Viyojit: Decoupling Battery and DRAM Capacities for Battery-Backed DRAM*

Rajat Kateja[1], Anirudh Badam[2], Sriram Govindan[2], Bikash Sharma[3], Greg Ganger[1]
[1]*Carnegie Mellon University,* [2]*Microsoft,* [3]*Facebook (work done while at Microsoft)*

Battery-Backed DRAM (BB-DRAM) is a popular form of non-volatile memory used in high performance systems. The poor scaling of battery capacity makes is challenging to provision high capacity BB-DRAM servers. Our paper presents Viyojit that decouples the battery and DRAM capacities, freeing BB-DRAM from the stunted battery growth. Viyojit dynamically bounds the number of dirty pages in BB-DRAM based on the provisioned battery. Experiments with the YCSB benchmarks using Redis key-value store demonstrate the efficacy of Viyojit in reducing the required battery capacity by up-to 89% with just 7% reduction in throughput. Viyojit addresses a practical challenge and has already piqued the interest of large data center designers.

1 Battery-backed DRAM and battery scaling

Battery-Backed DRAM (BB-DRAM) enables orders of magnitude improvement in storage system's performance by providing low-latency and high-bandwidth access to durable storage. BB-DRAM has been widely used for decades in high-end storage systems and is increasingly becoming popular in general purpose servers as well. BB-DRAM allows DRAM to be treated as durable storage with the battery used to keep the system running for long enough after a power failure to ensure that the contents of DRAM are flushed to a truly durable medium, such as a Flash SSD.

Unfortunately, it is difficult to provision battery backup for large DRAM. While DRAM capacities have experienced excellent scaling over time, allowing large DRAM capacities to be provisioned easily, battery capacities have scaled poorly. The stunted growth of battery capacities leads to (volumetrically) large and unwieldy batteries that pose many challenges for data center operators. Big data center designers, including Microsoft, Google and Facebook, use distributed server-level batteries for cost, energy, availability and maintainability reasons. Because of the poor battery capacity scaling, a server with 4 TB BB-DRAM requires a battery of roughly 25x the size of an average smart-phone battery. Making space for such large batteries is a major concern for data center operators, in addition to the extra packaging, cooling and maintenance costs. Furthermore, large batteries require an extended period of system outage after the power is restored for charging the batteries to full to re-enable battery backup.

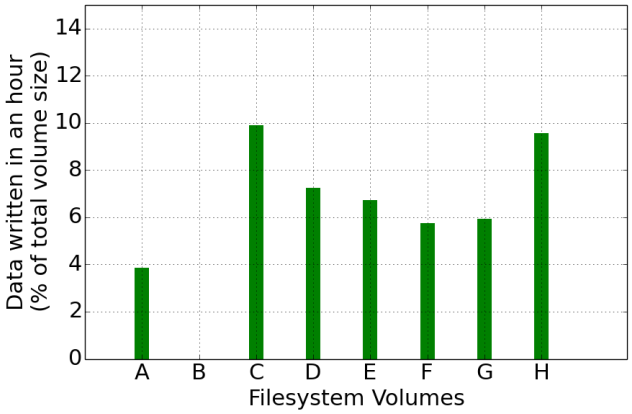

Figure 1: *Percentage of data updated for different file system volumes on an Azure blob storage server.*

2 Exploiting access-skew to reduce battery

Viyojit decouples battery requirements from DRAM capacities by leveraging the skew common in typical access patterns. Typically, applications write to, or update, only a small fraction of their total dataset within any given hour, day, etc. Most data is only read and does not need to be written back to a backing device in case of a power failure. This enables a much smaller battery capacity to support writing out just the dirty data. We support the above claim with an analysis of production workloads from Microsoft.

Our paper presents an analysis of real access traces from Microsoft's data center workloads to quantify the skew in typical applications. We analyze four workloads, namely, Azure blob storage, cosmos, page rank and search index serving. For each of the workloads, we analyze the amount of data that was written to within different time windows. We present a brief sampling of the results for Azure blob storage in Figure 1, demonstrating that the percentage of data written within an hour never exceeds 15% for any of the volumes. The results are similar for other workloads and time windows (see paper).

Based on the analysis, we conclude that a relatively small fraction of the total datasets are typically dirtied. The write skew motivates a design of BB-DRAM systems with reduced battery capacities. However, provi-

*The full paper, titled "Viyojit: Decoupling Battery and DRAM Capacities for Battery-Backed DRAM" was presented at the International Symposium on Computer Architecture (ISCA), June 2017.

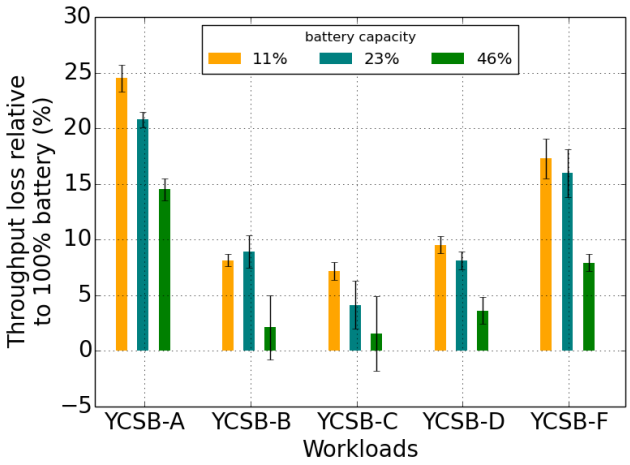

Figure 2: *Throughput reduction for various YCSB workloads with varying battery capacities relative to 100% battery.*

sioning smaller battery capacity necessitates a mechanism to ensure that the system will not have more dirty data than can be flushed using the battery capacity. Viyojit addresses that need by dynamically bounding the amount of dirty data in DRAM.

3 Bounding dirty data to reduce battery safely

Viyojit decouples the battery and DRAM capacities without compromising on data durability by enforcing a bound on the total amount of dirty data in BB-DRAM at any point of time. The bound is determined based on the provisioned battery capacity and enforced at a page granularity. For a bound of N pages, any (and only) N pages in DRAM can be dirty.

Viyojit is implemented as a shared library that exposes a mmap-like API. Applications use the API to allocate memory in BB-DRAM and then run as normal, unaware of the smaller battery capacity. Under the covers, Viyojit enforces the bound on the total number of dirty pages in the system. It write-protects all battery-backed pages allocated to any application, triggering a page fault when a page is written to. In the page fault handler, Viyojit 1) adds the faulting page to a list of dirty pages and checks the total number of dirty pages in the system, 2) writes back an already dirty page if required to maintain the bound, and 3) removes the write protection from the faulting page. This ensures that the total number of dirty pages never exceed what the provisioned battery can support.

To reduce the number of page write-backs, Viyojit identifies and writes-back infrequently written pages. It employs a least recently updated target selection policy to identify write-cold pages to write-back. The write coldness/hotness is determined by periodically checking and clearing the dirty bit of known-to-be-dirty pages. Clearing the dirty bits does not affect Viyojit's dirty page tracking because Viyojit maintains a list of dirty pages and takes charge of writing those back in the event of a power failure. Writing back write-cold pages avoids frequent write-backs of the same pages, such as write-hot pages that become dirty soon after a write-back.

Write-backs are performed in the background as well to reduce the number of on-demand write-backs. The number of pages to keep in dirty state rather than writing them back, and consequently the number of new dirty pages that can be absorbed without triggering an on-demand write-back, is determined dynamically based on the "dirty pressure". The dirty pressure is computed as an exponentially decaying average of the number of new dirty pages in past periods. The number of new dirty pages in each period is determined while checking the dirty bits in each period for identifying write-cold data.

4 Evaluation summary and discussion

Figure 2 (details in paper) summarizes the effect of smaller batteries on the throughput of various YCSB benchmarks relative to provisioning battery for 100% of BB-DRAM. As expected, the overheads increase with decreasing battery capacities and are workload dependent. With a battery capacity corresponding to merely 11% of the full battery capacity, the throughput reduction varies from 7% for a read-heavy workloads (YCSB-C) to 25% for a write-heavy workloads (YCSB-A). A similar trend is observed for request latencies as well. Since Viyojit only incurs overhead on writes, the overheads are higher for write-heavy workloads.

An interesting observation, presented in Figure 10 of our paper, is that as the total BB-DRAM sizes increase, the overheads of reduced battery capacity decrease. The decreasing overheads are due to the nature of skewed access patterns and demonstrate that Viyojit's approach to bound battery sizes should be increasingly effective as DRAM and data set sizes grow.

Viyojit's targeted environments, i.e. data centers, also offer a natural and elegant way to reduce the overheads for write-heavy workloads by amortizing them over multiple varied workloads. Each physical server in a data center typically hosts multiple virtual machines (VMs). Owing to inherent statistical multiplexing effects of such an environment, each physical server is expected to have a mix of read-heavy and write-heavy VMs. Carving up the (small) provisioned battery capacity among the VMs, with write heavy VMs getting larger than their proportional share at the expense of read-heavy VMs, would enable low overheads for each of the workloads.