IPL-P: In-Page Logging with PCRAM *

Kangnyeon Kim
School of Info. & Comm. Engr.
Sungkyunkwan University
Suwon 440-746, Korea
kangnuni@skku.edu

Sang-Won Lee
School of Info. & Comm. Engr.
Sungkyunkwan University
Suwon 440 746, Korea
swlee@skku.edu

Bongki Moon
Dept. of Computer Science
University of Arizona
Tucson, AZ 85721, U.S.A
bkmoon@cs.arizona.edu

Chanik Park
Samsung Electronics Co., Ltd.
San #16 Banwol-Ri
ci.park@samsung.com

Joo-Young Hwang
Samsung Electronics Co., Ltd.
San #16 Banwol-Ri
jooyoung.hwang@samsung.com

ABSTRACT

A great deal of research has been done on solid-state storage media such as flash memory and non-volatile memory in the past few years. While NAND-type flash memory is now considered a top alternative to magnetic disk drives, different types of non-volatile memory have also begun to appear in the market recently. Although some advocates of storage class memory (SCM) predicted that flash memory would give way to SCM in the very near future, we believe that they will co-exist, complementing each other, for a while until the hurdles in its manufacturing process are lifted and storage class memory becomes commercially competitive in both capacity and price. This demo presents an improved design of In-Page Logging (IPL) by augmenting it with Phase Change RAM (PCRAM) in its log area. IPL is a buffer and storage management strategy that has been proposed for flash memory database systems. Due to the byte-addressability of PCRAM and its faster speed for small reads and writes, the IPL scheme with PCRAM can improve the performance of flash memory database systems even further by storing frequent log records in PCRAM instead of flash memory. We report a few advantages of this new design that will make IPL more suitable for flash memory database systems.

1. INTRODUCTION

Flash memory has been the mainstream of solid state storage media for the past decade and is now considered a top alternative to magnetic disk drives [7]. The NAND-type flash memory is similar to conventional block storage devices in that the unit of a read or write operation is a page of typically 2K or 4K bytes. On the other hand, flash memory is a purely electronic device without any mechanical part, and provides much higher random access speed than magnetic disk drives.

One of the unique characteristics of flash memory is that no data item can be updated by overwriting it in place. In order to update an existing data item stored in flash memory, a time-consuming erase operation must be performed in advance for an entire block of flash memory containing the data item, which is much larger (typically 128 or 256 KBytes) than a page. Dealing with the erase-before-write limitation of flash memory has been one of the major challenges in developing solid state storage devices based on flash memory.

For the past few years, advances in the solid state drive (SSD) technology have made flash memory storage devices as a viable alternative to disk drives for large scale enterprise storage systems. Most enterprise class flash memory SSDs are equipped with parallel channels, a large over-provisioned capacity, and a large on-drive DRAM cache. Combined with advances in the Flash Translation Layer (FTL) technology, this new SSD architecture overcomes huddles in small random write operations and provide significant performance advantages over disk drives [7].

Media	Access time		
	Read	Write	Erase
Disk†	12.7 ms (2KB)	13.7 ms (2KB)	N/A
NAND Flash‡	75 μs (2KB)	250 μs (2KB)	1.5 ms (128KB)
PCRAM¶	206 ns (32B)	7.1 μs (32B)	N/A

†Disk: Seagate Barracuda 7200.7 ST380011A;
‡NAND Flash: Samsung K9F8G08U0M 16Gbits SLC NAND;
¶PCRAM: Samsung 90nm 512Mb PRAM Prototype [5]

Table 1: Disk, NAND Flash and PCRAM.

Non-charge-based non-volatile memory technologies have recently been under active development and commercialization by industry leading manufacturers [3, 5, 9]. Unlike charge storage devices such as flash memory, these non-volatile memory technologies provide memory states without

*This research was supported in part by MKE, Korea under ITRC NIPA-2011-(C1090-1121-0008) and the NRF funded by the Korea goverment(MEST)(No. 2010-0025649),(No. 2010-0026511).

Proceedings of the VLDB Endowment, Vol. 4, No. 12

electric charges [3]. Among those non-charge-based memory technologies, phase-change RAM (PCRAM) is considered a leading candidate for the next generation byte-addressable non-volatile memory.

PCRAM can be programmed in place without having to erase the previous state. Although PCRAM has a limited number of programming cycles due to repeated heat stress applied to the phase change material, it is considered to have greater write endurance than flash memory by a few orders of magnitude [3, 4]. Furthermore, in contrast to NAND type flash memory, PCRAM need not operate in page mode and allows random accesses to individual bytes like DRAM does.

It is reported in the literature that the read and write latencies of PCRAM are approximately an order of magnitude greater than those of DRAM [4]. As is shown in Table 1, however, contemporary PCRAM products do not deliver the promised performance as yet particularly for write operations. While PCRAM takes only about 206 *ns* to read 32 bytes, it takes as long as 7.1 *us* to write 32 bytes. A similar disparity in read and write speeds is observed in other PCRAM products as well [9].

In-Page Logging with PCRAM

Given the unique characteristics of PCRAM, we expect that PCRAM and flash memory will co-exist complementing each other for a while until the manufacturing cost of PCRAM is reduced to a level comparable to that of flash memory. For this reason, we are interested in the potential of PCRAM that would make the In-Page Logging (IPL) scheme more efficient, which is suggested as a new storage model for flash-based database systems [6]. Specifically, PCRAM allows for smaller units of writes, and the write time is approximately proportional to the amount of data to transfer.[1] These traits make PCRAM an excellent candidate for a storage medium for log data managed by the IPL scheme.

In this demo, we revisit the In-Page Logging (IPL) scheme that has been proposed as a new storage model for flash-based database systems [6] and elaborate how the IPL scheme can accelerate database systems further by utilizing PCRAM for storing log records. The byte-addressability of PCRAM allows for finer-grained writing of log records than flash memory, thereby reducing the time and space overhead for both page read and page write operations. The results of preliminary performance evaluation will be presented to illustrate the potential benefit of IPL adapted to hybrid storage systems equipped with both flash memory and PCRAM.

2. SYSTEM OVERVIEW

For a hybrid storage device equipped with PCRAM as well as flash memory, IPL can boost its performance by taking advantage of higher access speed and byte-addressability of PCRAM for writing and reading small log records. In this section, we present the new implementation of IPL that utilizes PCRAM to improve its performance further.

2.1 Granularity of Log Writes

When a dirty page is about to be evicted from the buffer pool, a conventional buffer manager would write the entire dirty page back to a secondary storage device, even though the actual change amounts to only a small fraction of the page. The key idea of IPL, as depicted in Figure 1 is that only the changes made to a dirty page be written to a flash memory storage device in the form of log records without writing the dirty page itself [6].

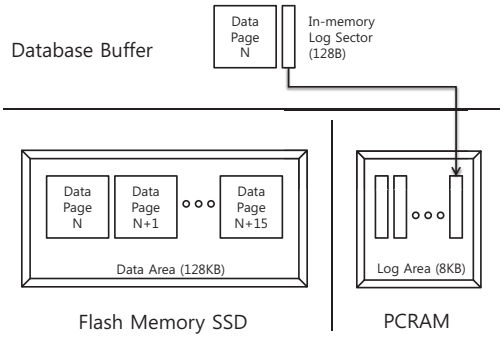


Figure 1: IPL with PCRAM.

In the OLTP applications, most page frames become dirty in the buffer pool by small changes. When a dirty page is to be evicted, the amount of log records that need to be propagated for the page is usually very small. (It was in the range of 50 to 150 bytes in the TPC-C workloads we had observed [8].) This implies that write reduction could be furthered considerably if log records were written in a granularity smaller than a 512B sector. Obviously such a fine-grained writing is not feasible with any contemporary flash memory device, but it can be done very efficiently with byte-addressable non-volatile memory like PCRAM.

2.2 *IPL-P* Prototype for PCRAM Logging

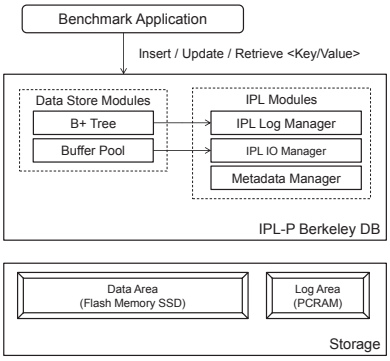
Under the new design of IPL with PCRAM, called *IPL-P*, regular data pages are stored in flash memory, while the corresponding log records are stored in PCRAM. Similarly to the flash-only IPL, when the PCRAM log area allocated for a certain flash memory block is used up, *IPL-P* applies all the log records stored in the log area to the corresponding data pages, stores the updated pages in a clean flash block, and the PCRAM log area is marked as empty. (In the current implementation of *IPL-P*, the task of flash memory management such as cleaning and overwriting blocks is left to the FTL module of the flash memory SSD.) The merge operation of *IPL-P* works essentially the same way as IPL [6] except that merge operations are triggered when a log area in PCRAM becomes full.

Finer-grained log writes with PCRAM allow *IPL-P* to utilize the log area more efficiently, which in turn leads to much less frequent merge operations. That also reduces the average latency of writing or reading log records considerably. Using the hardware platform to be used for demonstration (described in Section 2.4), we observed that writing 128 byte data to PCRAM was more than a factor of two faster than partial programming a 512 byte sector of flash memory. As for reading 128 or 512 bytes, respectively, PCRAM was at least an order of magnitude faster than flash memory.

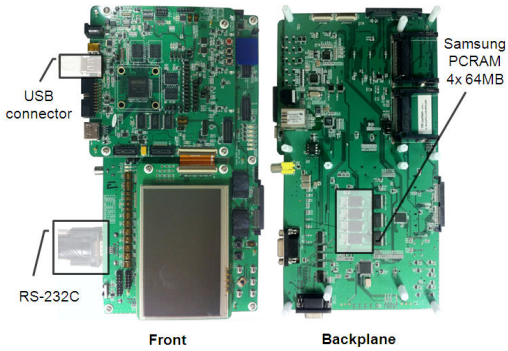
2.3 Implementation Details

For prototyping *IPL-P*, we modified the open source Berkeley DB system (version 4.7.25) [1]. As shown in Figure 2(a),

[1]For both read and write, the access time of PCRAM is not entirely proportional to the number of bytes to access, as there is an initial latency of about 78 *ns* for each operation [5].



(a) Berkeley DB with *IPL-P*



(b) Demonstration platform

Figure 2: System Architecture and Hardware Platform.

we added three IPL-specific modules including the IPL log manager, the IPL IO manager, and the metadata manager. We also modified the data store modules in the Berkeley DB for interactions with the IPL modules.

Let us briefly explain how *IPL-P* works using these additional functions. When a change is requested for a B^+-tree, corresponding functions such as insert, delete and update capture the change in the form of physiological log record and passes it to the IPL log manager. The log record stays in the in-memory log area until the in-memory log area becomes full or its data page is evicted from the buffer pool. In either case, the in-memory log record is flushed persistently to a log area in PCRAM by the IPL IO manager.

When a data page needs to be fetched to memory, the buffer pool manager first reads the old copy of the page from flash memory SSD, and then passes it to the IPL IO manager, which recomputes the most recent copy of the page by applying the corresponding log records. Another role of the IPL IO manager is to merge the data pages in flash memory SSD and the log data in PCRAM, when the log area becomes full. The metadata manager maintains the mapping information between the data blocks in flash memory SSD and the corresponding log areas in PCRAM.

2.4 Hardware Specifications

We will run the Berkeley DB server augmented with *IPL-P* on a commercial embedded development board [2]. The demonstration board (shown in Figure 2(b)) is equipped with an ARM Cortex-A8 core (S5PV210 model from Samsung) and 690MB DRAM. This hardware platform we will use for demonstration is a development kit that is mainly used for evaluating the basic operations of an S5PV210 chipset (used in Samsung smartphones) and for developing smart phone applications. The board was originally equipped with NAND flash memory chips as internal storage, but the NAND chips were replaced with PCRAM chips [10] so that we could implement *IPL-P* on the board. We also replaced a built-in NAND controller with a PCRAM controller. The board is operated by Linux kernel 2.6.29.

Even though the PCRAM chips are byte-addressable, the board itself is designed to transfer data between DRAM and PCRAM chips at the granularity of 128 bytes. Because of the intrinsic hardware limitation, the granularity of PCRAM read and write operations was also set to 128 bytes.

We attached an Intel flash memory SSD X25-M to the development board through a USB 2.0 interface. Because of the bandwidth limited by the USB interface, the random read performance of the flash memory SSD was substantially limited to about 1600 IOPS for 8KB pages. (This throughput is approximately a factor of six lower than an SATA interface can deliver). In contrast, there was no significant performance degradation in the case of random write operations, because the write throughput was already sufficiently lower than that of read not to be limited by the USB interface.

Table 2 shows the average access times of read and write operations that were actually measured from the PCRAM and the flash memory SSD installed on the board. Again, because of the hardware limitations, the read and write times of the multi-channel SSD were even worse than those of a single NAND chip (shown in Table 1). The access speeds of the PCRAM on the board were also much lower than those given in Table 1, and this is due to the deliberate reduction of access speeds for high yield of chips during mass production.

Media	Access time	
	Read	Write
Flash SSD†	620 μs (8KB)	1.6 ms (8KB)
PCRAM‡	6 μs (128B)	192 μs (128B)

†Flash SSD : Intel X25M
‡PCRAM: Samsung KPS1315EZA 512Mbits PRAM [10]

Table 2: Access Speed: Flash SSD vs PCRAM.

3. DEMONSTRATION DETAILS

The goal of this demonstration is to provide a proof-of-concept for *IPL-P* [8] using a real hardware platform. It will be shown that the *IPL-P* approach, as a hybrid scheme, can outperform either a flash memory only or a PCRAM-only approach by taking advantage of both storage media.

The demonstration will be set up with a host system and two development boards. The *IPL-P* approach will be compared with two other approaches: one that runs Berkeley DB only on the flash memory SSD with *IPL-P* disabled (*flash-only* approach) and the other that runs Berkeley DB, again with *IPL-P* disabled, only on PCRAM (*PCRAM-only* approach).

For performance evaluation, three simple synthetic benchmark programs will be used, each inserting, updating, or

retrieving a million records. Each program will be running on the development board. The host system will be used to initiate the benchmark programs and monitor the progress on the boards. The real-time performance measurements from the boards will be passed back to the host to see how many operations are executed in every second. The communication between the host system and the boards will be done bidirectionally over an RS-232C cable.

In order to demonstrate the online progress of the benchmark programs running on the boards, we have implemented a GUI-based performance monitor using the Labview development toolkit. As shown in Figure 3, the performance monitor displays the current transaction throughput in the meter gauges as well as progress bars.

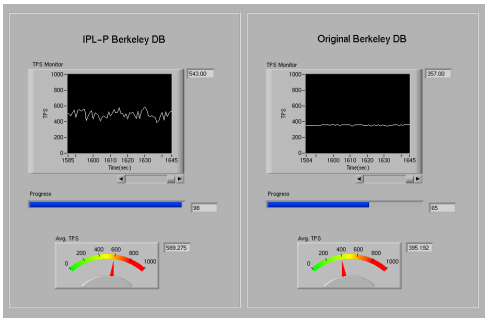


Figure 3: Real-time Performance Monitor: GUI.

Preliminary Performance Evaluation

As is described above, synthetic benchmark programs were used to insert, update, and retrieve one million records. One million key-value records with unique keys were inserted in random order, and searches were also done in random order. In order to simulate the skewed updates common in OLTP workloads, update requests were generated using a Zipfian distribution such that 80% of updates were directed to 20% of database pages.

The default page size of Berkeley DB was 8 KBytes and the length of a key-value record was 30 bytes (a 4 byte key and a 26 byte value). The database size was approximately 60MB, and the RAM buffer cache was limited to one MB to amplify IO effects with the small database. When either flash memory SSD or PCRAM was used as stable storage, it was accessed in O_DIRECT mode so as to minimize the interference from data caching by the operating system. For the same reason, the cache on the flash memory SSD was turned off.

A few key observations can be made from the performance results shown in Figure 4. First, *IPL-P* outperformed the other two approaches considerably for write intensive workloads by minimizing the absolute volume of writes. Recall that, as explained in Section 2.4, the read throughput of the flash memory SSD on the board was severely limited by the slow USB interface. This elevated unduly the read time portion of total elapsed time, and dampened the effect of flash write reduction by *IPL-P* on the overall performance. With a faster interface such as SATA, we expect that the performance gain by *IPL-P* will be more pronounced. Second, while the PCRAM-only approach was the best for read only workload, its performance was significantly worse than the others for write intensive workloads. This was because,

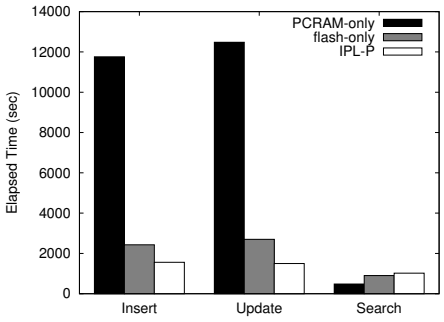


Figure 4: *IPL-P* Performance Evaluation.

for 8KB page writes, the throughput of the PCRAM on the board was about seven times worse than that of the flash memory SSD. Third, *IPL-P* was the worst among the three approaches for read only workload because of the extra overhead involved in recomputation of the current version of pages. However, the performance gap between *IPL-P* and the flash-only approach, both of which store the database in the flash memory SSD, was not significant.

In order to develop the next-generation database systems utilizing non-volatile memory devices effectively, it is essential to look for new ideas for storage and database buffer management. We believe that *IPL-P* provides a model case of hybrid storage and buffer management design based on flash memory and PCRAM.

4. REFERENCES

[1] Oracle Berkeley DB. http://www.oracle.com/technetwork/database/berkeleydb/.

[2] SMDKV210: Samsung S5PV210 Reference Board. http://www.yicsystem.com/contents/korea/smdkv210.htm.

[3] International Technology Roadmap for Semiconductors. Process Integration, Devices, and Structures. 2009 Edition.

[4] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. Architecting Phase Change Memory as a Scalable DRAM Alternative. In *Proceedings of International Symposium on Computer Architecture (ISCA)*, pages 2–13, 2009.

[5] K.-J. Lee et al. A 90 nm 1.8 V 512 Mb Diode-Switch PRAM With 266 MB/s Read Throughput. *IEEE Journal of Solid-State Circuits*, 43(1):150–162, 2008.

[6] S.-W. Lee and B. Moon. Design of Flash-Based DBMS: An In-Page Logging Approach. In *Proceedings of ACM SIGMOD*, pages 55–66, 2007.

[7] S.-W. Lee, B. Moon, and C. Park. Advances in Flash Memory SSD Technology for Enterprise Database Applications. In *Proceedings of ACM SIGMOD*, pages 863–870, 2009.

[8] S.-W. Lee, B. Moon, C. Park, J.-Y. Hwang, and K. Kim. Accelerating In-Page Logging with Non-Volatile Memory. *Bulletin of the Technical Committee on Data Engineering*, 33(4):41–47, 2010.

[9] Numonyx. Omneo P8P 128-Mbit Parallel Phase Change Memory. Data Sheet 316144-06, Apr 2010.

[10] Samsung Electronics. 512Mb A-die SLC PRAM (KPS1315EZA). Data sheet(rev. 0.1), Nov. 2009.