What makes a good OS page replacement scheme for Smart-Phones?

Hyojun Kim, Moonkyung Ryu, Umakishore Ramachandran
College of Computing, Georgia Institute of Technology
{hyojun.kim, rumuru, rama}@cc.gatech.edu

A significant percentage of the population in the developed world rely on smart-phones as their primary (and often the only) source of information access. Smart-phones are starting to have comparable power to regular computers; *e.g.*, the latest smart-phone has a dual core 1 GHz processor as well as 1 Gbytes of main memory capacity already. On the other hand, the technology used for storage on smart-phones lags significantly behind that used on regular computers. Inexpensive flash storage is the norm for smart-phone because of the limitations of size, cost, and power consumption. Such low-end flash storage exhibits very different performance characteristics relative to the traditional hard drive; plus the current smart-phone operating systems are not engineered to support flash storage adequately. Consequently, flash storage is often pointed to as the main source of performance bottleneck on smart-phones. While high-end flash storages are available and used in enterprise class machines, adoption of such storage for smart-phones is infeasible for reasons of cost, size, and energy consumption. Therefore, we argue that operating system level software support is critically needed for low-end flash storage to achieve high performance on smart-phones.

Flash storage is based on semiconductor technology, and hence shows very different performance characteristics when compared to the traditional magnetic disk. A number of studies have reported on the special performance characteristics of flash storage [1, 2]. It is well known that flash storage devices show a relatively low write-throughput for small, scattered (random) requests and a higher throughput for large, sequential write requests. At the same time, they are insensitive to the order of read requests, showing almost unchanging performance for sequential and random read requests. We ourselves have performed simple measurements to identify these differences in performance, and Figure 1 compares the measured read and write throughput of a hard disk drive (HDD) and a microSDHC card.

On an HDD, both read and write throughput are highly influenced by the sequence of the requests. In contrast, the read throughput of microSDHC card is not much influenced by request ordering. For the write requests, microSDHC shows uniformly lower throughput than the HDD. What is interesting to observe is the disparity in performance for sequential versus random writes on the

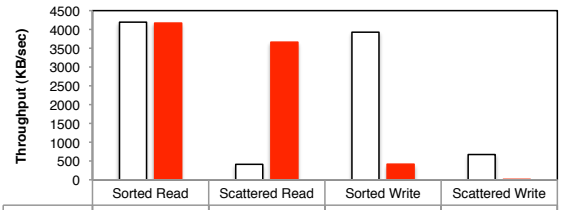

Figure 1: Comparison of an HDD (2.5" 5400 RPM HDD) vs. microSDHC (16GB, MLC) performances

flash storage in comparison to the HDD. On the HDD, this disparity is only a factor of 5.8 (3926 Kbytes/sec. for sequential versus 672 Kbytes/sec. for random); while on the flash storage this disparity is a factor of 33 (416.2 Kbytes/sec. for sequential versus 12.4 Kbytes/sec. for random). This result suggests that write request ordering is extremely important on flash storage, and proper write reordering can effectively enhance the performance of flash storage.

Even though different level solutions have been proposed related to the write request ordering, surprisingly, less attention has been given to write ordering at the level of the operating system page replacement schemes. Arguably, this is a huge missed opportunity since write requests are generated by the virtual memory system when dirty pages are chosen as victims by the page replacement algorithm. DULO [4] is a page replacement algorithm that exploits both temporal and spatial localities, but it only cares about request size, not write request ordering. LRU-WSR (Least Recently Used - Write Sequence Reordering) [5] is a page replacement scheme for flash storage; however, its goal is to reduce the number of write requests sent to the storage device by protecting (as far as possible) dirty pages from being chosen as victims. It has no mechanism for write request ordering when dirty pages *are* chosen to be written out to the storage device.

In this study, we propose to use new, write ordering aware page replacement schemes. Our Sorted-Clock algorithm builds on a prior art, namely, Wise Ordering for Writes (WOW) [3], which was proposed as a write cache management scheme on enterprise servers with RAID

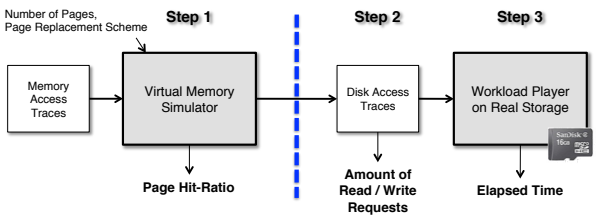

Figure 2: The Evaluation Framework for Page Replacement Schemes

storage. WOW combines the idea of Circular-SCAN to exploit spatial locality and Clock to exploit temporal locality. Despite the fact that WOW was proposed for the RAID architecture, the core idea of paying attention to request ordering in WOW is well matched with the performance characteristics of flash storage. In addition to ordering writes, reducing the number of writes to the storage device is important as well. While reducing the number of write requests may not impact performance as much (this is attested by our evaluation results), it can have a huge impact on the longevity of the flash storage (which is hard to quantify in this study). Therefore, the second algorithm we propose, Sorted-Clock-WSR, borrows the idea of preferential treatment for dirty pages from LRU-WSR, in addition to ordering writes.

To quantify the performance of page replacement schemes on real storage devices, we have built a simple but powerful framework (see Figure 2), which consists of two components: *virtual memory simulator* incorporating different page replacement schemes and a *workload player*. Virtual memory simulator takes memory access traces as its input, and generates storage access requests as its output. Workload player is a standalone program that plays the storage access requests on the actual storage devices and measures the performance (elapsed times and read/write throughput). The framework is especially useful to see the performance effect of write ordering on flash storage. We have compared six page replacement schemes: LRU, Low Inter-reference Recency Set (LIRS), Clock, LRU-WSR, Sorted-Clock, and Sorted-Clock-WSR.

Figure 3 shows the results with the traces of the parser and mcf programs. We show elapsed time with page hit-ratio, and we make two important observations from our evaluation results. First, focusing only on hit-ratio as a figure of merit to evaluate a page replacement scheme can lead to wrong conclusions on flash storage. Second, write ordering aware page replacement schemes perform much better than the other schemes that do not pay attention to this important attribute for flash storage. Elapsed time for Sorted-Clock and Sorted-Clock-WSR are 3.7 to 7.7 times less than the other schemes.

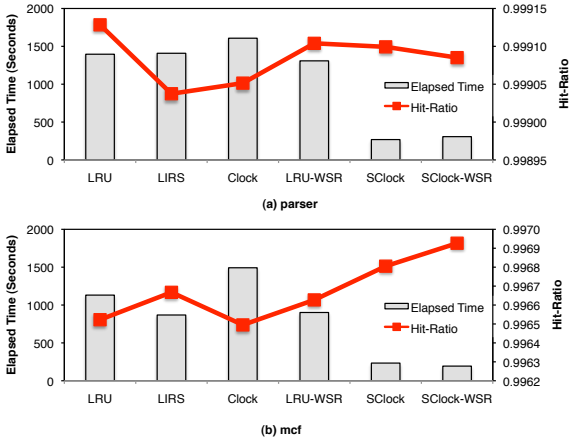

Figure 3: Evaluation results with traces of parser and mcf

Smart-phones are ubiquitous and are starting to serve as the primary source of information access and online services for a significant segment of the user community. Smart-phones usually incorporate low-end flash storage to conserve size, cost, and energy. We argue in this study that a good page replacement scheme for smart-phones should pay attention to the write request ordering (while replacing dirty pages) in addition to page hit-ratio. We propose two new page replacement schemes, Sorted-Clock and Sorted-Clock-WSR that respect write request ordering. Using a novel evaluation framework, we show that these two new schemes perform better than schemes that do not respect write request ordering. In particular, using cumulative elapsed time as the figure of merit, we show that these two new schemes outperform the competition by a factor of 3.7 to 7.7. Our future work includes implementation of these schemes into Linux/Android platform.

References

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy. Design tradeoffs for ssd performance. In *USENIX 2008 Annual Technical conf.*, 2008.

[2] F. Chen, D. A. Koufaty, and X. Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *Proc. of the 11th international joint conf. on Measurement and modeling of computer systems*, 2009.

[3] B. S. Gill and D. S. Modha. WOW: wise ordering for writes - combining spatial and temporal locality in non-volatile caches. In *Proc. of the 4th conf. on USENIX conf. on File and Storage Technologies - Volume 4*, 2005.

[4] S. Jiang, X. Ding, F. Chen, E. Tan, and X. Zhang. DULO: an effective buffer cache management scheme to exploit both temporal and spatial locality. In *Proc. of the 4th USENIX conf. on File and Storage Technologies*, 2005.

[5] H. Jung, H. Sim, P. Sungmin, S. Kang, and J. Cha. LRU-WSR: Integration of LRU and Writes Sequence Reordering for Flash Memory. *IEEE Transactions on Consumer Electronics*, 54(3):1215–1223, 2008.