer and P. Marwedel, *Fast, Efficient and Predictable Memory Accesses*. Dordrecht, The Netherlands: Springer, 2006.

[30] Micron Inc. (2014). *Automotive LPDDR2 SDRAM*. [Online]. Available: http://www.micron.com/~/media/documents/products/data-sheet/dram/mobile-dram/low-power-dram/lpddr2/2gb_automotive_lpddr2_u89n.pdf

[31] *SanDisk iNAND e.MMC 4.41 1/F—Data Sheet*, SanDisk Corporation, Milpitas, CA, USA, 2011.

[32] *LPDDR2-PCM Phase Change Memory 45nm Discrete*, Micron Inc., Boise, ID, USA, 2011.

[33] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.

[34] D. Jung, J. S. Kim, S. Y. Park, J. U. Kang, and J. Lee, "FASS: A flash-aware swap system," in *Proc. Int. Workshop Softw. Support Portable (IWSSPS)*, San Francisco, CA, USA, 2005.

[35] M. Saxena and M. M. Swift, "FlashVM: Revisiting the virtual memory hierarchy," in *Proc. Workshop Hot Topics Oper. Syst. (HotOS)*, 2009, p. 13.

[36] A. Badam and V. S. Pai, "SSDAlloc: Hybrid SSD/RAM memory management made easy," in *Proc. Symp. Netw. Syst. Design Implement. (USENIX NSDI)*, Boston, MA, USA, 2011, pp. 211–224.

[37] C. Wang, F. Yan, Y. Guo, and X. Chen, "Power estimation for mobile applications with profile-driven battery traces," in *Proc. IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Beijing, China, 2013, pp. 120–125.

[38] X. Chen, Y. Chen, M. Dong, and C. Zhang, "Demystifying energy usage in smartphones," in *Proc. 51st Annu. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2014, pp. 1–5.

[39] T. Li and L. K. John, "Operating system power minimization through run-time processor resource adaptation," *Microprocess. Microsyst.*, vol. 30, no. 4, pp. 189–198, 2006.

[40] X. Chen, Y. Chen, Z. Ma, and F. C. A. Fernandes, "How is energy consumed in smartphone display applications?" in *Proc. 14th Workshop Mobile Comput. Syst. Appl. (HotMobile)*, 2013, Art. ID 3.

[41] K. W. Nixon, X. Chen, H. Zhou, Y. Liu, and Y. Chen, "Mobile GPU power consumption reduction via dynamic resolution and frame rate scaling," in *Proc. 6th Workshop Power-Aware Comput. Syst. (HotPower)*, Broomfield, CO, USA, 2014, p. 5.

[42] H. Shen, M. Kumar, S. K. Das, and Z. Wang, "Energy-efficient data caching and prefetching for mobile devices based on utility," *Mobile Netw. Appl.*, vol. 10, no. 4, pp. 475–486, 2005.

[43] W. Lee, Y. Wang, D. Shin, N. Chang, and M. Pedram, "Optimizing the power delivery network in a smartphone platform," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 1, pp. 36–49, Jan. 2014.

[44] Z. Shao, Y. Liu, Y. Chen, and T. Li, "Utilizing PCM for energy optimization in embedded systems," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Amherst, MA, USA, 2012, pp. 398–403.

[45] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A hybrid PRAM and DRAM main memory system," in *Proc. 46th Annu. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2009, pp. 664–669.

[46] W. Zhang and T. Li, "Exploring phase change memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures," in *Proc. 18th Int. Conf. Parallel Archit. Compilation Tech. (PACT)*, Raleigh, NC, USA, 2009, pp. 101–112.

[47] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Proc. 36th Annu. Int. Symp. Comput. Archit. (ISCA)*, Austin, TX, USA, 2009, pp. 24–33.

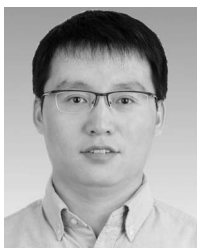
[48] D. Kim *et al.*, "Hybrid DRAM/PRAM-based main memory for single-chip CPU/GPU," in *Proc. 49th Annu. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2012, pp. 888–896.

[49] H. Park, S. Yoo, and S. Lee, "Power management of hybrid DRAM/PRAM-based main memory," in *Proc. 48th Design Autom. Conf. (DAC)*, New York, NY, USA, 2011, pp. 59–64.

[50] H. A. Khouzani, C. Yang, and J. Hu, "Improving performance and lifetime of dram-PCM hybrid main memory through a proactive page allocation strategy," in *Proc. 20th Asia South Pac. Design Autom. Conf. (ASP-DAC)*, Chiba, Japan, 2015, pp. 508–513.

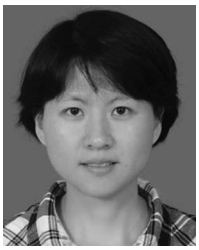
Kan Zhong received the B.Sc. degree in computer science from Chongqing University, Chongqing, China, in 2013, where he is currently pursuing the Ph.D. degree.

His current research interests include mobile computing, embedded system, and nonvolatile memory.



Duo Liu received the B.E. degree in computer science from the Southwest University of Science and Technology, Sichuan, China, in 2003, the M.E. degree from the Department of Computer Science, University of Science and Technology of China, Hefei, China, in 2006, and the Ph.D. degree in computer science from the Department of Computing, Hong Kong Polytechnic University, Hong Kong, in 2012.

He is currently an Assistant Professor with the College of Computer Science, Chongqing University, Chongqing, China. His current research interests include emerging memory techniques and embedded systems.



Liang Liang received the B.E. and M.E. degrees from the Southwest University of Science and Technology, Sichuan, China, in 2003 and 2006, respectively, and the Ph.D. degree in communication and information system from the University of Electronic Science and Technology of China, Chengdu, China, in 2012.

She is currently a Lecturer with the College of Communication Engineering, Chongqing University, Chongqing, China. Her current research interests include wireless communication and optimization, green radio, and wireless sensor networks.

Xiao Zhu received the B.Sc. degree in computer science from Chongqing University, Chongqing, China, in 2014, where he is currently pursuing the master's degree.

His current research interests include mobile computing, embedded system, and emerging memory techniques.

Linbo Long received the B.S. degree from the College of Computer Science, Chongqing University, Chongqing, China, in 2011, where he is currently pursuing the Ph.D. degree.

His current research interests include compiler optimization, emerging memory techniques, and embedded systems.

Yi Wang received the B.E. and M.E. degrees in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2005 and 2008, respectively, and the Ph.D. degree in computer science from the Department of Computing, Hong Kong Polytechnic University, Hong Kong, in 2011.

He is currently an Associate Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His current research interests include embedded systems and real-time scheduling for multicore systems.

Edwin Hsing-Mean Sha received the Ph.D. degree from the Department of Computer Science, Princeton University, Princeton, NJ, USA, in 1992.

From 1992 to 2000, he was with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA. Since 2000, he has been a tenured Full Professor with the Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA. Since 2012, he has been the Dean with the College of Computer Science, Chongqing University, Chongqing, China. He has published over 280 research papers in refereed conferences and journals.

Prof. Sha was a recipient of the Teaching Award, the Microsoft Trustworthy Computing Curriculum Award, the NSF CAREER Award, the NSFC Overseas Distinguished Young Scholar Award, the Chang Jiang Honorary Chair Professorship, and the China Thousand Talents Program. He has served as an Editor for many journals, and the Program Committee and the Chair for numerous international conferences.