Hailstorm
Disaggregated Compute and Storage for Distributed LSM-based Databases

Laurent Bindschaedler, Ashvin Goel, Willy Zwaenepoel
Hailstorm Improves Compute and Storage Load Balance in Distributed Databases

⇒ higher throughput and better resource utilization
Preview - Production Trace on MongoDB

Throughput (KOps/s)

- Without Hailstorm
- With Hailstorm

3X

Video of Presenter
Outline

1. Background
   A. Distributed Databases
   B. Load Imbalance in Distributed Databases
   C. Shard Rebalancing

2. Hailstorm Architecture

3. Evaluation

4. Conclusion
Outline

1. Background
   A. Distributed Databases
   B. Load Imbalance in Distributed Databases
   C. Shard Rebalancing

2. Hailstorm Architecture

3. Evaluation

4. Conclusion
Background - Distributed Databases

- Client
- Database
- Instance 0
  - Storage Engine
  - Local Disk
- Instance 1
  - Storage Engine
  - Local Disk
- Instance N
  - Storage Engine
  - Local Disk

Video of Presenter
Background - GET Operation

Client

GET(k_i) \rightarrow v_i

Instance 0

Storage Engine

Local Disk

Instance 1

Storage Engine

Local Disk

Instance N

Storage Engine

Local Disk

…

Database

GET(k_i)

…and

Video of Presenter
Background - PUT Operation

Client → PUT(k<sub>i</sub>,v<sub>i</sub>) → Database

<table>
<thead>
<tr>
<th>Instance 0</th>
<th>Instance 1</th>
<th>Instance N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage Engine</td>
<td>Storage Engine</td>
<td>Storage Engine</td>
</tr>
<tr>
<td>Local Disk</td>
<td>Local Disk</td>
<td>Local Disk</td>
</tr>
</tbody>
</table>

WRITE(v<sub>i</sub>)

Video of Presenter
Background - RANGE Operation

Client

Database

RANGE\((k_i, k_j) \rightarrow \{v_i...v_j\}\)

Instance 0

Storage Engine

Local Disk

Instance 1

Storage Engine

Local Disk

Instance N

Storage Engine

Local Disk

RANGE\((k_i, k_l)\)

SCAN\((k_i, k_l)\)

SCAN\((k_l, k_j)\)

RANGE\((k_l, k_j)\)

...
Load Imbalance Cause #1: Skew

- Client
- Database
- Storage Engine
- Local Disk

Instance 0
- Storage Engine
- Local Disk

Instance 1
- Storage Engine
- Local Disk

Instance N
- Storage Engine
- Local Disk

Video of Presenter
Load Imbalance Cause #2: LSM Compaction

Database

Instance 0
Storage Engine
Local Disk

Instance 1
Storage Engine
Local Disk

Instance N
Storage Engine
Local Disk

...
Background - Shard Rebalancing

⇒ Increase load on overloaded instances
⇒ Slow & often too late
CPU and Storage Load Imbalance in LSM Databases

- Skew
- Compactions

Diagram:
- Database
  - Instance
    - Storage Engine
      - Local Disk

Video of Presenter
Outline

1. Background
   A. Distributed Databases
   B. Load Imbalance in Distributed Databases
   C. Shard Rebalancing

2. Hailstorm Architecture

3. Evaluation

4. Conclusion
Hailstorm: Disaggregate Storage & Compute

Scale each resource independently

**Fine-grained storage pooling**
- Pool disks within a rack
- Split data in blocks and spread blocks uniformly
  ⇒ Achieve storage load balance

**Compaction offloading**
⇒ Improve compute load balance

Video of Presenter
Storage Architecture: Hailstorm Filesystem
The Hailstorm Distributed Filesystem

Drop-in replacement for local filesystem
⇒ Supports fine-grained storage pooling & compaction offloading

Blocks are spread in a deterministic order
⇒ Storage engines locate and access data independently

Compaction offloading is efficient
⇒ Blocks are everywhere
⇒ Just need file metadata to locate blocks
Disk Pooling & Data Block Spreading

Database

Instance 0
Storage Engine

Instance 1
Storage Engine

Instance N
Storage Engine

Hailstorm

Local Disk

Video of Presenter
Compaction Offloading

Instance 0
Storage Engine

Instance N
Compaction Process
Storage Engine

Hailstorm

Local Disk

Local Disk

Local Disk

Video of Presenter

Local Disk
Outline

1. Background
   A. Distributed Databases
   B. Load Imbalance in Distributed Databases
   C. Shard Rebalancing

2. Hailstorm Architecture

3. Evaluation

4. Conclusion
Evaluation

8 16-core machines
32GB of RAM, Intel S3500 SSD, 40GigE switch

MongoDB (key-value)
TiDB (SQL ACID)

Baseline vs Hailstorm

Uniform vs Zipfian (skewed)

RocksDB storage

Video of Presenter
MongoDB - Write-Intensive YCSB Throughput

YCSB A
(50% reads, 50% writes)

Throughput (KOps/s)

<table>
<thead>
<tr>
<th>Throughput</th>
<th>Uniform</th>
<th>Baseline</th>
<th>Zipfian</th>
<th>Hailstorm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>□</td>
<td></td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>50</td>
<td>□</td>
<td></td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>100</td>
<td>□</td>
<td></td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>150</td>
<td>□</td>
<td></td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>200</td>
<td>□</td>
<td></td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>

2.2X

Video of Presenter
MongoDB - Read-Intensive YCSB Throughput

YCSB B
(95% reads, 5% writes)

Throughput (KOps/s)

0 50 100 150 200

Uniform  Zipfian

 Baseline  Hailstorm

+46%

Video of Presenter
 MongoDB - Scan-Intensive YCSB Throughput

YCSB E
(95% scans, 5% writes)

Throughput (KOps/s)

Baseline
Hailstorm

0 5 10

Uniform Zipfian

22X

Video of Presenter
### TiDB - TPC-C & TPC-E

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Standalone TiDB</th>
<th>TiDB over Hailstorm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TPC-C</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tpmC</td>
<td>32,184</td>
<td>50,178</td>
</tr>
<tr>
<td>$ / tpmC*</td>
<td>3.10</td>
<td>2.00</td>
</tr>
<tr>
<td><strong>TPC-E</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tpsE</td>
<td>277.3</td>
<td>408.1</td>
</tr>
<tr>
<td>$ / tpsE*</td>
<td>360.60</td>
<td>245.05</td>
</tr>
</tbody>
</table>

*Estimated total system cost: $100,000.*

1.5X improvements
Additional Results in Paper

• Complete YCSB benchmark
• Throughput over time
• Response latency
• Large datasets
• Shard rebalancing
• Comparison with HDFS
• Performance breakdown
• Hailstorm for B-trees
Conclusion

Hailstorm improves load balance in LSM-based databases
⇒ Higher throughput & resource utilization

Hailstorm works with existing databases

Key idea: compute and storage disaggregation
• Fine-grained storage pooling
• Compaction offloading
Thank you and stay tuned for updates!

Hailstorm

Disaggregated Compute and Storage for Distributed LSM-based Databases