Who's Afraid of Uncorrectable Bit Errors?
Online Recovery of Flash Errors with Distributed Redundancy

Amy Tai
VMware Research

Andrew Kryczka
Facebook Inc.

Shobhit O. Kanaujia
Facebook Inc.

Kyle Jamieson
Princeton University

Michael J. Freedman
Princeton University

Asaf Cidon
Columbia University

Flash has become the dominant storage medium for hot data in datacenters [14, 15], since it offers significantly lower latency and higher throughput than hard disks. Many storage systems are built atop flash, including databases [4], caches [16], and file systems [12].

However, a perennial problem of flash is its limited endurance, or how long it can reliably correct raw bit errors. As device writes are the main contributor to flash wear, its lifetime is measured in the number of writes or program-erase (P/E) cycles the device can tolerate before exceeding an uncorrectable bit error threshold. Uncorrectable bit errors are errors that are exposed externally and occur when there are too many raw bit errors for the device to correct.

In hyper-scale datacenters, operators constantly seek to reduce flash wear by limiting flash writes. At Facebook, for example, a dedicated team monitors application writes to ensure they do not prematurely exceed manufacturer-defined device lifetimes. Even worse, each subsequent flash generation tolerates a smaller number of writes before reaching end-of-life (see Figure 1a) [10]. Further, given the scaling challenges of DRAM [13] and the increasing cost gap between DRAM and flash [8, 9], many operators are migrating services from DRAM to flash [2, 8], increasing the pressure on flash lifetime.

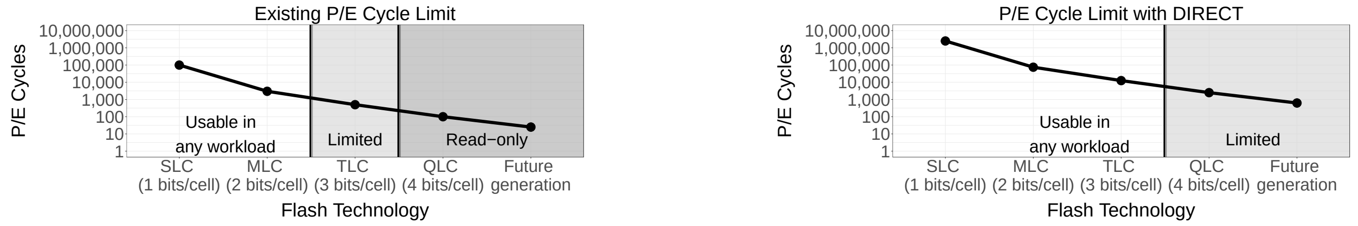
There is a variety of work that attempts to extend flash lifetime by delaying the onset of bit errors [1, 4, 7, 11, 17]. This paper takes a contrarian approach. We observe that flash endurance can be extended by *allowing* devices to go beyond their advertised uncorrectable bit error rate (UBER) and embracing the use of flash disks that exhibit much higher error rates. We can do so without sacrificing durability because datacenter storage systems replicate data on remote servers, and this redundancy can correct bit error rates orders of magnitude beyond the hardware error correction mechanisms implemented on the device. However, the challenge with higher flash error rates is maintaining availability and correctness.

We introduce Distributed error Isolation and RECovery Techniques (DIRECT), which is a set of three simple general-purpose techniques that, when implemented, enable distributed storage systems to achieve high availability and correctness in the face of uncorrectable bit errors:

1. **Minimize data error amplification.** DIRECT detects errors using existing error detection mechanisms (e.g., checksums) and recovers data from remote servers at the smallest possible granularity.
2. **Minimize metadata error amplification.** A corruption in local metadata (e.g., database index), often requires a large amount of data to be re-replicated. DIRECT avoids this by adding redundancy locally to local metadata.
3. **Ensure safe recovery semantics** by treating recovery operations as write operations. DIRECT serializes recovery operations on corrupted data against concurrent operations with respect to the system's consistency guarantees.

The difficulty of implementing DIRECT depends on two properties of the underlying storage system. The first property is whether the system is physically or logically replicated. Physically-replicated systems replicate *data blocks* between servers, while logically-replicated systems replicate the *commands* (e.g., write, update, delete). In physically-replicated systems, a certain object is stored in the same block or file on another server and therefore can be recovered efficiently by simply re-replicating the remote data block. This does not work for logically-replicated systems, where physical blocks are not identical across replicas. The second property is whether the data store supports versioning. In systems with versioning, we need to guarantee the recovered object does not override a more up-to-date version.

We demonstrate how to generalize DIRECT techniques by implementing them in two popular systems that are representative of two different classes of storage systems: (1) the Hadoop Distributed File System (HDFS), which is a physically-replicated storage system without versioning, and (2) ZippyDB, a distributed system that implements logical replication and transactions on top of RocksDB, a popular key-value store that supports key versioning. Objects in HDFS are physically-replicated, so it is straightforward for DIRECT



(a) Existing hardware-based error correction.

(b) Augmenting existing error correction with DIRECT.

Figure 1: For each generation of flash bit density, the average number of P/E cycles after which the uncorrectable bit error rate falls below the manufacturer specified level (10^{-15}). Beyond MLC, the number of flash writes the application can issue is limited [6]. With current hardware-based error correction, QLC technology and beyond can only be used for applications that are effectively read-only [3, 5]. DIRECT enables the adoption of denser flash technologies by handling errors in the distributed storage application. We model the UBER tolerated by DIRECT, while the UBER to P/E conversion was derived from data in a Google study [15].

to find the corrupt object in another replica and recover it at a high granularity. On the other hand, recovery is challenging in ZippyDB since the corrupted region of one replica is stored in a different location on another replica, so the recovered key-value pairs might not have consistent versions ZippyDB.

DIRECT leads to significant increases in device lifetime, since systems can maintain the same probability of application-visible errors (durability) at much higher device UBERs. In Figure 1b, we estimate the number of P/E cycles gained with DIRECT using an empirical UBER vs P/E cycle comparisons in a Google study [15]. Depending on workload parameters and hardware specifications, DIRECT can increase the lifetime of devices by 10-100×. This allows datacenter operators to replace flash devices less often and adopt lower cost-per-bit flash technologies that have lower endurance. DIRECT also provides the opportunity to rethink the design of existing flash-based storage systems, by removing the assumption that the device fixes all corruption errors. Furthermore, while this paper focuses on flash, DIRECT's principles also apply in other storage mediums, including NVM, hard disks, and DRAM.

In summary, this paper makes several contributions:

1. We observe flash lifetime can be extended by allowing devices to operate at much higher bit error rates.
2. We propose DIRECT, general software techniques that enable storage systems to maintain performance and high availabily despite high hardware bit error rates.
3. We design and implement DIRECT in two storage systems, HDFS and ZippyDB, that are representative of physical and logical replication, respectively. Applying DIRECT results in significant end-to-end availability improvements: it enables HDFS to tolerate bit error rates that are 10,000×-100,000× greater, reduces application-visible error rates in ZippyDB by more than 100×, and speeds up recovery time in ZippyDB by 10,000×.

References

[1] LevelDB. http://leveldb.org.

[2] McDipper: A key-value cache for flash storage. https://www.facebook.com/notes/facebook-engineering/mcdipper-a-key-value-cache-for-flash-storage/10151347090423920/.

[3] Micron 5210 ION SSD. https://www.micron.com/solutions/technical-briefs/micron-5210-ion-ssd.

[4] RocksDB. http://rocksdb.org.

[5] P. Alcorn. Facebook asks for QLC NAND, Toshiba answers with 100TB QLC SSDs with TSV. http://www.tomshardware.com/news/qlc-nand-ssd-toshiba-facebook,32451.html.

[6] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai. Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 521–526, Dresden, Germany, 2012.

[7] A. Eisenman, A. Cidon, E. Pergament, O. Haimovich, R. Stutsman, M. Alizadeh, and S. Katti. Flashield: a hybrid key-value cache that controls flash write amplification. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 65–78, Boston, MA, Feb. 2019. USENIX Association.

[8] A. Eisenman, D. Gardner, I. AbdelRahman, J. Axboe, S. Dong, K. M. Hazelwood, C. Petersen, A. Cidon, and S. Katti. Reducing DRAM footprint with NVM in Facebook. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*, pages 42:1–42:13, 2018.

[9] D. Exchange. DRAM supply to remain tight with its annual bit growth for 2018 forecast at just 19.6%. www.dramexchange.com.

[10] L. M. Grupp, J. D. Davis, and S. Swanson. The bleak future of NAND flash memory. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, pages 17–24, San Jose, CA, 2012.

[11] J. Jeong, S. S. Hahn, S. Lee, and J. Kim. Lifetime improvement of NAND flash-based storage systems using dynamic program and erase scaling. In *FAST*, pages 61–74, 2014.

[12] K. Kambatla and Y. Chen. The truth about MapReduce performance on SSDs. In *Proceedings of the 28th Large Installation System Administration Conference*, pages 118–126, Seattle, WA, 2014.

[13] S.-H. Lee. Technology scaling challenges and opportunities of memory devices. In *Electron Devices Meeting (IEDM), 2016 IEEE International*, pages 1–1. IEEE, 2016.

[14] J. Meza, Q. Wu, S. Kumar, and O. Mutlu. A large-scale study of flash memory failures in the field. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 177–190, Portland, Oregon, 2015.

[15] B. Schroeder, R. Lagisetty, and A. Merchant. Flash reliability in production: The expected and the unexpected. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies*, pages 67–80, Santa Clara, CA, 2016.

[16] L. Tang, Q. Huang, W. Lloyd, S. Kumar, and K. Li. RIPQ: Advanced photo caching on flash for Facebook. In *Proceedings of the 13th USENIX Conference on File and Storage Technologies*, pages 373–386, Santa Clara, CA, 2015.

[17] K. Zhao, W. Zhao, H. Sun, X. Zhang, N. Zheng, and T. Zhang. LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives. In *Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13)*, pages 243–256, San Jose, CA, 2013.