KIOXIA

Comparing Payload Sizes: RocksDB vs TRocksDB

Identifying when Each Database Storage Engine is Best Used

Scott Stetzer Edward Xiao Scott Harlin

Introduction

KIOXIA America, Inc. (formerly Toshiba Memory America, Inc.) published a Tech Brief in November 2019 entitled, "Introducing the TRocksDB Platform" (downloadable from GitHub®[1]). The brief compared performance and write amplification (WA) between RocksDB and TRocksDB database storage engines using four Facebook™-developed performance tests[2] and one WA test conducted internally by KIOXIA. Based on the results of these tests, TRocksDB achieved a lower WA for the system as a whole while delivering comparable or better random and sequential performance when tested against the RocksDB platform.

Though RocksDB is a popular database storage engine used by a range of database applications, it rewrites data at least one time for every level of the database, and in many cases, may perform multiple rewrites per level. Since the keys and values (records) are stored together in RocksDB, there is no mechanism in place to cache the keys separately. Given a single key-value pair, the total WA for RocksDB will be at least 21x, or more, depending on the workload, which in turn, can adversely affect application-level performance that may result in early wear-out of the SSD.

A new solution is under development at KIOXIA called TRocksDB. It, too, is a database storage engine with server-side software that works in conjunction with fast-performing SSDs to reduce the repeated data rewriting caused by application-generated WA. The TRocksDB design separates keys from values (records) that enable very fast and efficient database lookups, as well as faster and more efficient database queries versus RocksDB where it uses DRAM as a database cache. As validated in the November 2019 Tech Brief, TRocksDB reduced WA by up to 80% versus RocksDB[3]. In hard drives, WA is not an issue, but for SSDs where the life cycle of flash is based on how many writes the SSD can support (Drive Writes per Day or DWPD), it can have a negative effect on SSD endurance.

This application payload size comparison paper measures the performance between RocksDB and TRocksDB, with the theory that as the size of database records increase (also known as payload size), TRocksDB will gain more performance than the RocksDB engine. By separating keys from values (records), TRocksDB does not have to rewrite records over and over again during RocksDB compaction. The WA is reduced and performance of the overall database is improved, as well as the life expectancy of the SSDs. The results of the testing identifies the transition point in performance when one of the database engines is more advantageous to use over the other.

RocksDB / TRocksDB Overview

RocksDB and TRocksDB data storage engines have benefits and detriments associated with their respective data file architectures, but the one challenge when using RocksDB with SSDs is its high WA that results from the Log Structured Merge (LSM)-tree of the engine's design that is used to store data. KIOXIA set out to solve the endurance challenges associated with RocksDB WA and try to make it more versatile for SSDs.

Developed by Facebook in April 2012, RocksDB is a high-performance embedded database for key-value data that can be used as a storage engine within a larger database management system (DBMS). It has the ability to deliver massive scalability because it does not require database locks to invoke a read or write operation. This enables users to generate as many threads as they need to access the database without lock contention. Storage capacity is conserved through a layered data compaction scheme, where layers are of a defined size.

Data is structured through an LSM-tree, which is beneficial for providing indexed access to files with high insertion volume, such as log data, but rewrites data at least one time for every level of the database. When run in a steady state for longer intervals, the WA can increase even more requiring higher levels of repeated data rewriting that can negatively impact DWPD. As the LSM-tree structure design is responsible for an average WA of 21x, if one terabyte (1TB)[4]

of data is inserted into the RocksDB engine daily, an SSD could experience up to 21TB of data rewrites (1TB multiplied by 21x). Therefore, if the WA can be reduced, then the overall write workload to each SSD will also be reduced, enabling the SSD to record new data and still remain within its specified DWPD.

TRocksDB is a new database design (developed by KIOXIA) that improves storage and DRAM usage when compared to RocksDB as it reduces the repeated data rewriting caused by application-generated WA. It is extremely flexible and designed as an open-source improvement to RocksDB, with support for commercially-available fast SSDs. Its open-source approach comes with a version-controlled wiki that enables development code to be downloaded from GitHub.

TRocksDB enables values (records) to be stored in separately-managed files (Value Logs or VLogs), where only the keys, and not the values are stored in the LSM-tree. It uses an LSM-tree as a base to identify each key through a look-up table, and a reference pointer to find the value (record) of each key in a ring buffer. By caching keys in the LSM-tree, very small records can be kept in DRAM so that access to them can be much faster than with RocksDB.

Another key advantage of the TRocksDB design is when values (records) are much larger than keys, a low WA can be delivered by utilizing DRAM as a database cache. Predictive models showed that caching keys worked well in TRocksDB when the payload size was at least two times larger than the key size. In many real-world use case scenarios, records will be at least 10x larger, and more likely 20x to 500x larger. The larger the records, the more benefits that TRocksDB can deliver over traditional key-value stores, like RocksDB. These benefits include the ability to cache more levels of the database, decrease reads from storage, and reduce application-generated WA, all while delivering comparable or better SSD performance.

Benchmark Test Criteria

To validate the payload size benefits of TRocksDB when compared to RocksDB, a single benchmark test was conducted by KIOXIA America in a lab environment and based on the Yahoo! Cloud Service Benchmarking (YCSB) application. The all-encompassing test occurred in two phases that included the loading of data and the running of data. For each phase (Load or Run), two measurements were obtained that included the time it took to complete the transactions, as well as how many I/O transactions were required. The database engines under test included a current version of TRocksDB in development (t6.4.6) - a current release of RocksDB (v6.4.6) - and a YCSB default version of RocksDB (v6.2.2).

The two Load measurements obtained by the YCSB tool included Load Operations and Load Time as follows:

Load Operations (higher is better)	Load Time (smaller is better)
This measurement indicated how many more operations that TRocksDB could process versus RocksDB. The workload was write-intensive. Results were captured in operations per second using a range of payload sizes (500 to 4 million bytes) loaded in random order. An increase in load operations was the result goal, so the higher the result, the better its performance.	*This measurement determined how much new data that TRocksDB could push into a database versus RocksDB. The workload was write-intensive and included the time to load metadata, test data and transactional data. Results were captured in seconds using a range of payload sizes (500 to 4 million bytes) loaded in random order. A reduction in load time was the result goal, so the smaller the result, the better its performance.*

The two Run measurements obtained by the YCSB tool included Run Operations and Run Time as follows:

Run Operations (higher is better)	Run Time (smaller is better)
This measurement indicated how many more operations that TRocksDB could process versus RocksDB. The workload was mixed and included warehousing data, moving inventory, creating records and updating the database. Results were captured in operations per second using a range of payload sizes (500 to 4 million bytes) loaded in random order. An increase in run operations was the result goal, so the higher the result, the better its performance.	*This measurement determined the time it took for TRocksDB to read and write new data into a database versus RocksDB. This workload was mixed. Results were captured in seconds using a range of payload sizes (500 to 4 million bytes) loaded in random order. A reduction in run time was the result goal, so the smaller the result, the better its performance.*

From the data collected during the Run portion of the test, two additional measurements were obtained that included WA and database size as follows:

Write Amplification (smaller is better)	Database Size (smaller is better)
This measurement indicated the number of repeated data rewrites that were recorded within RocksDB using its LSM-tree structure design, and in TRocksDB using a ring buffer design. The goal was to demonstrate expected SSD life in host systems given a range of payload sizes (500 to 4 million bytes). A reduction in WA was the result goal, so the smaller the result, the better its performance.	*This measurement determined the size of the resulting database when the performance tests were run. The optimal goal was to generate 300GB of database given a range of payload sizes (500 to 4 million bytes). For some database applications, the size of the storage footprint may be relevant and worth measuring.*

The internal test environment developed by KIOXIA provided a common baseline to compare TRocksDB Load and Run measurements (with WA and database size) to RocksDB on the host-side and utilized the YCSB test application for comparison.

3a Test Equipment:
The hardware and software equipment used for the YCSB benchmark test and the six measurements included:

- **Databases Tested:**
  TRocksDB version t6.4.6.
  RocksDB versions 6.4.6 and 6.2.2.
- **Server:** Tyan AMD® EPYC™ 7601P dual-socket server featuring 32 processing cores, 2.20 GHz frequency, and 256GB of DDR4 RAM.
- **Operating System:** Ubuntu® 18.04.
- **Application:** Database.
- **Storage Devices:** Sixteen (16) NVMe™ SSDs in Linux® RAID0 with 14TB capacities.
- **Benchmark Test Software:**
  YCSB Git commit 4a990099a667c9226d5c33de7971f8b7ede9ffc0.
  YCSB Workload A for number of keys, value size and option files from RocksDB or TRocksDB.
- **Configuration Files:**
  RocksDB uses the 'unmodified standard options' file setting in the configuration file provided by the database.
  TRocksDB uses a different configuration file setting since the keys and records (values) are separated and set to 'optimized for general-purposes' level of operation and performance.

3b Test Procedures:
To create a workload that provided efficient testing of TRocksDB and RocksDB in a fair, consistent and efficient manner, and also enabled repeatable results, the YCSB benchmark tool was selected as the test engine. It provided a default workload (Workload A) from the YCSB application, in combination with the RocksDB Java® API, to deliver the six measurements provided in this white paper. A description of Workload A now follows:

Workload A: has a mix of 50/50 reads and writes, and generates data in the following manner:

- **Setup:** The number of keys was generated at run time: 300GB / payload size = # keys and used at the start of each the six measurements and at each payload size ranging from 500 bytes to 4 million bytes. The Workload A default uses 10 data fields for each record. The payload size for each record is 10 data fields x the data field length size. If the data field length is 50 bytes, then each record's payload size is 10 data fields x 50 bytes or 500 bytes.
- **Load Measurement Phase:** The first set of tests loaded a database file with random writes, up to the target database size of 300GB. Data was collected from the Load operation for the Load Time and Load Operations per second, and from these Load measurements, the WA portion of these tests was also recorded.
- **Run Measurement Phase:** The second set of tests performed update operations (50% read and 50% write) up to the target database size of 300GB. Data was collected from the Run portion of the test that included Run Time and Run Operations per second, and from these Run measurements, the WA and total database size portions of these tests was also recorded.
- **Repeat Measurement Process:** For each version of database engine (TRocksDB t6.4.6, RocksDB 6.4.6 and 6.2.2) and for each payload size (500 bytes to 4 million bytes).
- **Record Measurements:** Once each measurement was obtained for each payload size, the YCSB benchmark posted all of the measurements at the end of the test process.

Test Results and Analysis

The YCSB benchmark results are divided into the six measurements conducted internally by KIOXIA as follows:

Load Operations (Chart 1)

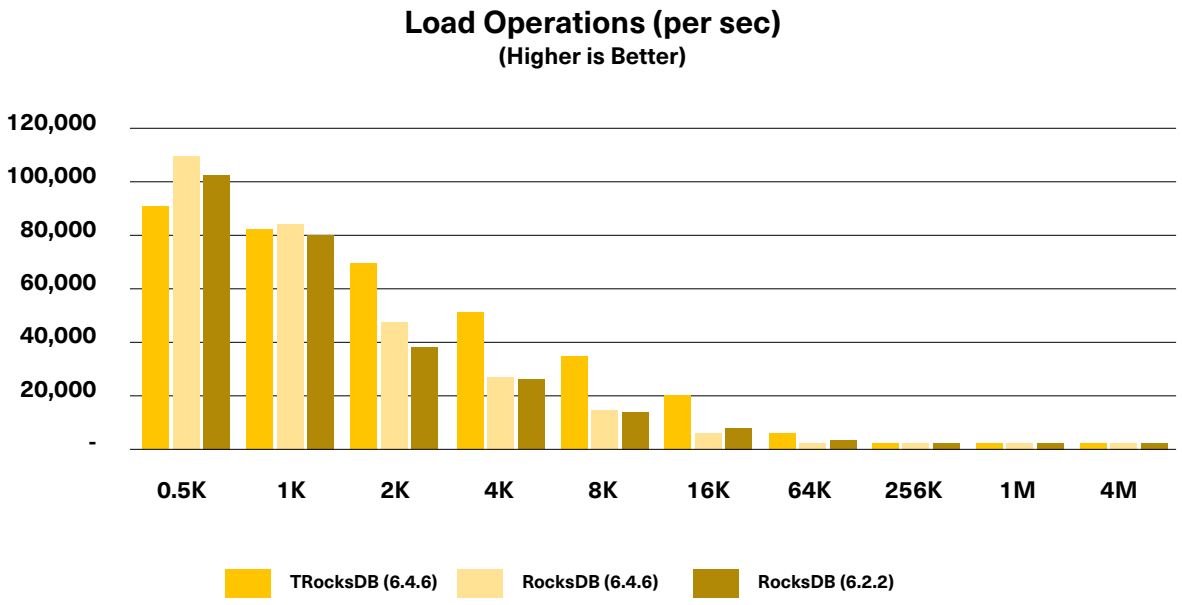

Chart 1: the test results depict how many more operations TRocksDB can process versus RocksDB

Data Points (higher is better):

Payload Size (in bytes)	TRocksDB t6.4.6 (operations / sec)	RocksDB v6.4.6 (operations / sec)	TRocksDB Advantage	RocksDB v6.2.2 (operations / sec)	TRocksDB Advantage
500	90,481	109,187	-17%	102,113	-11%
1,000	82,362	82,866	-1%	78,891	+4%
2,000	68,990	46,769	+48%	37,968	+82%
4,000	50,865	26,370	+93%	25,397	+100%
8,000	34,668	13,482	+157%	13,429	+158%
16,000	20,032	5,901	+239%	7,453	+169%
64,000	5,400	1,910	+183%	1,919	+181%
256,000	1,309	438	+199%	488	+168%
1,000,000	339	108	+214%	131	+158%
4,000,000	78	30	+159%	29	+167%

Analysis:

RocksDB is optimized for key-value operations making it very fast for small values (records). When the payload size increased, the RocksDB LSM architecture started to show its limits for efficiency. As depicted in Chart 1, RocksDB performed faster than TRocksDB at 500 bytes payload, but as the payload size increased to 1,000 bytes payload, the TRocksDB architecture started to exceed the performance of the normal RocksDB implementation. At a payload size of 2K, the performance improved by a significant margin and continued that way as the payload size increased to 4 million bytes payload.

Load Time (Chart 2)

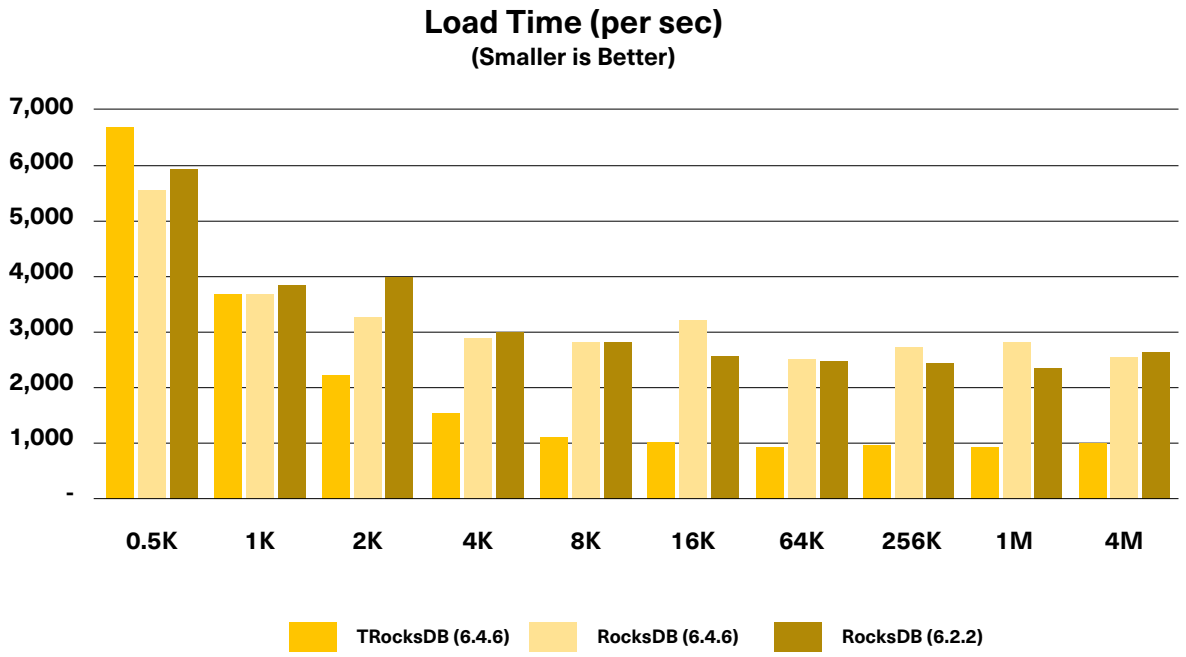

Chart 2: the test results show how much new data TRocksDB can push into a database versus RocksDB

Data Points (smaller is better):

Payload Size (in bytes)	TRocksDB t6.4.6 (in seconds)	RocksDB v6.4.6 (in seconds)	TRocksDB Advantage	RocksDB v6.2.2 (in seconds)	TRocksDB Advantage
500	6,361	5,495	-21%	5,876	-13%
1,000	3,642	3,620	-1%	3,803	+4%
2,000	2,174	3,207	+32%	3,951	+45%
4,000	1,474	2,844	+48%	2,953	+50%
8,000	1,082	2,782	+61%	2,792	+61%
16,000	936	3,178	+71%	2,516	+63%
64,000	868	2,454	+65%	2,443	+64%
256,000	895	2,674	+67%	2,401	+63%
1,000,000	886	2,783	+68%	2,296	+61%
4,000,000	967	2,505	+61%	2,583	+63%

Analysis:

Chart 2 measures of the total time to write the database to the 4 million payload limit of the test. These results are similar to the Load Operations results and a second data point that the transition for 100% write activity changes at a 1K payload size for TRocksDB.

Run Operations (Chart 3)

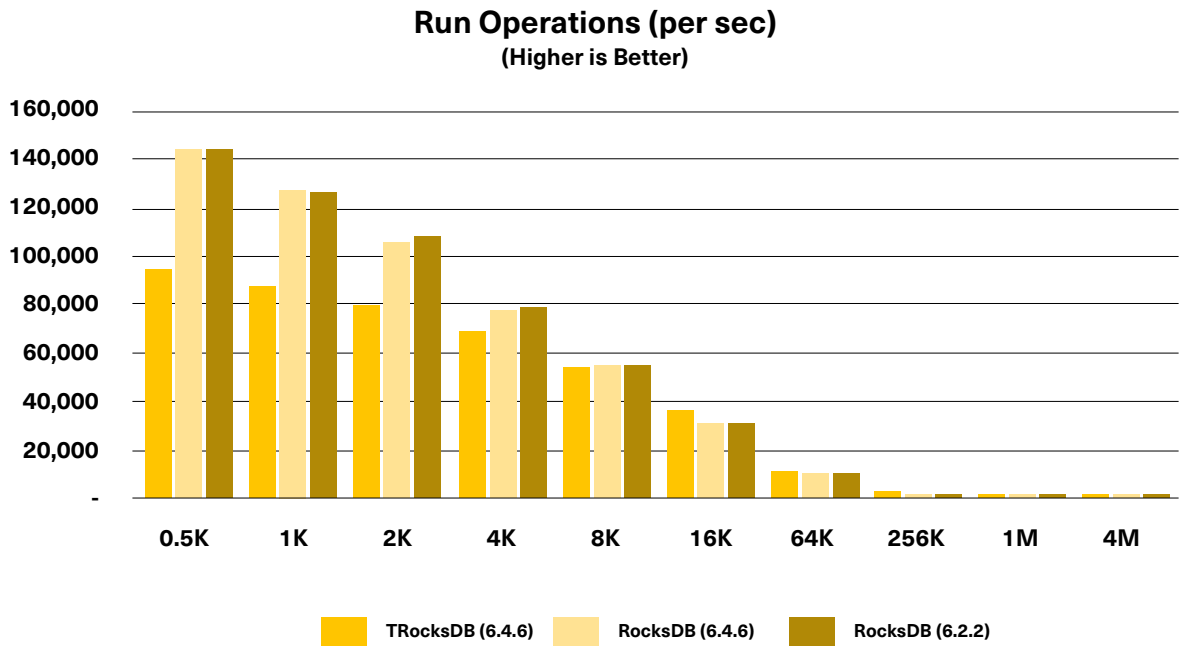

Chart 3: the test results present the time it took to run TRocksDB operations versus RocksDB

Data Points (higher is better):

Payload Size (in bytes)	TRocksDB t6.4.6 (operations / sec)	RocksDB v6.4.6 (operations / sec)	TRocksDB Advantage	RocksDB v6.2.2 (operations / sec)	TRocksDB Advantage
500	94,020	143,757	-35%	143,798	-35%
1,000	87,956	127,295	-31%	127,013	-31%
2,000	80,312	105,492	-24%	108,274	-26%
4,000	68,730	77,492	-12%	78,714	-13%
8,000	53,889	54,854	-2%	55,052	-2%
16,000	36,400	31,233	+17%	31,394	+16%
64,000	11,187	10,101	+11%	9,922	+13%
256,000	2,836	1,701	+67%	1,780	+59%
1,000,000	321	245	+31%	246	+31%
4,000,000	48	47	+3%	46	+5%

Analysis:

RocksDB is optimized for key-value operations making it very fast for small values. When the payload size increased beyond 8K, the RocksDB LSM architecture started to show its inefficiency. As depicted in Chart 3, RocksDB performed faster than TRocksDB at 500 to 4,000 bytes payload, but as the payload size increased to 8,000 bytes payload, the TRocksDB architecture started to exceed the performance of the RocksDB implementation. At a larger payload size of 256K, the performance improved by a significant margin and continued that way as the payload size increased to 1 million bytes payload.

Run Time (Chart 4)

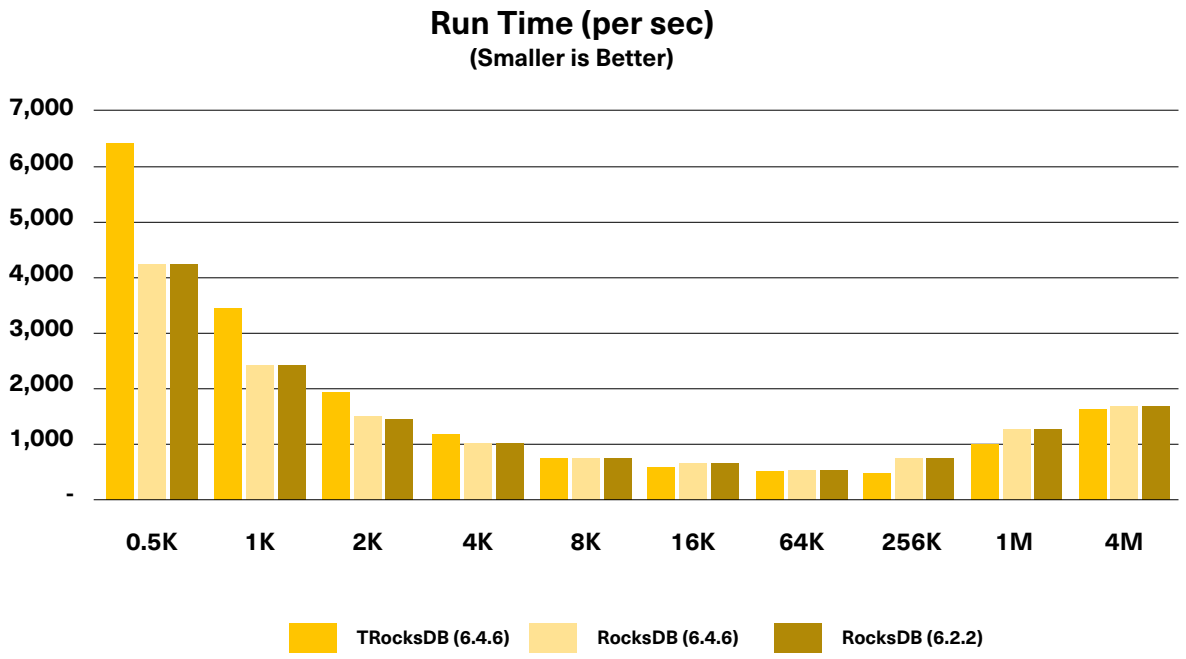

Chart 4: the test results present the time it took to read and write new data into the database

Data Points (smaller is better):

Payload Size (in bytes)	TRocksDB t6.4.6 (in seconds)	RocksDB v6.4.6 (in seconds)	TRocksDB Advantage	RocksDB v6.2.2 (in seconds)	TRocksDB Advantage
500	6,382	4,174	-53%	4,173	-53%
1,000	3,411	2,357	-45%	2,362	-44%
2,000	1,868	1,422	-31%	1,385	-35%
4,000	1,091	962	-13%	953	-15%
8,000	696	684	-2%	681	-2%
16,000	515	600	+14%	597	+14%
64,000	419	464	+10%	472	+11%
256,000	413	689	+40%	658	+37%
1,000,000	934	1,222	+24%	1,218	+23%
4,000,000	1,556	1,604	+3%	1,623	+4%

Analysis:

As depicted in Chart 4, RocksDB performed faster than TRocksDB at 500 to 4,000 bytes payload, but as the payload size increased to 8,000 bytes payload, the TRocksDB architecture started to exceed the performance of the RocksDB implementation. At a larger payload size of 256K, the performance improved by a significant margin and continued that way as the payload size increased to 1 million bytes payload.

Write Amplification (Chart 5)

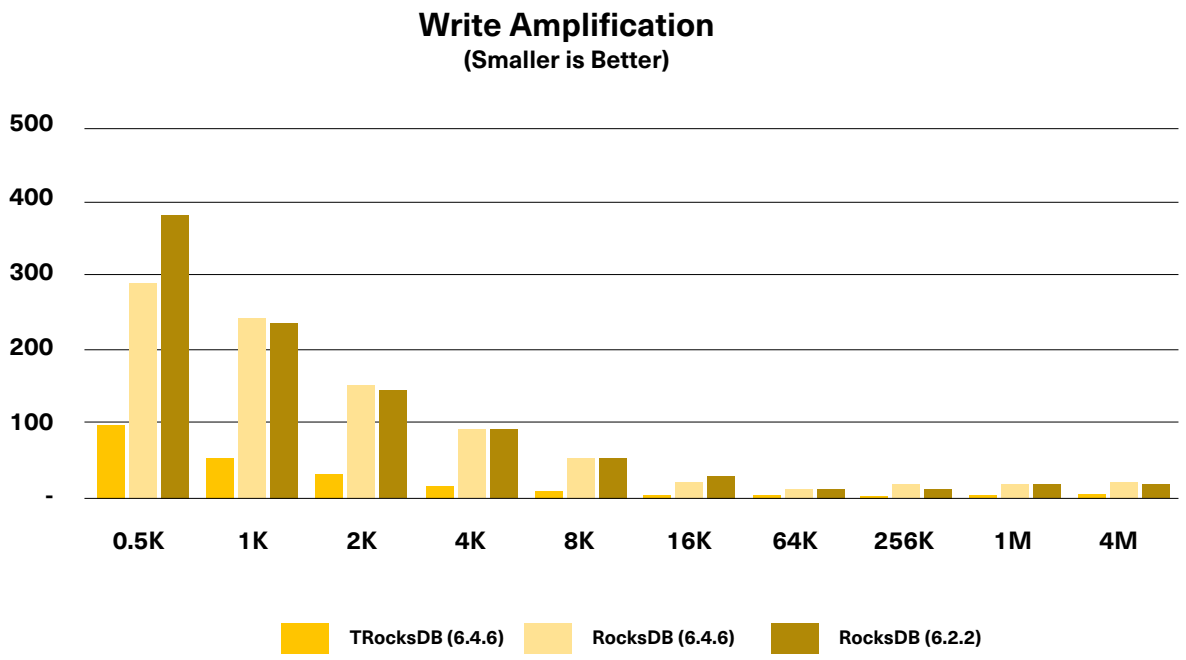

Chart 5: the test results show the number of repeated data rewrites recorded within TRocksDB and RocksDB

Data Points (smaller is better):

Payload Size (in bytes)	TRocksDB t6.4.6 (in WA)	RocksDB v6.4.6 (in WA)	TRocksDB Advantage	RocksDB v6.2.2 (in WA)	TRocksDB Advantage
500	98.5	289.4	+2.9x	383.5	+3.9x
1,000	51.3	245.9	+4.7x	238.7	+4.7x
2,000	31.5	151.1	+4.8x	144.1	+4.6x
4,000	18.2	90.8	+5.0x	92.4	+5.1x
8,000	9.8	54.1	+5.5x	51.4	+5.2x
16,000	3.0	12.7	+4.2x	12.8	+4.3x
64,000	2.9	12.8	+4.4x	13.0	+4.5x
256,000	2.8	15.3	+5.5x	14.3	+5.1x
1,000,000	3.5	16.0	+4.6x	15.6	+4.5x
4,000,000	4.6	18.7	+4.1x	18.6	+4.0x

Test 5 WA Analysis:

Given the range of payload sizes tested, TRocksDB was significantly more effective in reducing WA. Small payload sizes have significant write amplification. For workloads that begin at 16K, there is a significant step change in the WA. As a result, TRocksDB is much kinder to the SSD across the entire payload size test range and reduced the application-generated WA on average from 4x to 6x when compared to RocksDB. Reducing the WA will help to reduce write-specific workloads so that the SSDs deployed within the TRocksDB platform will be able to record new data and still remain within their recommended DWPD.

Results: Database Size (Chart 6)

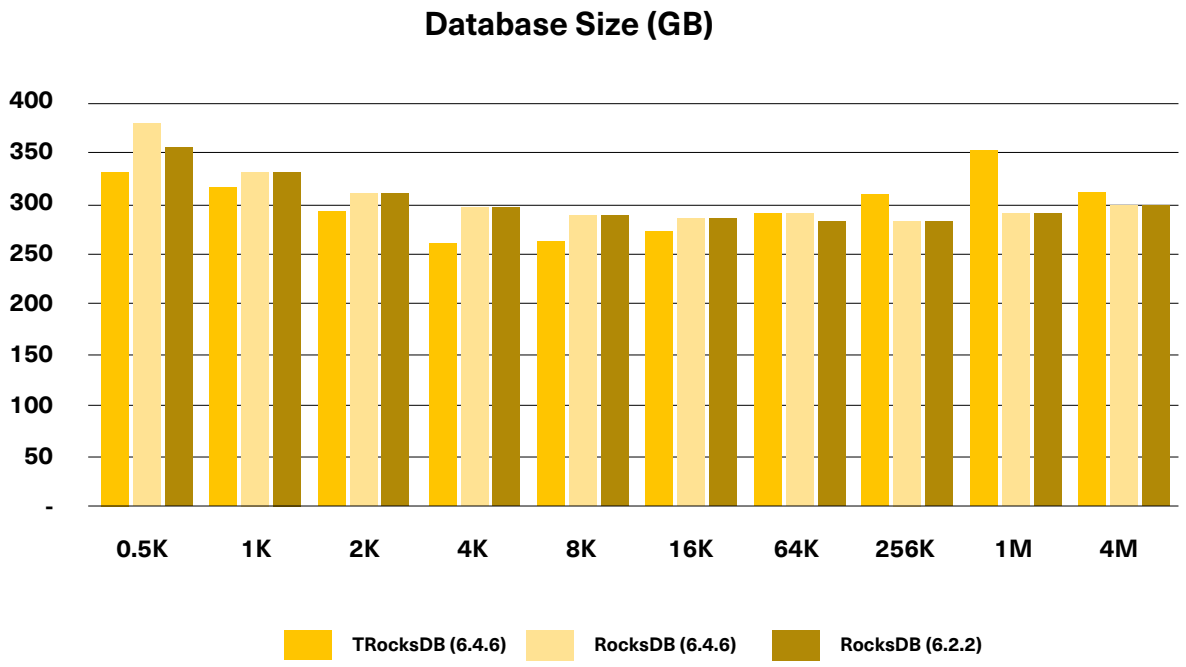

Chart 6: the test results show the size of the resulting database when running the tests between TRocksDB and RocksDB (the target of the testing was to generate 300GB of database for each test payload size - smaller is better)

Test 6 Data Points (smaller is better):

Payload Size (in bytes)	TRocksDB t6.4.6 (in GB)	RocksDB v6.4.6 (in GB)	TRocksDB Advantage	RocksDB v6.2.2 (in GB)	TRocksDB Advantage	Number of Keys Generated
500	331	375	+12%	354	+6%	600,000,000
1,000	311	328	+5%	329	+5%	300,000,000
2,000	291	306	+5%	306	+5%	150,000,000
4,000	257	294	+13%	294	+13%	75,000,000
8,000	259	287	+10%	287	-10%	37,500,000
16,000	271	284	+5%	284	+5%	18,750,000
64,000	288	281	-2%	281	-2%	4,687,500
256,000	306	281	-9%	281	-9%	1,171,875
1,000,000	351	289	-21%	289	-21%	300,000
4,000,000	308	295	-4%	295	-4%	75,000

Test 6 Database Size Analysis:

Overall, the database size measurement was consistent between both TRocksDB and RocksDB operations. While the total storage footprint may be relevant in some conditions, circumstances or scenarios as it related to the testing, there was not a significant or notable benefit when the RocksDB data file size was compared to the TRocksDB data file size.

Summary

The TRocksDB database engine design separates keys from values (records) and demonstrated performance benefits over the standard RocksDB platform when the payload size is larger. The benefit was realized at different payload sizes depending on the workload specified.

When payload sizes were relatively small, RocksDB provided an I/O performance benefit:

- *For 100% write-intensive workloads, RocksDB was faster than TRocksDB when payloads were under 1K in size.*
- *For mixed Read/Write workloads, RocksDB was faster than TRocksDB up to a payload size of 8K.*

TRocksDB showed a marked performance advantage over the standard RocksDB platform as the size of the payload increased:

- *For 100% write-intensive workloads, TRocksDB was faster than RocksDB when payloads were at or larger than 1K.*
- *For mixed Read/Write workloads, TRocksDB was equal at 8K and noticeably faster at 16K or larger.*

Write Amplification is a measure of importance to SSDs for endurance and life of the media. A lower WA means the life of the SSD will be improved:

- *For any configuration or payload size, the WA was vastly improved using TRocksDB option settings.*

Conclusion

If an application causes every write operation to be re-written many times within a database, the amount of new data that can be written to an SSD can be significantly reduced. Those that are addressing this challenge today, whether they be IT professionals, database administrators, system architects, data analysts and scientists, or chief data officers, will be interested in cost-effective solutions.

The TRocksDB platform responds to database queries faster and more efficiently then RocksDB as it uses DRAM as a database cache. In RocksDB, the keys and values are stored together with no mechanisms in place to simply cache keys. TRocksDB separates keys from records (values), and from the internal benchmark tests, reduced the associated WA for every payload size versus RocksDB. The results of the Load and Run tests were straightforward as well:

- *For very small values (<1K), the use of RocksDB is the better solution*
- *For anything larger than 1K, the use of the TRocksDB engine will provide better results*
- *If SSD life and endurance is a factor, TRocksDB is the preferred choice*

TRocksDB is server software that is now available under the terms of open-sourced licensing. The software is designed to work with fast-performing enterprise SSDs from any vendor and will run on any Linux hardware that RocksDB runs today. TRocksDB is fully integrated and compatible with RocksDB as both can be run from the same compiled code with the change of a configuration switch.

TRocksDB can be downloaded from the KIOXIA GitHub site at https://github.com/kioxiaamerica.

Notes:

[1] The GitHub branch platform includes a website and cloud-based service that stores and manages the code that a company develops. It is also a platform that tracks and controls any change to their code. Included in the KIOXIA America GitHub site is a TRocksDB tech brief that discusses its purpose, reasoning, history and architectural changes.

[2] These performance benchmarks were developed by Facebook and generated in July 2018. They measure RocksDB performance when data resides on flash storage and tested using RocksDB 6.1.2 and RocksDB 6.2.2. The benchmarks and test descriptions are available at https://github.com/facebook/rocksdb/wiki/Performance-Benchmarks. The test results were captured and accurate at the time of the testing and may vary with time as software and hardware technology evolves.

[3] This test measured the time it took to load a large dataset of 8 billion keys into each database in sequential order while overwriting 2 billion keys in random order, as outlined in the tech brief entitled, "Introducing the TRocksDB Platform," Section 4 Benchmark Test Criteria. Once this performance test was completed, the WA test was conducted on each database using the test results. The tech brief is available at https://github.com/kioxiaamerica.

[4] Definition of capacity: KIOXIA Corporation defines a megabyte (MB) as 1,000,000 bytes, a gigabyte (GB) as 1,000,000,000 bytes and a terabyte (TB) as 1,000,000,000,000 bytes. A computer operating system, however, reports storage capacity using powers of 2 for the definition of 1GB = 2^{30} bytes = 1,073,741,824 bytes, 1TB = 2^{40} bytes = 1,099,511,627,776 bytes and therefore shows less storage capacity. Available storage capacity (including examples of various media files) will vary based on file size, formatting, settings, software and operating system, and/or pre-installed software applications, or media content. Actual formatted capacity may vary.