The Design and Implementation of UniKV for Mixed Key-Value Storage Workloads (Supplementary File)

Qiang Zhang, Yongkun Li, Patrick P. C. Lee, Yinlong Xu.

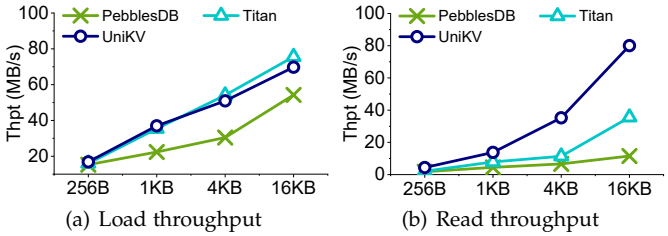

Fig. 1. Experiment S1 (Impact of KV pair size).

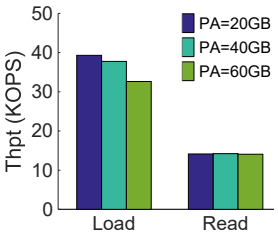

Fig. 2. Experiment S2 (Impact of partition size).

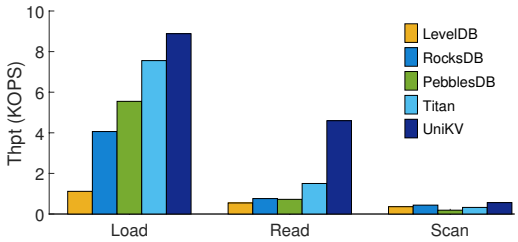

Fig. 3. Experiment S3 (Performance under direct I/O).

We present additional experimental results on the performance impact of UniKV for different parameters.

Experiment S1 (Impact of KV pair size). We study the impact of KV pair size varied from 256 B to 16 KB, and keep other parameter settings as before. Figure 1 shows the throughput of randomly loading 100 GB KV pairs, as well as the throughput of reading 10 GB KV pairs; note that the figure reports the throughput of KV stores in terms of MB/s instead of KOPS to better illustrate the performance trend with respect to the amount of data being accessed. As the KV pair size increases, all KV stores have higher throughput due to the efficient sequential I/Os. Compared with PebblesDB, UniKV consistently improves load and read performance under different KV pair sizes. When KV pairs become larger, the improvement of UniKV decreases for the throughput of loading a KV store, and the improvement increases for the throughput of reads. The reason is that PebblesDB maintains more SSTables in the first level as KV pair size increases. This reduces the compaction overhead, but causes reads to check these SSTables one by one, leading to degraded read performance. In contrast, UniKV always maintains fixed UnsortedStore and partition sizes. Thus, all the throughput of load and read in UniKV grows steadily as the KV pair size increases.

Besides, compared with Titan, UniKV always improves read performance under different KV pair sizes due to the hash indexing acceleration and two-layer architecture design. However, UniKV achieves slightly worse write performance than Titan when KV pair size is greater than or equal to 4 KB. The main reason is that UniKV only adopts partial KV separation which can simultaneously improve write performance and ensure good read/scan performance for the data in the UnsortedStore, but Titan adopts full KV separation which can obtain greater write performance improvement when KV size is larger.

Experiment S2 (Impact of partition size). We study the impact of partition size on UniKV. We again randomly load 100 M KV pairs and issue 10 M reads. Figure 2 shows the results by varying the partition size from 20 GB to 60 GB, while fixing the UnsortedStore size as 4 GB. The partition size only has a small impact on write performance and almost no impact on read performance. The reason is that GC is operated within each partition independently. Thus, the smaller the partition, the more efficient the GC operations for finer granularity of GC. However, the partition size influences the memory cost as UniKV needs to allocate a MemTable for each partition. Thus, a smaller partition size may incur more memory usage, so the size should be limited.

Experiment S3 (Performance under direct I/O). To avoid the impact of OS page cache on performance, we study the performance under direct I/O. Since all KV stores except RocksDB do not support direct I/O, we modify their source code to include the `O_DIRECT` attribute in the `open()` calls and turn off write-caching of disk to enable direct I/O. We first randomly load 100 M KV pairs, and issue 10 M reads and 1 M scans. Figure 3 shows the results. The throughput of writes, reads, and scans for all KV stores drops significantly under direct I/O since all operations involve disk I/O access. Nevertheless, UniKV outperforms other KV stores under direct I/O. It achieves 1.2-8.0× load throughput, 3.1-9.3× read throughput, 1.3-3.0× scan throughput compared with other KV stores. The improvements of UniKV for reads and scans increase under direct I/O, since the multi-level access and serious compactions in other KV stores read and write more SSTables through disk.