NAND Flash Storage: The Restricted RAM Solution

Jessica McKellar

January 26, 2009

1 Fun With Math

$a^2 + b^2 = c^2$

2 Introduction

This paper describes a NAND flash storage system which meets the user's expectations in a *restricted RAM environment* - one in which the amount of RAM is a) significantly smaller than the size of an erase block and thus insufficient for functioning as swap space when a sector needs to be overwritten, and b) too small to hold a full mapping of sectors to pages, necessitating keeping at least part of a pointer table structure, if one were to exist, in flash memory.

I don't really have anything to quote, here but if I did, "the scheme described in this paper meets these performance demands by maximizing the use of the limited RAM as a way to store particularly frequently updated information without taking an erase penalty." Meanwhile, the generous size of the flash memory is exploited in conjunction with a wear leveling strategy that accounts for keeping frequently updated sector-to-page mappings in flash memory.

3 Data Structures

Define *reserved* flash as the erase blocks currently constituting the pointer table (L2 table). All other erase blocks are available for the storage of user data and thus termed *data-writable*.

Level 1 Table (L1): This table is held within RAM and contains 92 entries; each entry has a pointer to the beginning of a *page map block* the size of an erase block containing pointers to the locations of sector data. The 92 page map blocks together comprise the *level 2 table*. For example, the first entry in the level 1 table is a pointer to the start of page block 0, which is a page map block in the level 2-designated part of flash which contains the addresses for the data for sectors 0 through 43647. The second entry in the level 1 table is a pointer to page block 1, which contains the addresses for sectors 43648 through 87295, and so on up through page block 92.

Pointer Cache: This cache is held within RAM and in fact consumes all remaining unallocated RAM after accounting for the L1 table and 3 pointers. Each entry must encode a sector and its corresponding page, at 3 bytes each. The cache can thus contain up to 6619 6-byte sector-to-page mapping. The pointer cache is empty after initialization.

4 Algorithms

4.1 Pointer Cache Flush

When the pointer cache has 6619 sector-to-page mappings and is thus full, the L2 table in flash is updated to reflect the status of the sector-to-page mapping contained in the cache so that the cache can be cleared. The pointer cache clearing process is as follows:

1. Examine the first pointer in the cache.
2. Collect at the top of the cache all pointers that would be on the same L2 block as that pointer.
3. Read the contents of the first mapping page of the L2 block that needs to receive an updated pointer.
4. Insert the subset of the collected pointers that belong on that page at the appropriate lines in the buffer.
5. Write that buffered page to the corresponding page of the next clean block in front of the frontier pointer.
6. Repeat steps 2-5 until all the collected pointers for that block have been updated (*64 reads, 64 writes total per page*).
7. Remove the pointers that have just been written to their new L2 from the cache.
8. Repeat steps 1-7 until the cache is empty.
9. Move the reserve pointer to the start of the new L2 table location.

4.2 Wear Leveling: Cycling Reserved Blocks

A data-writable block is erased once per cycle of the queue through the flash, namely when it is in the tailing 104 blocks being consolidated during garbage collection. Similarly, because the L2 table blocks are moved to the front of the queue after every pointer cache flush, a data-writable block only functions as an L2 block once per cycle of the queue through the flash. This shuffling results in an even distribution of erases across blocks.

4.3 Garbage Collection

Garbage collection on the tail of the data queue occurs directly after every pointer cache flush. In this implementation, the goals were twofold:

- clear invalid pages in small enough chunks that the process doesn't create unreasonably long halts during data transfer
- keep all of the valid data consolidated within the queue and all of the empty blocks consolidated in front of the queue

4.4 Speed

4.4.1 Reads

Sequential and random reads take the same amount of time because sectors are remapped regardless. Fetching a sector's data requires either:

- 1 read because the pointer to the corresponding page is located in the pointer cache and the page can be sought and read directly, plus 1 write to the meta data of the old page to tag it as invalid.
- 2 reads because the pointer had to be read from a page in the L2 table before the contents could be fetched
- 2 reads because the pointer had to be read from a page in the L2 table before the contents could be fetched, plus 1 write to the meta data of the old page to tag it as invalid.

Thus in the worst case a read request requires 2 reads and a write, for .28 ms

4.4.2 Writes

Sequential and random writes also take the same amount of time because a sector's data will be written to the first free page at the front of the queue and cause an update to the pointer cache regardless of write type.

At worst, 6619 writes happen before the pointer cache becomes full (at best a small set of sectors are overwritten frequently and thus result in updates to the cache rather than new entries). At that point the pointer cache updates must be written to flash, followed by garbage collection.

4.5 Lifetime

As discussed above, the simple L2 shuffle wear-leveling strategy allows for at least 52 percent of the maximum possible usage of 100,000 writes per page. This corresponds to 218103808000 page-writes or 446676 GB of data-writes.

5 Summary

This restricted RAM flash storage solution maximizes the size of the data-writable flash without significant penalty to read and write performance but with the cost of a factor of two in lifetime. Retaining only half of the maximum possible writes before wearing out the system is a reasonable trade-off considering the huge (400,000+ GB) amount of data transfer still permitted.