Constructing and Analyzing the LSM Compaction Design Space

Subhadeep Sarkar         Dimitris Staratzis
Zichen Zhu              Manos Athanassoulis
Log-Structured Merge-tree
LSM-tree
LSM-tree
Why LSM?
Why **LSM**?

- fast writes
- competitive reads
- good space utilization
fast writes  competitive reads  good space utilization
fast writes  competitive reads  good space utilization

COMPACTION
COMPACTATION
COMPACTTION

- space amplification
- write performance
- scan performance
- lookup performance
- write amplification
- delete performance
COMPACTION
COMPACTION
workload → COMPACTION → performance

LSM tuning
Our **Goal**
Our Goal

1. Roadmap to pick compactions
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 2614
buffer

level 1
buffer

level 1
buffer

level 1
compaction?
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?
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?
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
Data Layout

number of runs per level
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
Compaction Granularity

data moved per compaction
Compaction Granularity

data moved per compaction

levels
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
Compaction Granularity

data moved per compaction

levels

files

sorted runs in a level
Data Movement Policy
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
Compaction Trigger

invoking the compaction routine
Compaction Trigger

invoking the compaction routine

level saturation
Compaction Trigger

invoking the compaction routine

level \textit{saturation}
Compaction Trigger

invoking the compaction routine

level saturation

number of sorted runs

age of a file

space amplification
Data Layout
Compaction Granularity
Data Movement Policy
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>File staleness</td>
</tr>
<tr>
<td>RocksDB [30], Monkey [22]</td>
<td>Leveling / 1-Leveling</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>Lethe [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
Blueprint for *Experiments*

10 compaction strategies
Blueprint for Experiments

10 compaction strategies

612 metrics
Blueprint for Experiments

10 compaction strategies

primitives

612 metrics
Blueprint for **Experiments**

10 compaction strategies

primitives

workloads

[distribution + composition]

612 metrics
Blueprint for **Experiments**

- **10** compaction strategies
  - **primitives**
  - **workloads**
  - **LSM tuning**

- **612** metrics

[diagram showing distribution and composition]
Compacting data at smaller granularity reduces data movement.
Compacting data at smaller granularity reduces data movement.
Compacting data at smaller granularity reduces data movement.
Compacting data at smaller granularity reduces data movement.

![Chart showing tail write latency for different compaction strategies.](chart.png)
Compacting data at smaller granularity reduces data movement.

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

![Graph showing tail write latency for different compaction strategies](image)

- Compaction strategies: Full, LO+1, LO+2, RR, Cold, Old, Tier, 1-Lvl
- Tail write latency (ms): 0, 5, 10, 15, 20, 25
- Comparison: Tier is 5x higher than the other strategies.
Compacting data at smaller granularity reduces data movement.

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

Compaction is **key to LSM-performance**.
Summary

Compaction is **key to LSM-performance.**

Compaction as first-order **design primitives.**
Summary

Compaction is **key to LSM-performance**.

Compaction as first-order **design primitives**.

**Guidelines to design and tuning** through experiments.
Compaction is **key to LSM-performance.**

Compaction as first-order **design primitives.**

**Guidelines to design and tuning** through experiments.

**Summary**

Thank You!