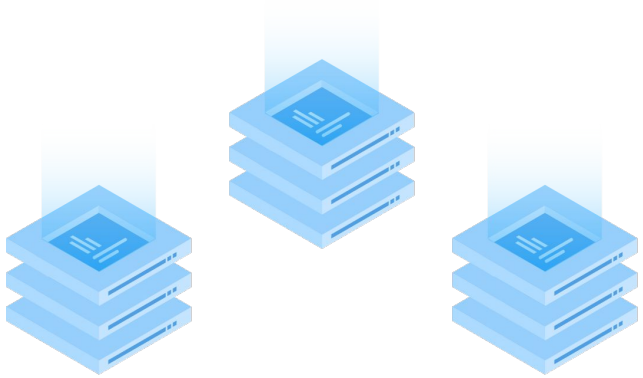
X-Engine: An Optimized Storage Engine for Large-scale E-commerce Transaction Processing

Presented by Yuanjia Zhang

PingCAP TiDB



Introduction

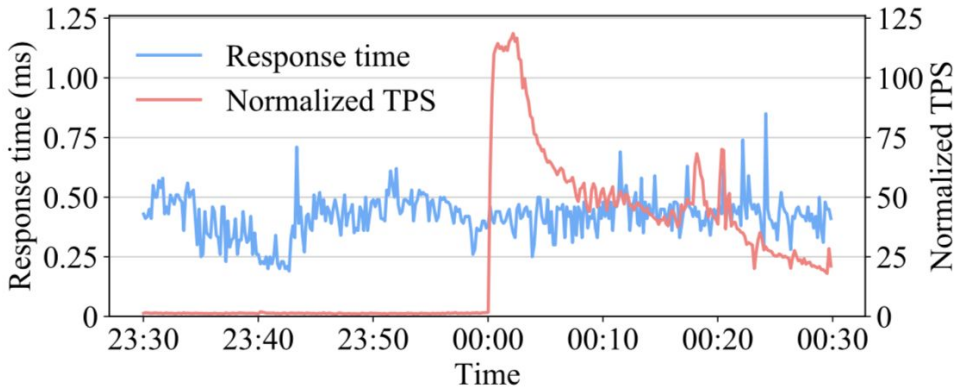
X-Engine is a OLTP storage engine based on an refined LSM-tree structure:

- storage engine
- OLTP
- refined LSM-tree

Key challenges

Key challenges for e-commerce workloads:

- **The tsunami problem**: drastic increase in TPS with the kickoff of major sales and promotional events.
- **The flood discharge problem**: large amount of hot records that can overwhelm system buffers.
- **The fast-moving current problem**: quick shift of the temperature (hot/warm/cold) of records.



Overview

- Storage layout
  - refined LSM-tree
  - tiered
- The read path
  - new extent format
  - multi-version metadata index
  - row/block cache
- The write path
  - multi-staged pipeline write
  - async-writes in transactions
- Flush and compaction
  - data reuse
  - FPGA-accelerated compactions

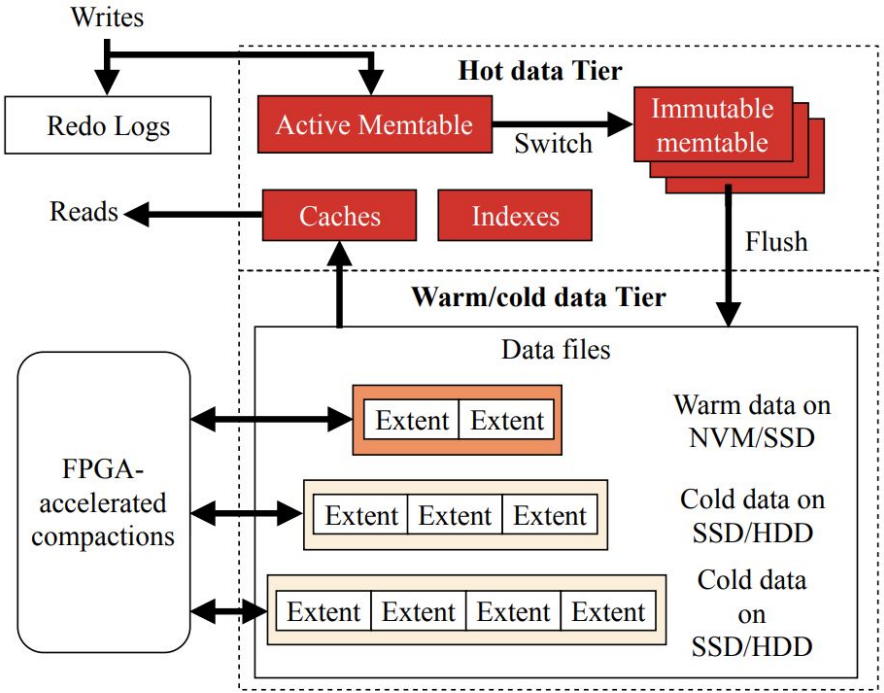


Figure 2: The architecture of X-Engine.

Extent format

Extent in X-Engine = SST in traditional LSM-Tree

- row-oriented style
- ordered
- multiple data blocks
  - data reuse in compaction
- schema data
  - convenient for DDL

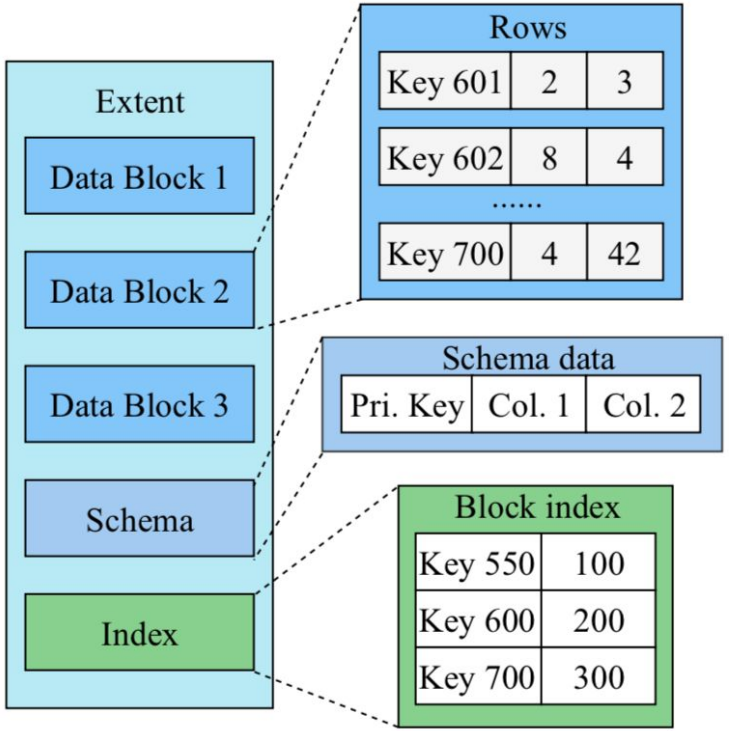


Figure 4: The layout of an extent.

Cache

The Row Cache

- optimized specifically
- LRU policy
- only keep the latest versions of records
- old versions are replaced during flush

The Block Cache

- missed point lookups + range lookups
- invalid after compaction

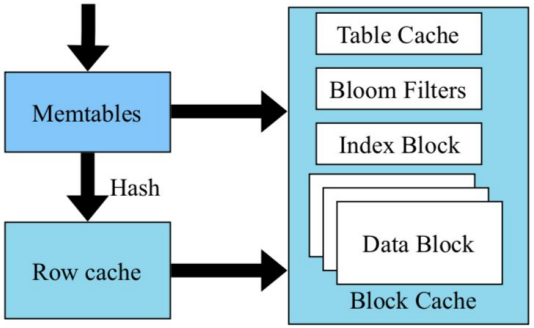
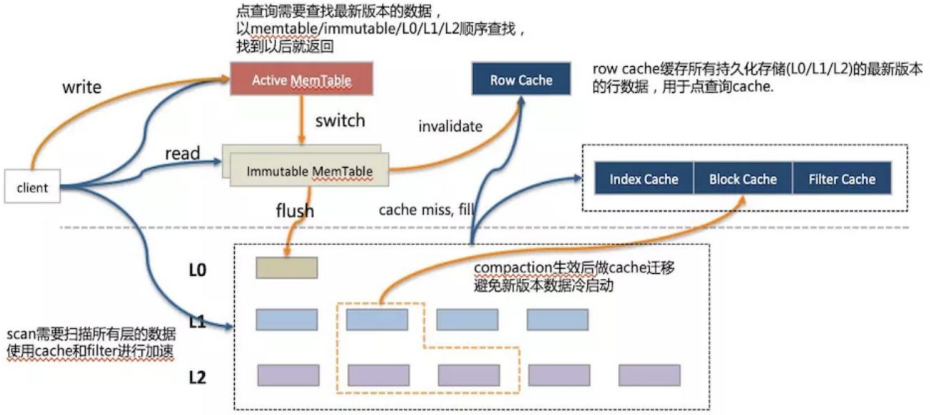
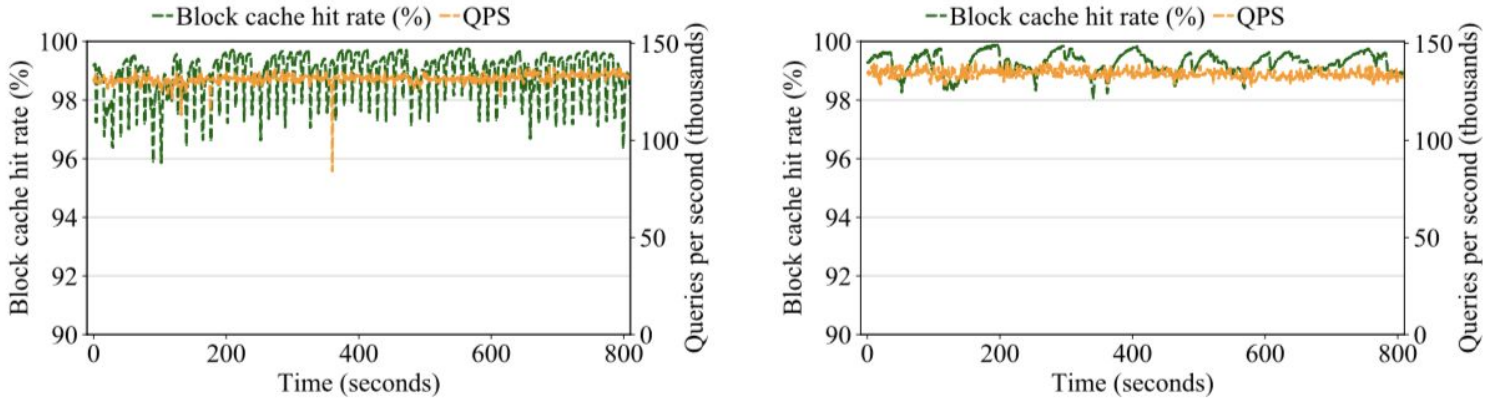


Figure 5: Structure of the caches.



Incremental replacement

Replace the old blocks in the cache with the newly merged ones during compaction, instead of simply evicting all old data blocks.



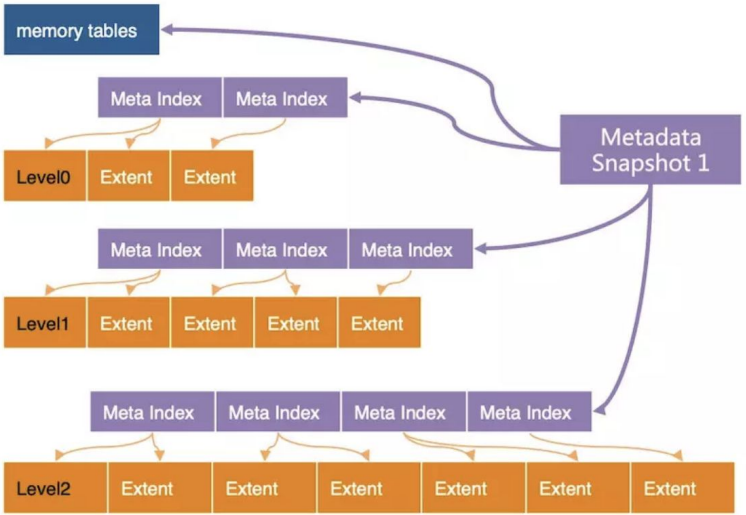
(a) Without incremental cache replacement.

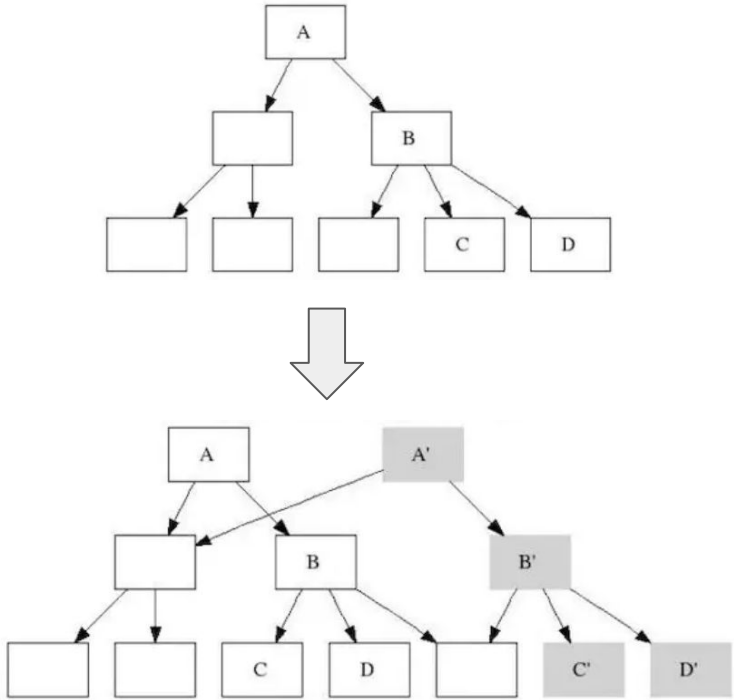
(b) Wth incremental cache replacement.

Multi-version Metadata Index

Metadata Index

- accelerate read.
- each sub-table has its associated metadata index
- modified when compaction and flush

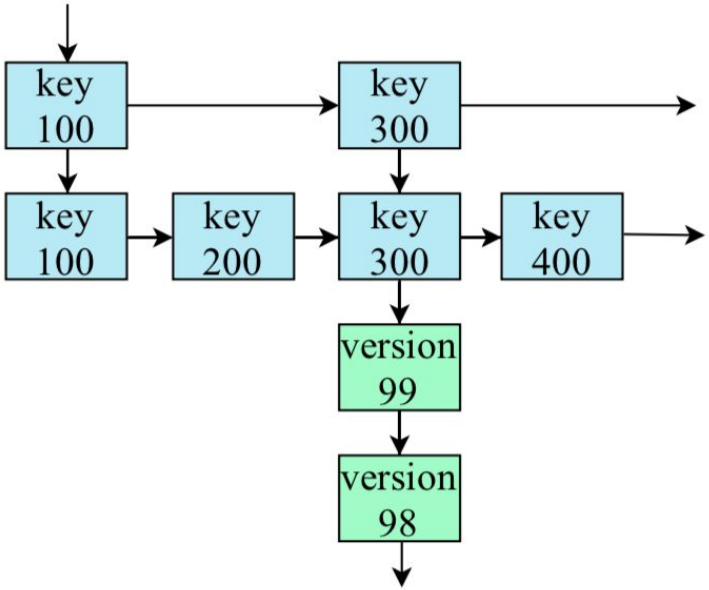




Multi-version memtable

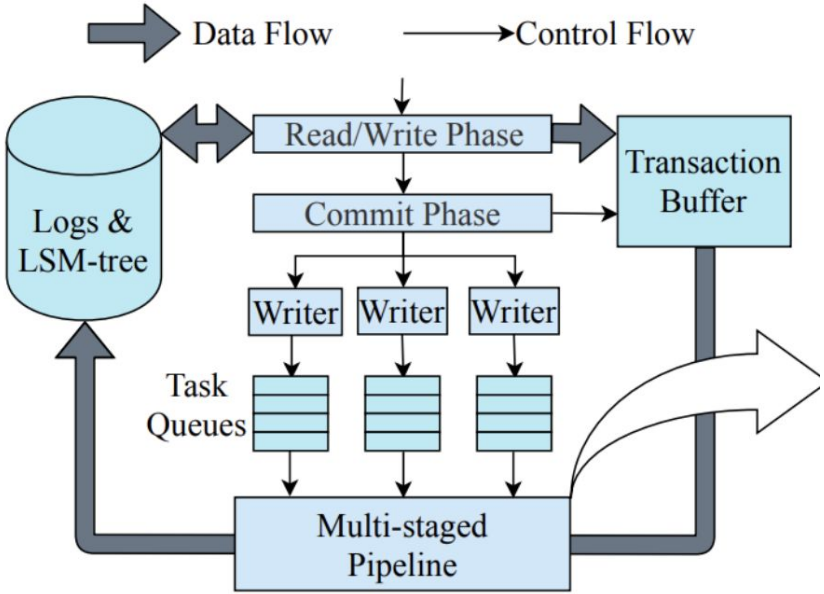
Memtable is implemented as a multi-version lock-free skiplist:

- key node
- version node



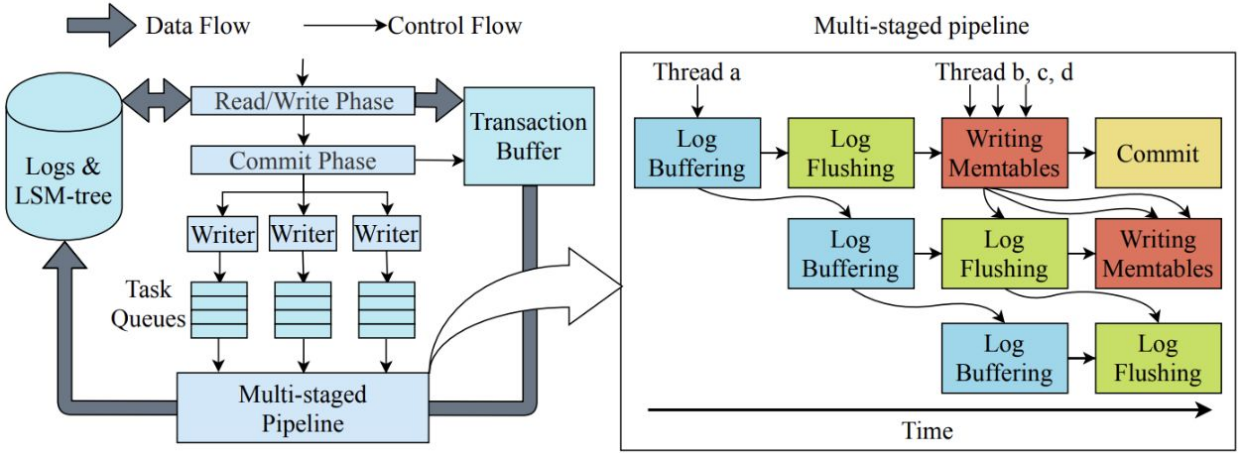
Async writes in transactions

- decouple the commits from their corresponding transactions
- group these commits for batch processing



Multi-staged pipeline

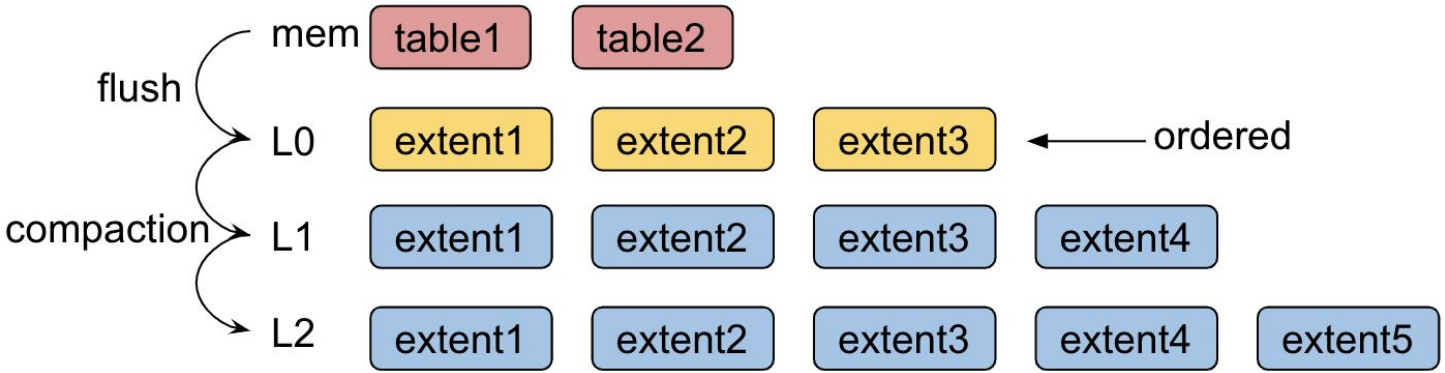
- Log Buffer
  - write WALs to memory-resident log buffers (memory access)
  - calculate CRC32 error-detecting codes (computation)
- Log Flushing (disk I/O)
- Writing Memtables parallelly (memory access)
- Commit (disk I/O)



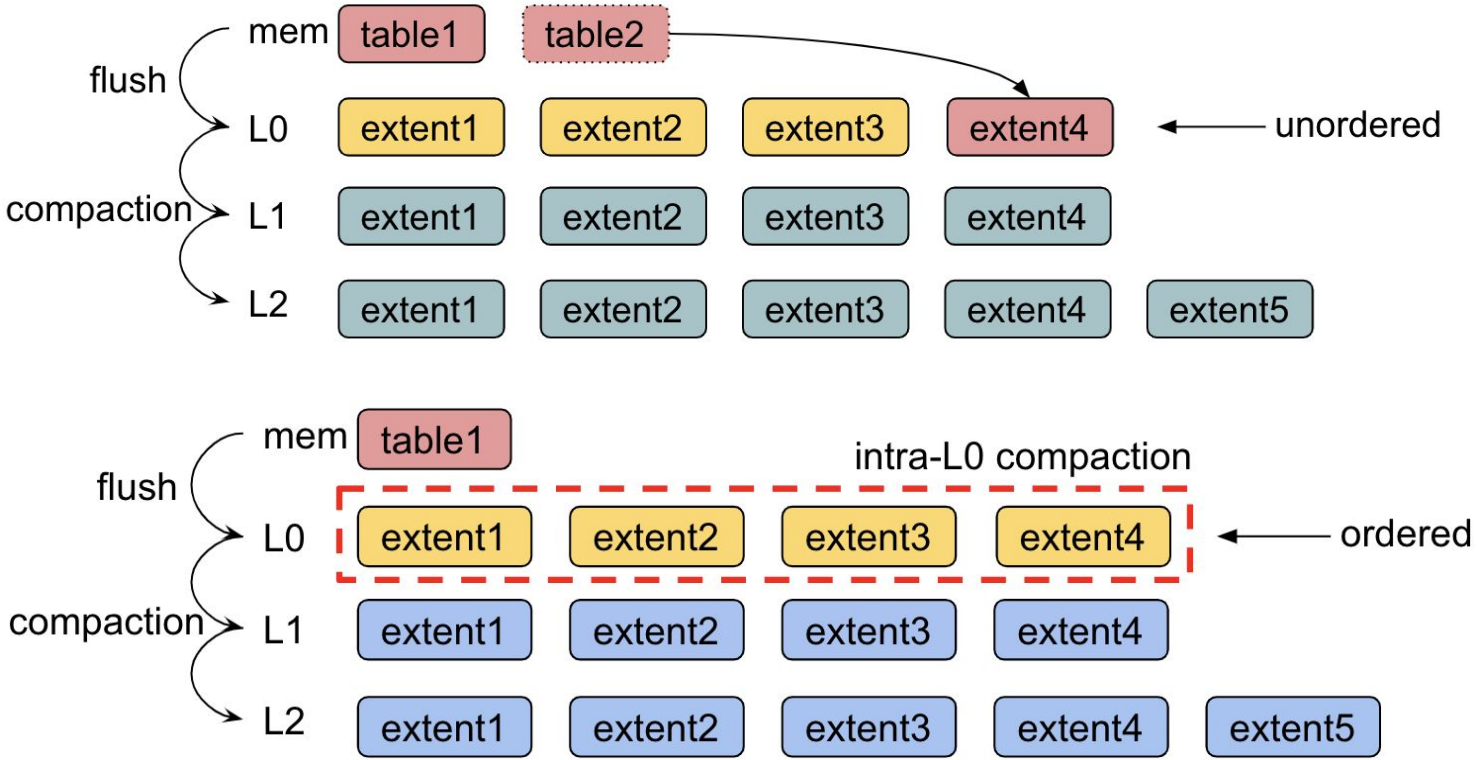
Fast flush of warm extents in Level0

Flush:

- immutable table -> extent
- hot -> warm

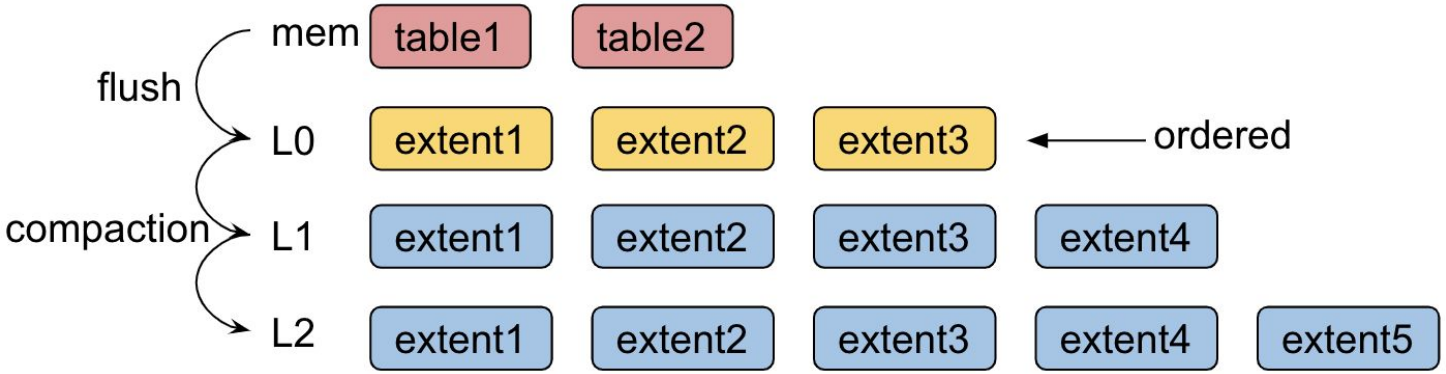


Fast flush of warm extents in Level0



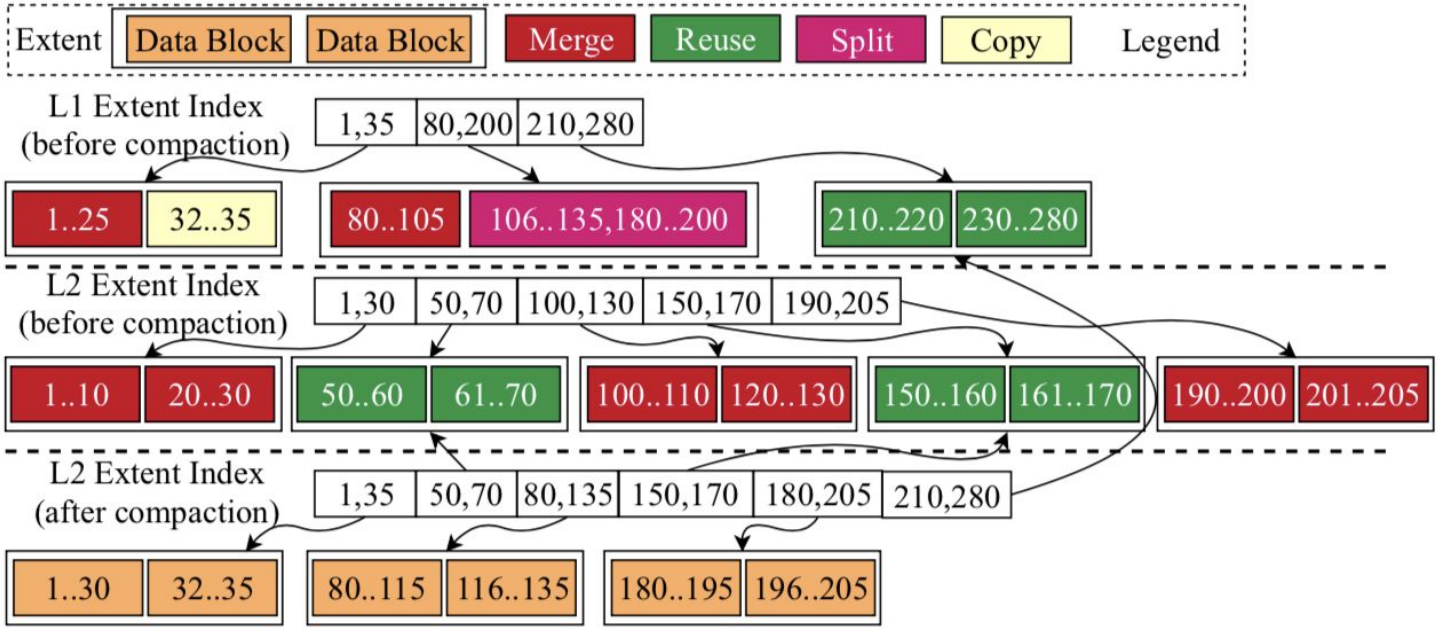
Accelerating compactions

- Data reuse
- FPGA offloading



Data reuse

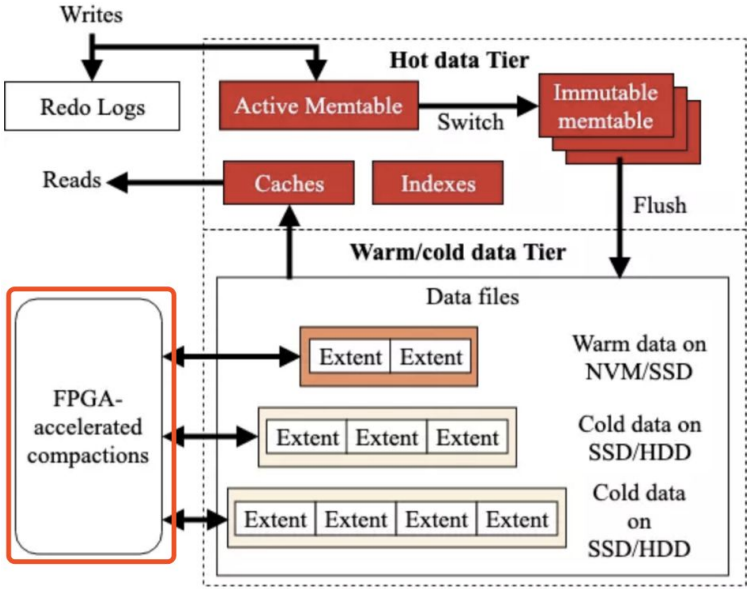
If the key range of an extent does not overlap with other extents, we reuse it by simply updating its corresponding metadata index.



FPGA offloading

Tasks are generated by CPUs and executed by FPGAs:

- Accelerate compaction
- Reduce the resource consumption on CPUs



Scheduling compactions

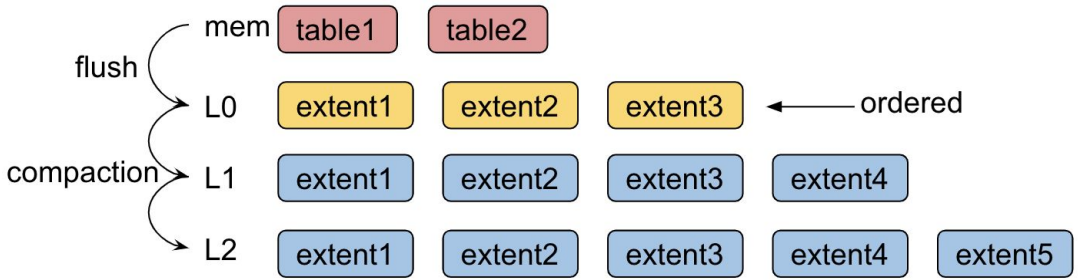
Five kinds of compactions:

- intra-Level0 compactions
- minor compactions (two adjacent levels except the largest level)
- major compactions (the largest level and the level above it)
- self-major (merging within the largest level)

Rule-based scheduling, it triggered when

- the total size or
- the total number of extents

of a level reach their predefined thresholds.



Compacting cold records

When a compaction is performed in a level, X-Engine

- selects the coldest extents with the number specified by a threshold(500),
- pushes them to the deeper level to do compaction.

An extent's temperature is calculated by its access frequency in a recent window. ML-based algorithms are investigated, and the problem is translated into a binary classification problem:

- a record is cold if it a has never been accessed in a recent window,
- otherwise it is considered warm.

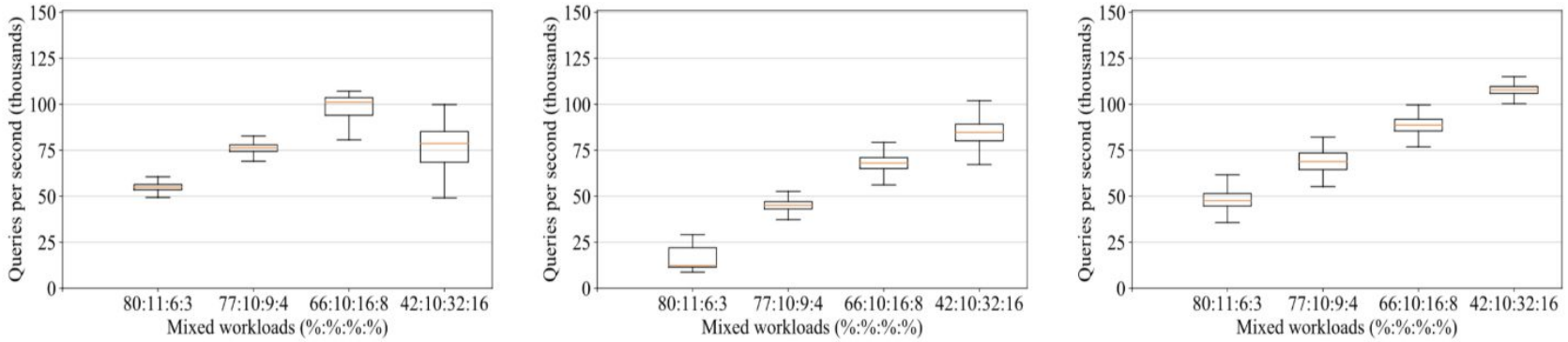
Summary of optimizations

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.	

Evaluation

The 42:10:32:16 mixture is very close to the mixture observed during the Singles' Day



(a) InnoDB (b) RocksDB (c) X-Engine

Figure 9: Throughput of MySQL with the three storage engines using the e-commerce workloads with different mixtures of point lookups, range scans, updates, and inserts. The y-axis of these figures are the same.

Evaluation

FPGA improves throughput by 27% by reduced variances.

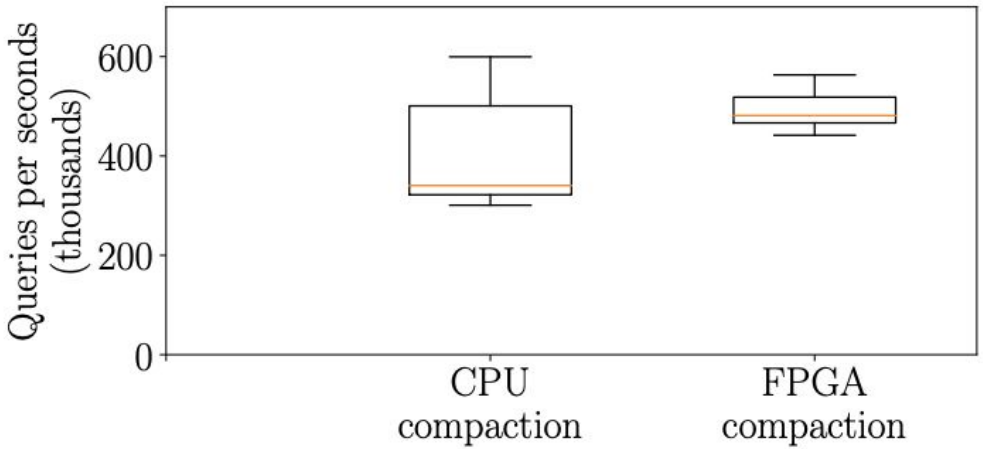
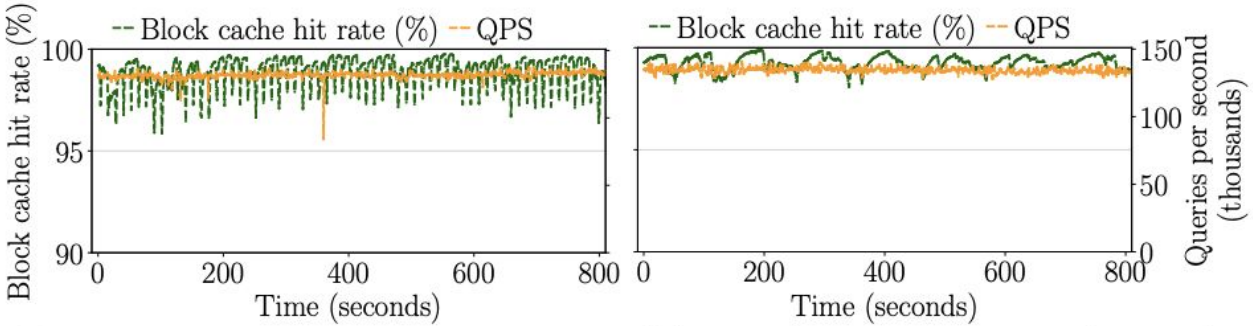


Figure 15: Throughput of MySQL (X-Engine) with and without FPGA offloading for compactions.

Evaluation

The incremental cache replacement significantly reduces the variations of the block cache hit rate.



(a) Without incremental cache replacement. **(b) Wth incremental cache replacement.**

Figure 17: The block cache hit rates of X-Engine while processing the e-commerce workload. These two figures share the same axes.

PingCAP

Q&A

Thank You !