X-Engine论文读书笔记

0、知识点概览

0.1、现状

X-Engine服务6亿用户，GMV达到7680亿。在双11的最大TPS为7000w。

0.2、X-Engine解决的问题

- 阿里交易系统的三个特点：
  - 活动时的交易陡增带来的海啸。
  - 热点数据打爆缓存。
  - 数据的热、温、冷状态转化极快。

0.3、X-Engine

- a write-optimized storage engine of POLARDB
- X-Engine是基于POLARDB的一个写优化引擎。

- which utilizes a tiered storage architecture with the LSM-tree (log-structured merge tree)
- 使用了基于LSM-tree的分层体系架构。

- to leverage hardware acceleration such as FPGA-accelerated compactions
- 使用了FPGA来进行LSM-tree的合并操作。

- a suite of optimizations including asynchronous writes in transactions
- 对交易的异步写操作进行优化。

- multi-staged pipelines
- 多级流水线优化。

- incremental cache replacement during compactions

- 合并过程中缓存增量替换。

0.4、海啸和泄洪的解决

- 应对交易陡增的解决方案。
  - Alibaba adopts a shared-nothing architecture for OLTP databases where sharding is applied to distribute transactions among many database instances, and then scales the number of database instances up before the spike comes.
  - 采用了shared-nothing的架构，通过水平进行数据库实例的水平扩容来应对业务徒增。

  - we address this problem by improving the single-machine capacity of storage engines, a core component of OLTP databases, so that the number of instances required for a given spike and throughput is reduced, or the achievable throughput given a fixed cost is increased.
  - 水平扩容也是昂贵的，我们进行单机优化，降低数据存储成本。

- 泄洪问题的解决
  - The flood discharge problem. The storage engine must be able to quickly move data from the main memory to the durable storage (i.e., discharge the flood building up in the memory) while processing highly concurrent e-commerce transactions.
  - 快速将内存中数据移动到持久化设备，迎接后续更高的交易流量。

  - X-Engine leverages this hierarchy by adopting a tiered storage layout, where we can place data in different tiers according to their temperatures.
  - 主要是通过将不同温度的数据存储在不同的层次。

  - The LSM-tree structure is a natural choice for a tiered storage.
  - LSM-tree天然就是一种分层的架构。

0.5、X-Engine的优化以及贡献

- X-Engine的优化和贡献。
  - We identify three major challenges for an OLTP storage engine in processing e-commerce transactions, and design X-Engine based on the LSM-tree structure.
  - 基于LSM-tree构建的分层的电商交易存储系统。

- We introduce a suite of optimizations for addressing the identified problems, such as a refined LSM-tree data structure, FPGA-accelerated compactions, asynchronous writes in transactions, multi-staged pipelines, and multi-version metadata index.
- 优化LSM-tree的数据结构、FPGA加速合并、交易的异步写入、多级流水、元数据度版本索引。

- We evaluate X-Engine extensively using both standard benchmarks and e-commerce workloads. Results show that X-Engine outperforms both InnoDB and RocksDB in e-commerce workloads during online promotions. We also show that X-Engine has stood the test of the tsunami of the Singles' Day Shopping Festival.
- X-Engine的性能优于InnoDB和RocksDB，并且经过双11交易的洗礼。

Table 1: Summary of optimizations in X-Engine.

Optimization	Description	Problem
Asynchronous writes in transactions	Decoupling the commits of writes from the processing transactions.	Tsunami
Multi-staged pipeline	Decomposing the commits of writes to multiple pipeline stages.	
Fast flush in $Level_0$	Refining the $Level_0$ in the LSM-tree to accelerate flush.	Flood discharge
Data reuse	Reusing extents with non-overlapped key ranges in compactions.	
FPGA-accelerated compactions	Offloading compactions to FPGAs.	
Optimizing extents	Packaging data blocks, their corresponding filters, and indexes in extents.	Fast-moving current
Multi-version metadata index	Indexing all extents and memtables with versions for fast lookups.	
Caches	Buffering hot records using multiple kinds of caches.	
Incremental cache replacement	Replacing cached data incrementally during compactions.	

- X-Engine的底层文件系统
  - X-Engine can be deployed on top of the POLARFS as part of POLARDB [4]. POLARFS has utilized many emerging techniques to achieve an ultra-low latency file system such as RDMA and NVMe.
  - POLARDB底层使用了POLARFS作为低延迟的文件系统。

0.6、X-Engine架构

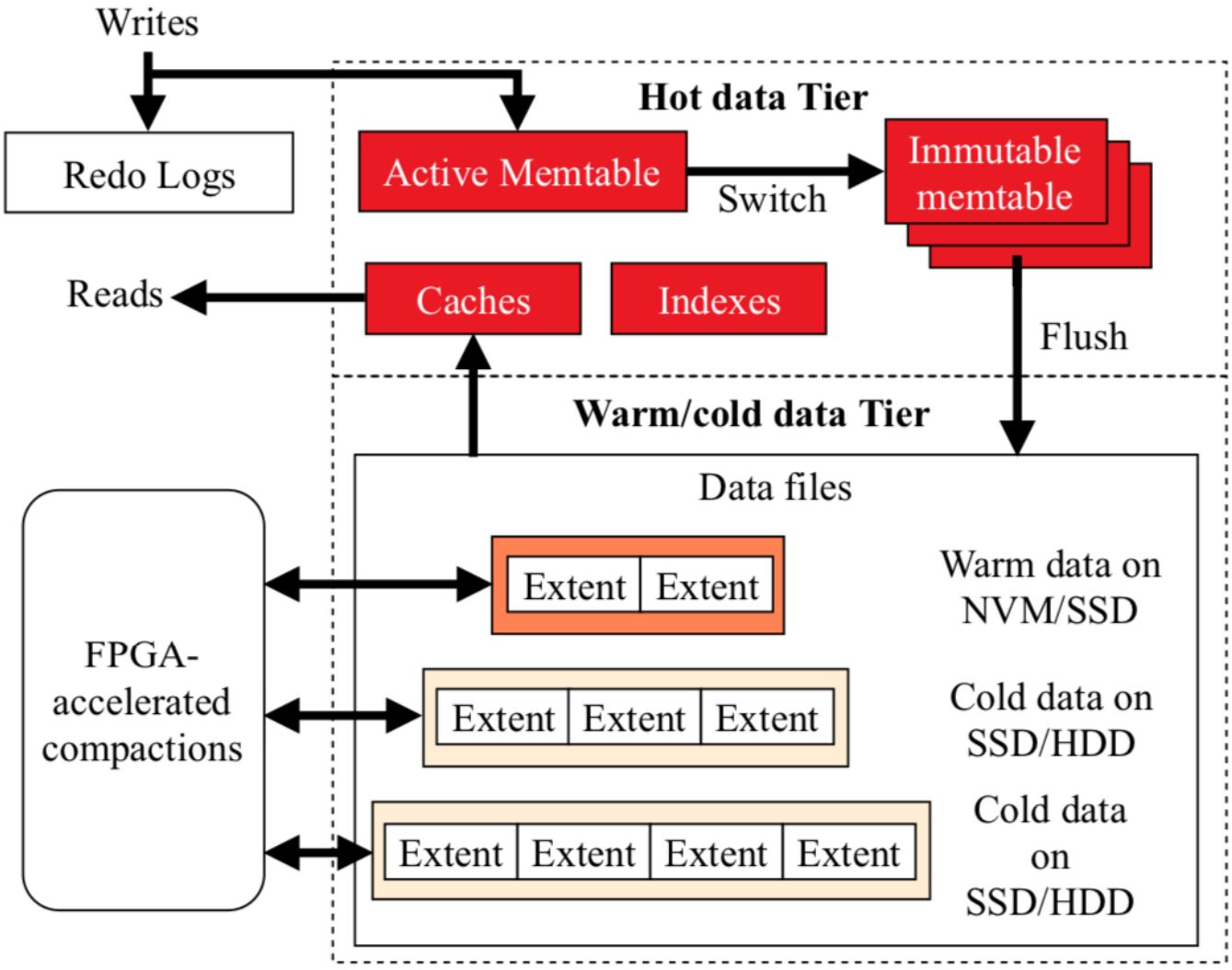

Figure 2: The architecture of X-Engine.

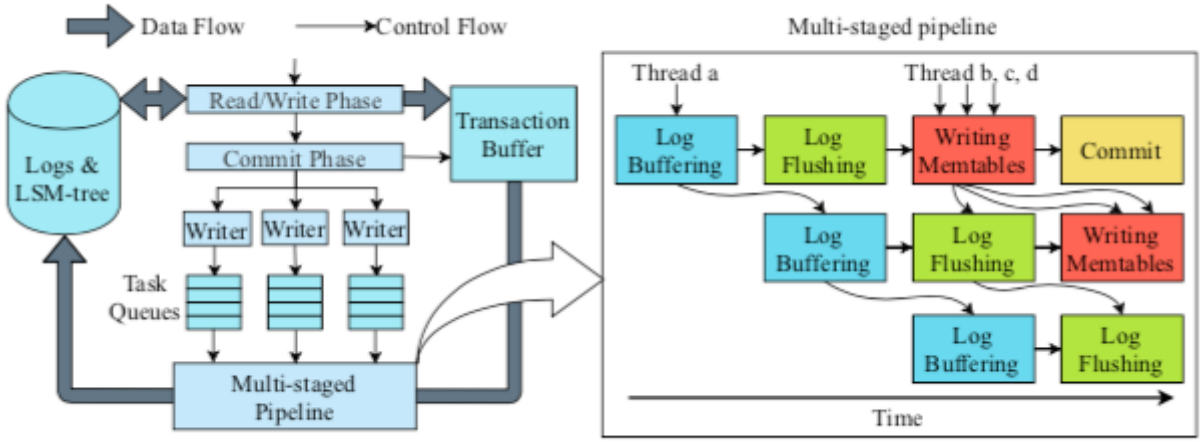

Figure 3: Detailed design of transaction processing in X-Engine.

0.6.1、X-Engine的MVCC

- X-Engine exploits redo logs, metasnapshots, and indexes to support Multi-version Concurrency Control (MVCC) for transaction processing.
- X-Engine利用了redo logs、metasnapshots、index来实现交易处理的MVCC。

0.6.2、X-Engine实现的技术细节

- Each metasnapshot has a metadata index that tracks all memtables, and extents in all levels of the tree in the snapshot.
- 元数据索引可以快速定位到所有的memtables.

- tables are partitioned into several sub-tables. Each sub-table has its own hot, warm and cold data tiers (i.e., LSM-trees).
- 每个表基于Range切分为多个子表。每个子表都有自己的热、温、冷数据层。

- X-Engine stores records in a row-oriented format.
- 采用了面向行存的格式。

- We design a multi-version memtables to store records with different versions to support MVCC.
- 通过一个多版本的memtables来实现不同数据版本的MVCC。

- The read path（读路径）
  - LSM-tree 通常对于点查并不友好。为了加速查询引入了manifest file来快速定位SST文件中key。每个文件中Key的快速定位还是使用了BloomFilter技术。

  - we optimize the design of extents, introduce a metadata index that tracks all memtables and extents, and a set of caches to facilitate fast lookups. We also propose an incremental cache replacement method in compactions to reduce unnecessary cache evictions caused by compactions. We introduce snapshots to ensure queries read the correct versions of records.
  - 采用了metadata index来快速定位到memtables，以及一系列缓存来加速查询。同时提出了缓存增量替换技术来减少合并过程中不必要的缓存回收。通过快照技术来保证查询多个版本的正确。

- The write path （写路径）
  - LSM-tree的通用写入路径为 log -> memtable -> immemtable -> storage。X-Engine将日志持久化和写入memtable进行了多级流水来提高吞吐。（We differentiate long-latency disk

I/Os and low-latency memory accesses in this process and organize them in a multi-staged pipeline to reduce idle states per thread and improve the overall throughput.）

- To achieve a high level of concurrency, we further decouple the commits of writes from the processing of transactions and optimize the thread-level parallelism for them separately.
- 为了应对事务的提交，我们通过解耦来转化为线程级的并发优化。

- Flush and compaction （刷盘和合并）
  - A LSM-tree relies on flush and compaction operations to merge data that may overwhelm the main memory from memtables to disks and keep the merged data in a sorted order.
  - Flush操作主要是及时释放内存，而合并操作是保证数据是有序的。

  - The original compaction algorithm has several drawbacks:
    - it consumes significant CPUs and disk I/Os;
    - 合并操作消耗了大量的CPU和磁盘IO

    - the same record is read from and written to the LSM- tree multiple times, causing write amplification;
    - 相同的记录被读取和写入多次，造成写放大。

    - it invalidates the cached contents of records being merged even if their values stay the same.
    - 合并的数据的缓存会失效，即使他们的值仍然保持不变。

  - In X-Engine, we first optimize the flush of immutable memtables. For compactions, we apply data reuse that reduces the number of extents to be merged, asynchronous I/O that overlaps merging with disk I/Os, and FPGA offloading to reduce CPU consumption.
  - X-Engine首先优化了 immutable memtables的Flush。对于合并，通过数据重用来减少合并的数据的放大, asynchronous I/O that overlaps merging with disk I/Os, 使用FPGA来加速合并。

  - X-Engine applies MVCC and 2PL (two-phase locking) to achieve the isolation level of SI (snapshot isolation) and RC (read committed) to guarantee the ACID properties for trans- actions.
  - 使用了MVCC和2PL来实现快照级别的隔离和读提交，从而保证事务的ACID特性。

- Different versions of the same record are stored as separate tuples with auto-increment version IDs. X-Engine tracks the maximum of these versions as the LSN (log sequence number). Each incoming transaction uses the LSN it sees as its snapshot. A transaction only reads tuples with the largest version that is smaller than its own LSN, and adds a row lock to each tuple it writes to avoid write conflicts.
- 同一条记录的不通版本通过自增版本ID来区分。最大的ID作为LSN。每个进入的事物都是用LSN作为快照。每个事物读取哪些比LSN都有的版本，并且为每个版本加一把行锁来避免写入冲突。

- X-Engine - Read - Extent
  通过额外的Extent来加快索引。

- We store the schema data with versions in each extent to accelerate DDL (Data Definition Language) operations. With this design, when adding a new column to a table, we only need to enforce this new column on new extents with new versions, without modifying any existing ones. When a query reads extents with schemas of different versions, it accords with the most recent version and fills default values to the empty attributes of records with the old schema. This fast DDL feature is important for online e-commerce businesses which adapt their designs of database schemas to changes in their requirements frequently.
- 通过Schema Data携带版本信息，升级后通过版本进行默认值填充。

1、X-Engine

1.1、Read

1.1.1、Cache

通过hash引擎，将不在memtable的查询减小到O(1)。在每次memtable进行flush的时候更新hash指向。

1.1.2、Multi-version Metadata Index

这里的多版本类似RocksDB的VersionSet管理机制。

1.1.3、Incremental cache replacement

每次不是删除所有旧的Extent的缓存信息，而是更新哪些有改动的。

1.2、Write

1.2.1、Multi-version memtable

在跳表的基础上，值是一个带version的linkedlist。

1.2.2、Asynchronous writes in transactions

通过任务队列和线程池进行异步写。

1.2.3、Multi-staged pipeline

通过将多个WAL和刷Memtable的多级流水来提高并发。

1.3、Flush and Compaction

1.3.1、Fast flush of warm extents in Level0

1.3.2、Accelerating compactions

1.3.2.1、Data reuse

尽量做一个切分，在合并过程中没有变更的Entent和DataBlock重用，减少磁盘IO。

1.3.2.2、FPGA offloading

通过FPGA进行数据Compaction，更充分利用宝贵的CPU资源。

1.3.3、Scheduling compactions

1.3.3.1、Compactions for deletions

1.3.3.2、Intra-Level0 compactions

merging sub-levels within Level0

1.3.3.3、Minor compaction

除最大层外，任意两次之间的合并。

1.3.3.4、Major compaction

最大层和之前所有层的合并。

1.3.3.5、Self-major compactions

最大层内的合并，较少碎片和删除数据回收。

2、总结

- X-Engine是一个OLTP的存储引擎。
- X-Engine解决和应对海量交易陡增带来的海啸。
- X-Engine解决了内存数据的泄洪问题。
- X-Engine基于优化的LSM-tree进行构建。
- X-Engine使用了FPGA进行Complaction加速。
- X-Engine将事物的写改为异步写入提高并发。
- X-Engine使用多级流水进一步提交WAL和memtable的吞吐。
- X-Engine使用了增量缓存替代技术进行数据服用减少磁盘IO。
- X-Engine使用了多种缓存加速点查。