PM aware storage engine for MongoDB

Moshik Hershcovitch
IBM Research
moshikh@il.ibm.com

Revital Eres
IBM Research
eres@il.ibm.com

Adam McPadden
IBM Systems
mcpaddea@us.ibm.com

ABSTRACT

With the maturity of Persistant Memories (PM) such as storage class memory technologies, e.g., STT-MRAM, PCM, ReRAM and 3DXpoint, we expect to see practical implementation of data structures, data stores and databases for use-cases such as IoT, mobile, and cloud. We developed a PM-aware storage engine for MongoDB which leverages PM hardware capabilities such as byte addressability and persistency. With our storage engine we see improved latency, less write amplification, less capacity and simpler implementation due to the fact that some code paths become unnecessary compared to past implementations.

KEYWORDS

Storage Class Memory, Database Architecture, Storage Engine

Introduction. MongoDB is the most popular NoSQL database, and the fifth most popular among all databases [2]. It stores JSON documents and the write operations are atomic for single document [1]. The component in mongoDB which is responsible for managing the data storage in a single node is the *storage engine*. The default commercial optimized storage engine in MongoDB is called *WiredTiger* (WT).

For our work we assumed that the data reside entriely in persistant memory without the need for additional storage devices. Moreover, we assume that the latency to the PM device is somewhat slower than the latency of DRAM but on the same order of magnitude.

The key idea behind our innovative design relies on the fact that PM hardware features enable us to adopt and explore new ways to ensure the persistency and the consistency of the database without using journaling and checkpointing, which are the traditionally techniques used in databases.

SYSTOR '18, June 4–7, 2018, HAIFA, Israel

ACM ISBN 978-1-4503-5849-1/18/06…$15.00
https://doi.org/10.1145/3211890.3211912

Design. Our PM aware storage engine uses a log structure architecture to store the document content, with groups of documents called segments that reside on PM, similar methods were used by RAMCloud [3]. During document insert we append the entire document to the end of the log. Then we update the index in DRAM to point to its persistent location in the PM. We achieve document level atomicity by adding CRC to the document, which allows us to identify consistent documents by validating the CRC following a crash, and deleting inconsistent documents which have invalid CRC. In this way we can avoid journaling and checkpointing, and still have a consistent view of the database.

Our PM aware storage engine supports regular storage engine commands (such as search, insert, delete, and update) and recover to a stable state after a crash. It leverages a garbage collection mechanism to compact segments and supports collection level concurrency control. For better performance we use direct access (DAX) mode (direct access to the PM, without DRAM copy) to access PM.

Benchmarks. Our PM aware storage engine is emulated on PM flexible prototyping platform named *ConTutto* which enables new memory technologies for IBM POWER servers. ConTutto is connected to NVDIMM-N as the PM device.

The performance evaluation of single-threaded YCSB benchmark shows that (1) WT is running x2 faster with PM as the storage device than running with SSD as the storage device; (2) MongoDB with Our PM aware storage engine is running up to 40% faster than MongoDB with WT running on PM (depends on the DRAM/PM ratio).

In addition to the performance improvement, we also gain better endurance (lower write amplification), better capacity and lower code paths in our PM aware storage engine since journaling and checkpointing are unnecessary.

Acknowledgments. The work described in the poster included collaboration with *Ronen Kat, Joel Nider, Hillel Kolodner, Michael Factor, Oliver OHalloran* from IBM.

REFERENCES

[1] 2018. Atomicity and Transactions. MongoDB, Inc. https://docs.mongodb.com/manual/core/write-operations-atomicity/
[2] 2018. DB-Engines Ranking. DB-Engines. https://db-engines.com/en/ranking
[3] Stephen M. Rumble, Ankita Kejriwal, and John K. Ousterhout. 2014. Log-structured memory for DRAM-based storage. In *Proceedings of the 12th USENIX conference on File and Storage Technologies, FAST 2014, Santa Clara, CA, USA, February 17-20, 2014*. 1–16.