Query Fresh: Log Shipping on Steroids

Tianzheng Wang*

Ryan Johnson

Ippokratis Pandis

*Currently at Simon Fraser University
High availability through log shipping

Primary: Read + Write

Backup(s): Read + Failover

“Real” database

Network

Log

Replay

Log

Widely used in practice
Desirable properties

- Easy impl. & maintenance
- Fresh
- Safe
- Fast primary
- High resource utilization
Strong safety and freshness

Synchronous log shipping

Fast log replay

<table>
<thead>
<tr>
<th>Primary</th>
<th>Commit?</th>
<th>Persist + ship + wait ack</th>
<th>Committed</th>
</tr>
</thead>
</table>

| Backup(s) | Persist log | Replay |

Sync or async

I/O, network and/or replay on the critical path
Synchronous log shipping: infeasible

- ERMIA* TPC-C, 2-socket, 16 physical cores, **10Gbe**

Synchronous log shipping: infeasible

- ERMIA* TPC-C, 2-socket, 16 physical cores, 10Gbe

Network + I/O: major bottleneck

Reality: *asynchronous* log shipping $\rightarrow$ freshness gap

Safety and freshness traded for primary speed
Query Fresh

• Synchronous log shipping: leverage modern hardware
• Fast replay: append-only storage + indirection
Modern HW: synchronous log shipping possible

Non-volatile RAM (NVRAM)

NV-DIMM

3D XPoint

Memristor
Trend: network tracks memory speed

Network no longer the biggest bottleneck

* https://www.infinibandta.org/infiniband-roadmap/
Modern HW: synchronous log shipping possible

NVRAM → Fast persistence

High BW network → Fast transfer

RDMA over NVRAM: fast *synchronous* log shipping

See paper for challenges & soln.
Desirable properties

- Easy impl. & maintenance
- Fresh
- Fast primary
- Safe
- High resource utilization
Sync. Shipping != Fresh Reads

- Two durable copies
- Create actual tuples
- Memory allocation
- Many index operations (esp. secondary indexes)

Heavyweight record creation + serial replay = stale
Append-only storage: freshness possible

- Only keep one durable copy of data – *the log*
- Redo-only logging, log record == data tuple
- LSN == position in the log, directly comparable

Query Fresh: *Log == Database with RDMA + NVRAM*

- Parallel (see paper)

The log

RDMA over NVRAM

Replay

- Sync. commit: safe
- Log tail in NVRAM
- Indexes: key $\rightarrow$ RID
- Queries check both arrays
- Extract tuple location
- Little memory allocation
- No index operation (except for inserts)

<table>
<thead>
<tr>
<th>RID</th>
<th>Where?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LSN 10</td>
</tr>
<tr>
<td>1</td>
<td>LSN 20</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

(New) Per-table replay array

Fast sync log shipping + append-only = safe & fresh
Query Fresh vs. Existing

Query Fresh balances all aspects
Evaluation

• 8 x 16-core (2-socket) nodes
  • 1 primary + up to 7 backups
  • Xeon E5-2650 v2, 64GB RAM, logs in tmpfs
  • Target NV-DIMM: DRAM as log buffer + CLWB/FLUSH emulation

• Network
  • Query Fresh: 56Gbps Infiniband FDR 4x + RDMA
  • Other schemes: 10Gbps Ethernet + TCP

• Benchmarks in ERMIA
  • Primary: Full TPC-C, low contention
  • Backups: StockLevel + OrderStaus
Query Fresh: maintains fast primary

- 16 workers on primary, 4 replay threads + 12 workers on backups
- Utilization = 75% (12 workers out of 16 total)
Query Fresh: fresh and high utilization

- Freshness: backup read view / primary read view * 100%
Conclusions

• Slow network + Fast OLTP = Stale and Unsafe
  • Redundant data copies (dual-copy architecture)
  • Often serial, heavy-weighted log replay

• **Query Fresh** = Fast network + NVRAM
  + Append-only storage with indirection

Find out more in our paper and code repo!
https://github.com/ermia-db

Thank you!