A Case for Heterogeneous Flash in the Datacenter

Di Wang, Anand Sivasubramaniam, and Bhuvan Urgaonkar
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA
{diw5108, anand, bhuvan}@cse.psu.edu

Abstract
We explore the idea of heterogeneous NAND flash which possesses pages and blocks of multiple sizes. This heterogeneity can then be exploited to accommodate the diversity in data access patterns found in most datacenter storage workloads. We identify various trade-offs offered by such pages and blocks. By characterizing seven real-world I/O traces, we identify metrics that have a bearing on the efficacy as well as design of such a heterogeneous flash. We use the observations of our workload characterization to discuss the pros and cons of different implementation styles, management schemes and implications for datacenters.

I. Introduction
A variety of non-volatile memories (NVM) based on solid-state technologies have come into prominence recently. At the forefront of these developments is the NAND Flash, which has already gained significant commercial acceptance. NAND Flash has been employed in many embedded and consumer devices and is also beginning to replace traditional hard disk drives (HDDs) in other market segments such as many datacenters [18], enterprises [8] and HPC [1]. Some other NVM technologies are also on the horizon, with STT-RAM [4] and PCM [15] being the most promising. These NVMs offer numerous well-known benefits over HDDs - lower access latencies for random requests, smaller form factors, lower power consumption, lack of noise, and higher robustness to vibrations and temperature.

An additional significant benefit offered by these NVMs has, however, been largely ignored. Unlike with HDDs, it is easier to build NVM-based devices possessing heterogeneous granularity of data storage/retrieval. This heterogeneity can then be exploited to accommodate the diversity in data access patterns present within most datacenter storage workloads. Specifically, with HDDs, manufacturers were forced to have a single-sized sector - the unit of data read/write for a disk - due to the mechanical nature of the disk head’s operation. Why should we continue to live with such uniform granularity offered by the storage medium when there is diversity both within and across workloads that can benefit from device heterogeneity in the datacenter? Due to their solid state nature, NVMs are amenable to offering heterogeneous granularity storage/retrieval.

Given that NAND flash is likely to be the only mature and commercially viable NVM in the near future, we focus only on this technology, although some similar concerns may apply to other NVMs as well.

There are two basic units of data manipulation on NAND flash: pages and blocks. Whereas reads/writes are done at the page granularity, erases are done on blocks (consisting of several physically contiguous pages) to amortize the high latency of erase operation over several writes. Page sizes have slowly increased over the past few years, changing from very small 256 bytes to the more desirable 4-8 KB in order to improve read and write throughput. For a logical I/O request of a given size, a larger page size (i) poses lower overheads on the I/O bus and (ii) requires the flash device to issue fewer read/write requests. Therefore, increasing the page size (i.e., reducing the ratio of logical request size to page size) is likely to result in increased device throughput for many workloads. Blocks have also grown in the meanwhile from 8 to 128 pages per block, for two main reasons: (i) larger blocks offer more new writes per erase (i.e., higher erase efficiency), and (ii) larger blocks allow for higher storage density [9]. Although the trend for page and block sizes has been consistently grown bigger due to these benefits, they also have the downside of posing higher fragmentation (both internal and external), which translates to (i) higher wasted space,
(ii) poorer efficiency of garbage collection (GC), and (iii) possibly, even lower lifetime due to the increased write amplification (number of additional writes induced by GC, which in turn require erases) [20].

Figure 1 presents a qualitative description of these pros and cons for a range of page and block sizes. We would like to note that only some of these concerns are similar to those explored in previous work (e.g., deciding optimal block size in file systems or the design of memory managers with superpages [13]), and most of these are entirely novel and unique. As shown, flash devices have moved from region $A$ to the currently prevalent region $B$. Given these trade-offs, we pose the following questions that will be important to consider as we go forward:

- Should we continue to build flash devices that fall in region $B$, or should we consider other regions (e.g., $C - G$ shown in Figure 1)? E.g., in the recently emergent workloads found in MapReduce clusters, the underlying file system (GFS) is designed to issue large/sequential requests (hundreds of KB to even MB) [5], which would clearly benefit from a device in regions $D - F$ (larger pages).
- Should we consider heterogeneous configurations that combine multiple regions on a flash device? E.g., the workload above may continue to have some small requests (e.g., meta-data), which may benefit from the presence of pages/blocks in regions $B, C, G$. This might imply that this workload may benefit from a device that combines pages/blocks from multiple regions.

Clearly, the answers to these questions must carefully weigh the pros and cons of different page/block sizes which depend on a variety of workload characteristics. This paper is our first step towards developing such understanding. We make the following specific research contributions.

- We characterize seven real-world I/O traces identifying metrics (some traditional, others novel) that have a bearing on the need and efficacy of heterogeneous flash.
- We use our study to identify the opportunities that heterogeneous flash can offer. We provide a number of insights related to the design and use of such a device.
- We discuss different implementation styles for realizing a heterogeneous flash and use our analysis to discuss their pros and cons.

Our study opens up a number of interesting directions to pursue for the realization and use of such a device in datacenters.

The rest of the paper is organized as follows. In Section II, we present our workload characterization and its implications for the design and efficacy of heterogeneous flash. In Section III, we discuss directions for future work.

**II. Workload Characterization and Insights**

In this section, we present our workload characterization and insights gained.

A. **Workload Description**

We use seven real-world block-level I/O traces that have been studied in prior literature. Three of these are commercial workloads: financial [19], tpch [21], and cello [16], while the remaining four are traced from servers in datacenters: prxy, rsrch, usr and web [12]. Table I presents information about the size (in number of unique addresses) of these workloads and number of I/O requests within them. Note that ideally we would like workloads hosted on flash SSDs whereas all these are hosted on HDD-based systems. Unfortunately, real-world workloads from SSD-based systems are not yet easily available. We post-process one aspect of these traces to approximate what they might be in a system using flash SSDs, while noting that such transformation is non-trivial in its fullness, and an orthogonal area of research. Assuming that in the presence of large pages/blocks, it would be beneficial if the host system software coalesces requests whenever possible (using its internal DRAM buffer), we transform our traces by considering groups of 64 successive requests and coalescing logically contiguous ones.

B. **Request Size**

Metrics: There are two aspects of request size that are important to us. First is the well-studied request size distribution, that captures the fraction of requests of a certain size. This is an indicator of whether/how
much certain portion of a workload would benefit from page/block size heterogeneity within the device. Second, we consider a related metric, I/O contribution distribution, that captures the fraction of overall I/O traffic (in bytes and not just requests) contributed by requests of a certain size.

### Table 1

<table>
<thead>
<tr>
<th>Workload</th>
<th>Description</th>
<th>Size GB</th>
<th>Requests in million</th>
</tr>
</thead>
<tbody>
<tr>
<td>financial</td>
<td>OLTP</td>
<td>0.50</td>
<td>6.50</td>
</tr>
<tr>
<td>tpch</td>
<td>Trans. process</td>
<td>1.97</td>
<td>1.62</td>
</tr>
<tr>
<td>cello</td>
<td>HP-UX OS</td>
<td>0.46</td>
<td>0.44</td>
</tr>
<tr>
<td>prxy</td>
<td>Web proxy</td>
<td>0.33</td>
<td>2.44</td>
</tr>
<tr>
<td>rsrch</td>
<td>Research projects</td>
<td>0.36</td>
<td>0.66</td>
</tr>
<tr>
<td>usr</td>
<td>User home dirs</td>
<td>0.34</td>
<td>0.16</td>
</tr>
<tr>
<td>web</td>
<td>Web server</td>
<td>1.56</td>
<td>5.33</td>
</tr>
</tbody>
</table>

**Figure 2. Request size and I/O contribution distribution**

**Observations and Insights:** From Figure 2, we find that all workloads contain requests spanning a large range of sizes: from less than 2 KB to more than 1 MB. Figure 2 (a) shows that a majority of requests are in the 2-16 KB range, for which current flash devices (region B) are well-tuned. Therefore, a heterogeneous flash should continue to possess pages/blocks similar to those in existing devices. Figure 2 (b), however, reveals further information that is important: all workloads have significant I/O traffic (bytes) resulting from large-sized requests (these are requests that would need more than one page in a current flash, e.g., of size more than 8 KB.) The highest portion of large requests is found in web where more than 80% of the I/O traffic (bytes) is due to large-sized requests; even cello, the trace with the lowest portion, still has significant (about 40%) I/O traffic (bytes) from large-sized requests. These observations imply that all of these workloads would experience overall throughput benefit from large pages (i.e., those in regions D – F). However, the ideal capacity from such pages and the ideal size of blocks containing these large pages are likely to be different for different workloads. Specifically, workloads with higher sequentiality (e.g., web and tpch) may benefit more from pages/blocks in regions D and E (enough large-sized pages are likely to be found invalid at the same time, allowing for GC efficiency to be not affected by the increased block size). On the other hand, workloads with much less sequentiality (e.g., cello) are likely to find pages/block in region F more suitable to them.

### C. Address Popularity and Update Frequency

**Metrics:** We use the well known address popularity distribution where the popularity of an address is captured by number of references to it. This is likely to have an impact on the efficacy of data management in a heterogeneous flash as we explain below. Next, we define a new metric cluster size distribution to capture how many groups (clusters) of logical block addresses (LBAs) are updated a comparable number of times over a time period. This is relevant since LBAs with similar update frequencies are candidates for residing on pages in the same block (which is good for GC). E.g., if we find a large-sized cluster (i.e., many LBAs) which is updated many times, we would like to choose a large-sized block.

**Observations and Insights:** We present the address popularity distribution for all our workloads in Figure 3 (a). We present the cluster size distribution for cello in Figure 3 (b) and Figure 3 (c), noting that other workloads show similar behavior and we omit them here. We observe that only a small portion (less than 10%) of LBAs (i.e., small-sized cluster) are accessed quite often (more than 50% of the total accesses) for workloads such as cello, rsrch, and prxy. These LBAs are likely to be easy to identify (usually called “hot” data), while the remaining LBAs (i.e., large-sized clusters) are much less accessed (and comprise “warm” and “cold” data). Such hot data may benefit from large-sized blocks (regions C – D) for better erase efficiency and chip density. Warm and cold data may be difficult to separate out and known to be the main cause of write amplification. This implies the use of small-sized blocks (regions F – G) for these. In fact, as a first-cut approximation, ignoring wear-leveling issues, the cluster size distribution in Figure 3 (b) and Figure 3 (c) offers the number of blocks of different sizes that would result in ideal GC behavior (i.e., one block of appropriate cluster size for every point in this scatter-plot). Note that even though Figure 3 (b) shows large-sized clusters have only a few references, the total number of bytes accessed from these large clusters are quite large as shown in Figure 3 (c). This indicates the necessity of choosing appropriate block sizes for these large clusters.
D. Does Size of Request Containing an LBA Change?

**Metrics:** This is important because it is likely to affect the feasibility and ease of an on-line management scheme that would try to place requests into suitable-sized pages to extract latency and throughout benefits of heterogeneity. In order to capture how the size of the request containing an LBA corresponding to a certain logical address changes over time, we use the metric temporal request size variation. This metric is likely to have an important role to play in whether heterogeneity can be effectively exploited in practice. There can be multiple ways to capture this property. E.g., one could simply consider the entire time-series representing the size evolution for each logical address. Clearly, this would result in an unmanageable amount of information. Instead, we choose a more compact representation based on the conditional frequency \( f(x|y) \) which is defined as follows: the number of times the same logical address appears in a request of size \( y \) and its very next occurrence is in a request of size \( x \).

**Observations and Insights:** Figure 4 shows the request size variation as a scatter-plot for prxy. We find similar results for other workloads and omit them here. We find that the majority of consecutive requests containing the same LBA do not change their sizes over their lifetime (i.e., fall on the diagonal of Figure 4). This is good news for the feasibility and ease of online management and data placement in a heterogeneous flash. Specifically, when using such a flash, the majority of LBA accesses would not have to alternate between regions \( C, B, G \) and regions \( D, E, F \).

### III. Future Directions

Our proposal for a heterogeneous flash opens up a new line of research for its design and implementation in the datacenter.

**Implementation Style:** Multiple styles are possible, each unique in the specifics of the trade-offs associated with different basic units. Two such styles already exist: ganging/superblocks [2] and SLC/MLC hybrids [6], [14], [7]. As shown in Figure 5 (a), heterogeneity created using ganging/superblocks allows multiple page accesses in one step by sharing control lines across multiple planes/chips. It essentially implements large-sized pages via parallelism. However, the level of parallelism is restricted by the number of planes and buses of a chip.

A second way of constructing heterogeneity as shown in Figure 5 (b) is via SLC/MLC hybrids. This approach takes advantage of the higher density of MLC and the higher endurance and lower write latency of SLC. By programming the corresponding bits, a cell can provide dual modes: SLC mode and MLC mode, which allows flexible switch of page sizes. However, endurance and performance are orders of difference between these two modes.

A third way, that we plan to explore and compare with the previous two, is to build heterogeneous flash from the scratch: manufacturing different-sized pages and blocks on a chip as shown in Figure 5 (c). This approach offers the most flexibility in terms of hetero-
Each plane consists of blocks and each block is composed of a number of pages.

<table>
<thead>
<tr>
<th>Impl. styles</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ganging/Superblocks</td>
<td>flexible switch of pages/blocks of different sizes</td>
<td>throughput gains depend on buses, heterogeneity restricted to max number of planes</td>
</tr>
<tr>
<td>SLC/MLC</td>
<td>flexible switch of pages of different sizes, high density of MLC</td>
<td>large endurance and performance difference, limited heterogeneity</td>
</tr>
<tr>
<td>Heterogeneity on chip</td>
<td>many heterogeneity options</td>
<td>manufacturing cost, static configuration</td>
</tr>
</tbody>
</table>

Table II

**Pros and Cons of Different Underlying Implementation Styles for Heterogeneity**

geneity and hence may achieve better trade-offs between performance, density, GC efficiency and lifetime.

Each of the above implementation styles has its own pros and cons which are summarized in Table II. Note that these approaches can also be combined to introduce several levels of heterogeneity to exploit the best of their design trade-offs.

**Management Concerns:** There are several questions concerning the management of an heterogeneous flash that we plan to explore. Would the flash translation layer (FTL) and/or storage software (e.g. drivers or file systems) be able to gainfully exploit any heterogeneity? What should be the data placement strategies and how important is load balancing to such devices? Would GC be performed locally to a region or across regions? How will wear-leveling be affected? Does the powerful computing capability of SSD controllers imply that complex management strategies can be implemented in such heterogeneous devices?

**Implications on Datacenters:**

- **Datacenter Heterogeneity:** The heterogeneity in datacenters reflects in both hardware and software aspects. Due to machine replacements and upgrades, a datacenter may be composed of several generations of servers with diverse hardware configurations [3], [10]. Besides hardware diversity, datacenters usually host a mix of interactive (e.g., web search) and batch (e.g., map-reduce) types of applications [3]. These co-located applications typically exhibit different request access patterns such as request rates, data sizes, update frequency, temporal and spatial locality, etc [12], [5], [3]. Prior work [10] have explored the heterogeneity in the underlying servers as well as the mix of applications that will run on these machines and demonstrated significant performance opportunity due to the heterogeneity in datacenters. Heterogeneous flash brings another level of heterogeneity into memory/storage configurations, which may be explored to accommodate the diversity of data access patterns found in datacenter applications. This may further increase the performance opportunity of datacenter applications.

- **Resource Management:** Heterogeneous flash may affect datacenter resource management which poses several interesting research questions. For instance, what are the effective application placement and consolidation strategies in order to take advantage of the heterogeneity? Should datacenter level system software such as distributed file systems be aware of this heterogeneity? How would data placement and replication be affected to explore the heterogeneity? How would other tiers in the memory/storage hierarchy leverage heterogeneous flash to achieve better performance and cost trade-offs? We expect that these questions will open up new and interesting lines of research.

IV. Acknowledgments

This work was supported in part by NSF grants 0811670, 1152479, 1205618, 1213052, and 1147388.

References


