Qeios

Peer Review

Review of: "Host-Guided Data Placement—Whose Job Is It Anyway?"

Stergios V. Anastasiadis[1]

1. Department of Computer Science and Engineering, University of Ioannina, Greece

Strategies to optimize placement utilizing device features can be expressed in a separate shim layer that interposes between the application and the operating system. Many configurations (the product of devices and applications) must be implemented, but these are easier to implement in isolation than within applications or operating systems. Reshim is a dynamic library that allows transparent hint injection or redirection of data. It consists of three components, the application call interception, the data placement engine and the device manager. In hint-based interfaces the interception requires limited state tracking, while in host-based interfaces flash is managed by a lightweight storage engine, called reshim-engine.

Reshim introduces two abstractions, the stream and the lifetime group. A stream represents a single logical writer, which performs writes for a set of related data. A lifetime group is a temporal subset of a stream, a group of blocks written together. The incoming data of an application can be distinguished into streams with a heuristic approach based on the operation and documentation of the application, or an automated approach that captures workload data to train a variety of algorithms.

Reshim-engine maps streams to individual zones with extent-based logic, allocates a number of extent-sized buffers on boot and allocates a new buffer to each writable file, and implements lazy garbage collection deferring data movement as long as possible. Reshim inherits the restrictions of preloading, has scope restricted to log-structured applications, and focuses on data placement.

The authors experimentally compare the performance of Reshim with that of f2fs and zenfs for RocksDB, and f2fs for MongoDB and CacheLib. Reshim is faster than f2fs in inserts and updates, and faster than f2fs and zenfs in random I/O tail latency. Reshim is faster than WiredTiger under MongoDB in log-structured merge tree mode, and faster than f2fs in the read latency of CacheLib. Reshim reduces write amplification and memory usage in comparison to f2fs.

doi.org/10.32388/F6JPDY

The manuscript provides an interesting approach to address a relevant problem and is relatively readable. The description includes experimental motivation about the problem solved and relatively detailed presentation of the interfaces and their implementation across the supported approaches. However, the description of the data placement based on heuristics and automation is rather vague with respect to the generated rules or the application of clustering algorithms.

The experimental evaluation provides performance results accompanied by an insightful breakdown of the cpu time across the different operations over RocksDB. Similarly, the authors demonstrate experimentally the reduced overheads of Reshim in garbage collection, data movement and buffer space. The figures of the experimental results (e.g., Fig 6, 7, 8, 9, 10, 11, could be improved by adding general title at the top, title and metric of the input parameter under the X axis and lines at the X and Y axis. Similarly, several of the tables could be aesthetically improved (e.g., Table 2, 3, table about memory in 4.4.2).

Although the authors cover a reasonable amount of related work, they could also cite Diciclo (ACM TOCS, May 2024) as an alternative approach of placing systems functionality at user level to achieve performance and security isolation across different applications sharing the same machine.

Overall, the manuscript makes a relevant contribution and deserves to be published.

Declarations

Potential competing interests: No potential competing interests to declare.