Constructing and Analyzing
the LSM Compaction Design Space

Subhadeep Sarkar
Zichen Zhu

Dimitris Staratzis
Manos Athanassoulias
Log-Structured Merge-tree
LSM-tree
LSM-tree

Other databases mentioned:
- HBase
- Tarantool
- Bigtable
- SQLite
- Riak
- Scylla
- Cassandra
- RocksDB
- DynamoDB
- Accumulo
- InfluxDB
- QuasarDB
Why LSM?

- Fast writes
- Competitive reads
- Good space utilization
fast writes  competitive reads  good space utilization
COMPACTATION

- Space amplification
- Write performance
- Scan performance
- Lookup performance
- Write amplification
- Delete performance
workload

LSM tuning

COMPACTION

performance
Our **Goal**

1. **Roadmap to pick compactions**
2. **Answer to complex design questions**
Our Goal

1. break the black box
2. learn from 2000+ experiments
buffer
buffer

level 1

level 2

compaction
compaction?
How much data to compact at once?

How many runs per level?
How much data to compact at once?

How many runs per level?
How many runs per level?

How much data to compact at once?
How much data to compact at once?

How many runs per level?
What are the **design choices**?

How does a choice **affect performance**?

- How many runs per level?
- How much data to compact at once?
What are the **design choices**?

How does a choice **affect performance**?
What are the design choices?

How does a choice affect performance?
What are the **design choices**?

How does a choice **affect performance**?
1. **How** to organize the data on device?

2. **How much** data to move at-a-time?

3. **Which** block of data to be moved?

4. **When** to re-organize the data layout?
Data Layout

1. How to organize the data on device?

Compaction granularity

2. How much data to move at-a-time?

Data movement policy

3. Which block of data to be moved?

Trigger

4. When to re-organize the data layout?
Data Layout

number of runs per level
1 Data Layout

number of runs per level

leveling [eager]

 tiering [lazy]
Data Layout

number of runs per level

leveling

1-leveling

L-leveling

tiering
Compaction Granularity

data moved per compaction
Compaction Granularity

data moved per compaction
Compaction Granularity

data moved per compaction

levels

files
Compaction Granularity

data moved per compaction

levels

files

sorted runs in a level
Data Movement Policy

*which data to compact*

files
Data Movement Policy

*which data to compact*

- round-robin
- minimum *overlap with parent* level
- file with most *tombstones*
- *coldest* file
Compaction Trigger

invoking the compaction routine

level saturation
Compaction Trigger

invoking the compaction routine

level saturation
Compaction Trigger

invoking the compaction routine

- level saturation
- number of sorted runs
- age of a file
- space amplification
1. Data Layout
2. Compaction Granularity
3. Data Movement Policy
4. Compaction Trigger
Data Layout  
Compaction Granularity  
Data Movement Policy  
Compaction Trigger

Any Compaction Algorithm
<table>
<thead>
<tr>
<th>Database</th>
<th>Data layout</th>
<th>Compaction Trigger</th>
<th>Compaction Granularity</th>
<th>Data Movement Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Level saturation</td>
<td>#Sorted runs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>File staleness</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Space amp.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tombstone-TTL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sorted run</td>
<td>File (single)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>File (multiple)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Round-robin</td>
<td>Least overlap (+1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Least overlap (+2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Coldest file</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Oldest file</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tombstone density</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expired TS-TTL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N/A (entire level)</td>
<td></td>
</tr>
<tr>
<td>RocksDB [30], Monkey [22]</td>
<td>Leveling / 1-Leveling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Tiering</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LevelDB [32], Monkey (J.) [21]</td>
<td>Leveling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SlimDB [47]</td>
<td>Tiering</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dostoevsky [23]</td>
<td>$L$-leveling</td>
<td>✓, ✓</td>
<td>✓, ✓</td>
<td>✓, ✓</td>
</tr>
<tr>
<td>LSM-Bush [24]</td>
<td>Hybrid leveling</td>
<td>✓, ✓</td>
<td>✓, ✓</td>
<td>✓, ✓</td>
</tr>
<tr>
<td>Lette [51]</td>
<td>Leveling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Silk [11], Silk+ [12]</td>
<td>Leveling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>HyperLevelDB [35]</td>
<td>Leveling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PebblesDB [46]</td>
<td>Hybrid leveling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cassandra [8]</td>
<td>Tiering</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>WiredTiger [62]</td>
<td>Leveling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>X-Engine [34], Leaper [63]</td>
<td>Hybrid leveling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>HBase [7]</td>
<td>Tiering</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AsterixDB [3]</td>
<td>Leveling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Blueprint for **Experiments**

10 compaction strategies

- primitives
- workloads
  - [distribution + composition]
- LSM tuning

612 metrics
Compacting data at smaller granularity reduces data movement.
Compacting data at smaller granularity reduces data movement.
Compacting data at smaller granularity reduces data movement.

Tiered data layout has the highest write throughput but also the highest tail write latency.

![Compaction strategies graph]
Compacting data at smaller granularity reduces data movement.

Tiered data layout has the highest write throughput but also the highest tail write latency.

Hybrid data layouts dominate point lookup performance.
Compacting data at smaller granularity reduces data movement.

Tiered data layout has the highest write throughput but also the highest tail write latency.

Hybrid data layouts dominate point lookup performance.

For update-intensive workloads, tiering dominates the performance space.
Compacting data at smaller granularity reduces data movement.

Tiered data layout has the highest write throughput but also the highest tail write latency.

Hybrid data layouts dominate point lookup performance.

For update-intensive workloads, tiering dominates the performance space.

The relative benefits of compaction strategies are marginally affected by LSM-tuning.
Summary

Compaction is key to LSM-performance.

Compaction as first-order design primitives.

Guidelines to design and tuning through experiments.

Thank You!