Storage Engine of InfluxDB v0.13

simpcl
2016.9.12

Time Series Data

- 异常检测
- 消息
- 个性化
- 股票交易
- 市政基础设施管理
- GPS服务
- 量子物理研究
- 销售点系统
- 制造与家庭自动化
- 运输和物资保障

问题

- Persistent Storage or Cache？
- Ordered or Disordered？
- Normal Flat File？
- Clustered Index or Secondary Indexes？
- B-Tree or LSM-Tree？

For Writes

- Write-mostly is the norm; perhaps 95% to 99% of operations are writes, sometimes higher.
- Writes are almost always sequential appends; they almost always arrive in time order. There is a caveat to this.
- Writes to the distant past are rare. Most measurements are written within a few seconds or minutes after being observed, in the worst case.
- Updates are rare.
- Deletes are in bulk, beginning at the start of history and proceeding in contiguous blocks. Deletes of individual measurements or deletes from random locations in history are rare. Efficient bulk deletes are important; as close to zero cost as possible. Non-bulk deletes need not be optimal.
- Due to the above, an immutable storage format is potentially a good thing. As a further consequence of immutable storage, a predefined or fixed schema may be a problem long-term.

For Reads

- Data is much larger than memory and rarely read, so caching typically doesn't work well; systems are often IO-bound.

- Reads are typically logically sequential per-series, ascending or descending.

- Concurrent reads and reads of multiple series at once are reasonably common.

RDBMS & NoSQL

- MySQL
- MongoDB
- Cassandra/Riak
- HBase
- ElasticSearch

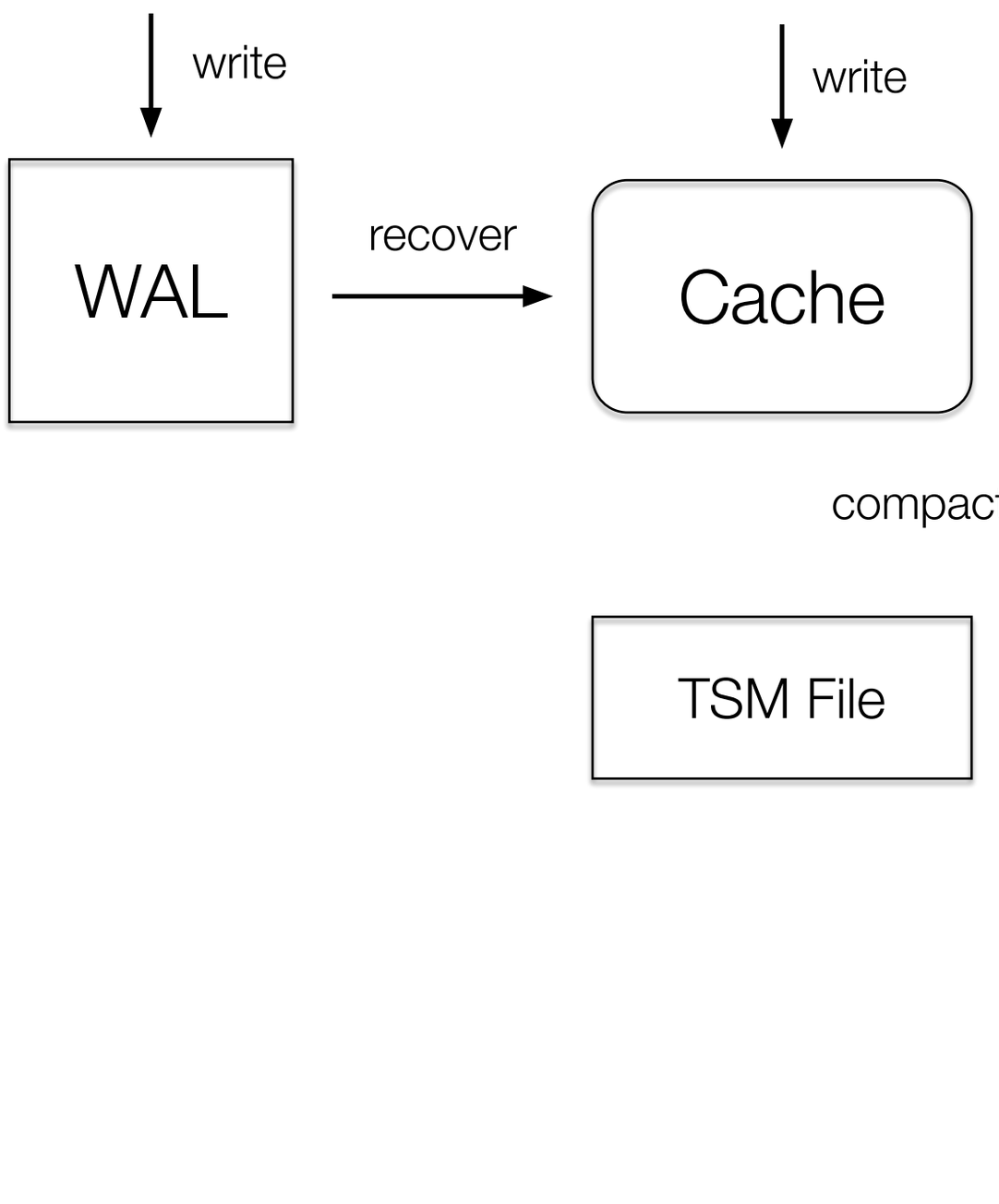
InfluxDB Key Concepts

- Database
- Measurement
- Tags
- Fields and Values
- Series

问题

- 一个 Series 一个Measurement？
- Tags 是否可以用 Secondary Indexes？
- 按 Key 排序，按时间排序？
- 如何界定 Tags 与 Fields？
- 如何规划 Measurement？

tsm1

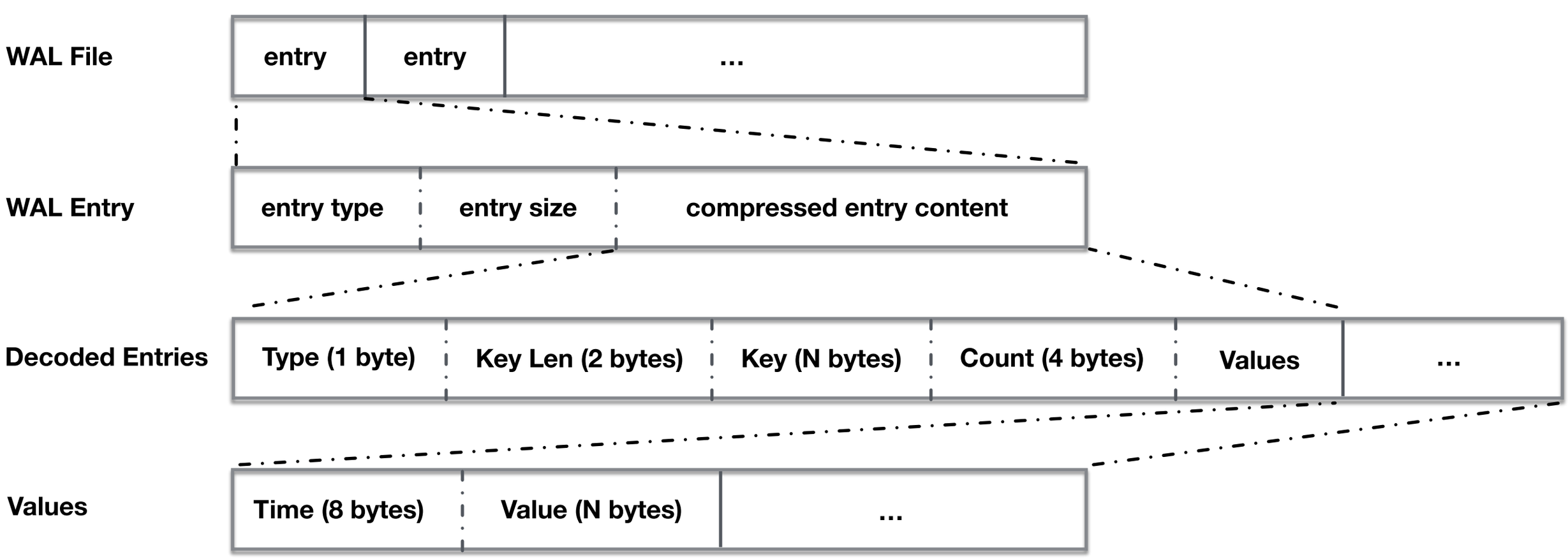

write
write
WAL
recover
Cache
compact
TSM File
compact
TSM File
TSM File
TSM File
compact

tsm1 Key Concepts

- WAL
- Cache
- TSM Files: [generation]-[sequence].tsm
- Key: [Measurement]+[Tag List]+[Field Key] ?
- Block ?
- Compaction
- Generation
- Sequence

WAL

- Write Ahead Log
- 2 MB per file
- The file is fsync’d for every write

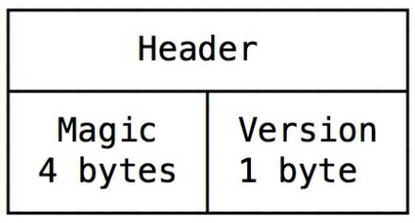


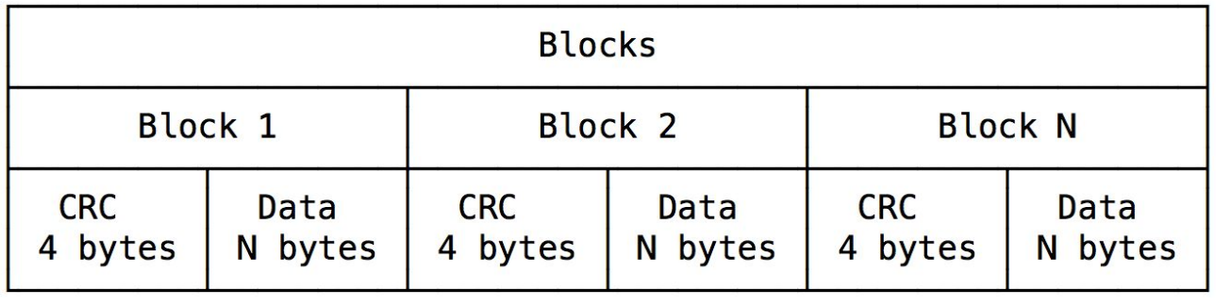
Cache

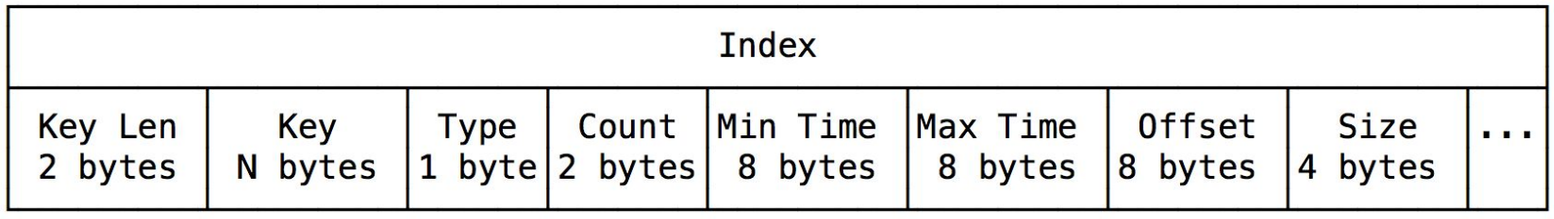
- Hash Table
- 数据先写缓存，然后转换为TSM File
- 转换按照时间限制和内存限制两个维度
- 转换过程称为Cache Compaction
- Cache Compaction过程中，使用两个Hash Table

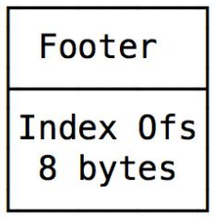
TSM File


Header
5 bytes
Blocks
N bytes
Index
N bytes
Footer
8 bytes


Header
Magic
4 bytes
Version
1 byte


Blocks
Block 1
Block 2
Block N
CRC
4 bytes
Data
N bytes
CRC
4 bytes
Data
N bytes
CRC
4 bytes
Data
N bytes


Index
Key Len
2 bytes
Key
N bytes
Type
1 byte
Count
2 bytes
Min Time
8 bytes
Max Time
8 bytes
Offset
8 bytes
Size
4 bytes
...


Footer
Index Ofs
8 bytes

Compaction

- go e.compactCache()
- go e.compactTSMFull()
- go e.compactTSMLevel(true, 1)
- go e.compactTSMLevel(true, 2)
- go e.compactTSMLevel(false, 3)

Q & A

Thank You !