Replication-driven Live Reconfiguration for Fast Distributed Transaction Processing

Xingda Wei, Sijie Shen,
Rong Chen, Haibo Chen

Institute of Parallel and Distributed Systems
Shanghai Jiao Tong University, China
Transactions: Key pillar for many systems

Alibaba.com

$9.3 billion/day

PayPal

11.6 million payments/day
Skewed workloads hurt performance

- Modern in-memory transactions are fast
- TXs can scale-out on balanced workloads
- But can fail with skewed workloads
Skewed workloads hurt performance

- Unbalanced workload: Idle worker

Partition plan:

| P0, P1 | N0  
| P2    | N1  

Throughput

Latency

Task Queues

Data partitions
Skewed workloads hurt performance

- Unbalanced workload: Idle worker
- More distributed TX ratio, aborts

Sample data

| P0, P1 | N0
| P2  | N1

Data partitions

Partition plan

Latency

Task Queues

Throughput

Distributed TXs
Solution: Live Reconfiguration

E-store\textsuperscript{[VLDB’14]} repartitions the database to balance the workload on each server.
Efficient live reconfiguration

- Generate the re-partition plan **is fast**
- How to **lively** migrate the data?

Sample data

<table>
<thead>
<tr>
<th>P0</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
</table>

Data partitions

**Partition plan**

<table>
<thead>
<tr>
<th>P0, P1</th>
<th>N0</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>N1</td>
</tr>
</tbody>
</table>

**Re-partition plan**

| P1 => Node 1 |

**Task Queues**

- P0
- P1
- P2
E-store uses Squall\textsuperscript{[SIGMOD’15]} to migrate data.

1. Commit the plan (i.e., P is at N1)
2. Pull data on-demand or asynchronously
SOL#1 Migrating data with Post-copy

- E-store uses Squall[^SIGMOD'15] to migrate data

1. Blocked by missing data. (Possibly many times)
2. Aborted by migrated data.
Post-copy is unsuitable for fast TXs

- Using Squall to balance skewed TPC-C for DrTM+R[ Laosys’16 ]
- Due to many affected TXs

<table>
<thead>
<tr>
<th>Time (second)</th>
<th>Throughput (MTX/second)</th>
<th>Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5.15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1.4</td>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>2.45</td>
<td>0.75</td>
<td>2.45</td>
</tr>
<tr>
<td>6.25</td>
<td>1.0</td>
<td>6.25</td>
</tr>
</tbody>
</table>

Data migration

vs. before 81%↓
vs. before 4.3X↑
SOL#2 Migrating data with Pre-copy

- TXs can **safely** access data at sources

Manager

Start

Data migration

Commit

Done

Node0

Node1

TX

\[ \ldots \]

Pull data

\[ P \Rightarrow N1 \]

1. Migrating **all** data to destination
2. Commit the new plan destination
Pre-copy is not free

- Pre-copy requires **tracking & syncing** dirty data

① Migrating **all** data to destination  
② Commit the new plan
Pre-copy is not free

- Pre-copy requires **tracking** & **syncing** dirty data

1. **Longer** migration time & **larger** data transmissions
2. TX’s tracking overhead
DrTM+B: Fast & Seamless reconfiguration

- Data migration is the most **costly** part

- **Avoids possible data migration** by preferring existing data replicas

- **Pre-copy based approach**: minimizing costs
  - Avoids above shortcomings by leveraging existing fault tolerance mechanisms, i.e. logging
Outline

- System architecture
- Reduce data transfer with existing backup
- Data-migration process
- Implementations & Evaluations
Outline

- System architecture
- Reduce data transfer with existing backup
- Data-migration process
- Implementations & Evaluations
System architecture

- Sharded & replicated memory store
  - FaSST[^OSDI’16], DrTM+R[^Eurosys’16], FaRM[^SOSP’15]

- Primary-backup synced with TX’s logs
  - Logs are processed *asynchronously* for efficiency
Reduce data transfer with replicas

- R-partition plan assigns **hot data** -> cold server
- Yet data migration is **costly**
Reduce data transfer with replicas

- R-partition plan assigns **hot data -> cold server**
- Direct loads to **server with backup data**

**Partition plan**

**Re-partition plan**

**Task Queues**

---

Sample data

<table>
<thead>
<tr>
<th>P0, P1</th>
<th>N0</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>N1</td>
</tr>
</tbody>
</table>

**P1 -> Node 1**

---

**Primary**

- P0
- P1

**Backup**

- P2

**Node 0**

---

**Node 1**

- P2
- P1
- P0
Reduce data transfer with replicas

- R-partition plan assigns **hot data -> cold server**
- Direct loads to **server with backup data**

---

**Partition plan**

<table>
<thead>
<tr>
<th>P0, P1</th>
<th>N0</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>N1</td>
</tr>
</tbody>
</table>

**Re-partition plan**

| P1 => Node 1 |

---

**Task Queues**

**Primary**

- P0
- P1

**Backup**

- P2
- P1

**Node0**

- P2
- P1

**Node1**

- P1
- P0
Reduce data transfer with replicas

- Resource is **sufficient** in skewed workloads
- Direct loads to **server with backup data**
- Data at backup’s server does not need migration
Improve pre-copy with log forwarding

- Pre-copy requires tracking & syncing dirty data during data migration
Improve pre-copy with log forwarding

- Pre-copy requires **tracking** & **syncing** dirty data during data migration
  
  ➡ **Logs** tracks the dirty data

- **Forwards** log with data migration
Data migration phase

Manager

Node0

Node1

NodeX

Start

Data migration

Commits

TX

Log

Pull data

Process P's data

1. Forwards logs to destinations
2. Pulls data from sources
Commit phase

Manager

Node0

Node1

NodeX

Start — Data migration — Commit — Done

③ Collect log offsets
④ Wait for all pending logs to be processed
⑤ Commit the new plan

P

Log

Commits

TX

Pull data

Processed all log

Log states

Processed P's data

P => N1

Collect log offsets

Wait for all pending logs to be processed

Commit the new plan
Challenge: Overloaded primary

Manager

Node0

Node1

NodeX

Start  Data migration

TX reqs

Data pulling reqs

Pull data

Primary has become overloaded

➡ Competing CPU resources
Optimization: parallel data fetching

Parallel fetching data from all replicas

As backup contains nearly the same content as primary
Challenge: Stale backup

- Logs are **asynchronously** processed at backups
- Directly fetching from backup causes inconsistency
Pre-sync before parallel data fetching

Manager

Node0

Node1

NodeX

Start

Data migration

Pending logs are processed

1. Forwards logs
2. Syncs log states
3. Wait for pending logs
4. Parallel data fetching
Other Specific Implementation

- Based on DrTM+R\textsuperscript{[Eurosys'16]}
- Cooperative commit protocol
- Replication-aware planner
- Workload monitor
- Fault tolerance
Evaluations

Platform:
- 6-node local cluster
- 3-way replication enabled

Benchmarks:
- TPCC & Smallbank with 2 skewed settings\(^1\)

Comparison
- Squall\(^{\text{SIGMOD'15}}\) on DrTM+R

\(^1\) Low: 60% accesses goes to 1/3 warehouse
High: 40% accesses goes to 4 warehouses on one node, the rest is the same as low skew
Performance of load balance

Reconfiguring TPC-C with low skew

Throughput (MTX/second)

- Squall
- DrTM+B
- DrTM+B/no copy

Detect imbalance

+ no data migration

vs. before 6% ↓

vs. before 81% ↓
Performance of load balance

Reconfiguring TPC-C with low skew

Latency(ms)

- Detect imbalance
  - vs. before 0.05% ↑
  - vs. before 4.3X% ↑

Done

+ no data migration

Squall
DrTM+B
DrTM+B/no copy
Breakdown of data migration

- Reconfiguring TPC-C with high skew

**Throughput (MTX/second)**

- DrTM+B/copy 1 primary
- DrTM+B/copy 1 backup
- DrTM+B/copy parallel

**Time (second)**

- Detect imbalance
- Time 1(s)
- vs. before 4%
- vs. before 24%
Affected TXs & Network transferred

- Micro-benchmark based on TPC-C
- Swapping partitions between 2 nodes

**Affected TXs**

<table>
<thead>
<tr>
<th>Swapped partitions</th>
<th>Same</th>
<th>47X</th>
<th>86X</th>
<th>120X</th>
<th>235X</th>
<th>367X</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Network transferred (KB)**

- Data migration dominates
- 1.4X

Legend:
- DrTM+B/no copy
- DrTM+B
- Squall
Conclusion

■ Real workloads are **dynamic & skewed**
  ■ Requires fast & seamless live reconfiguration

■ DrTM+b provides fast live reconfiguration
  ■ **Optimized with features** in transactional systems
  ■ Nearly **no-effect** to TXs

Thanks & Questions?