# Query Fresh: Log Shipping on Steroids

Tianzheng Wang*

Ryan Johnson

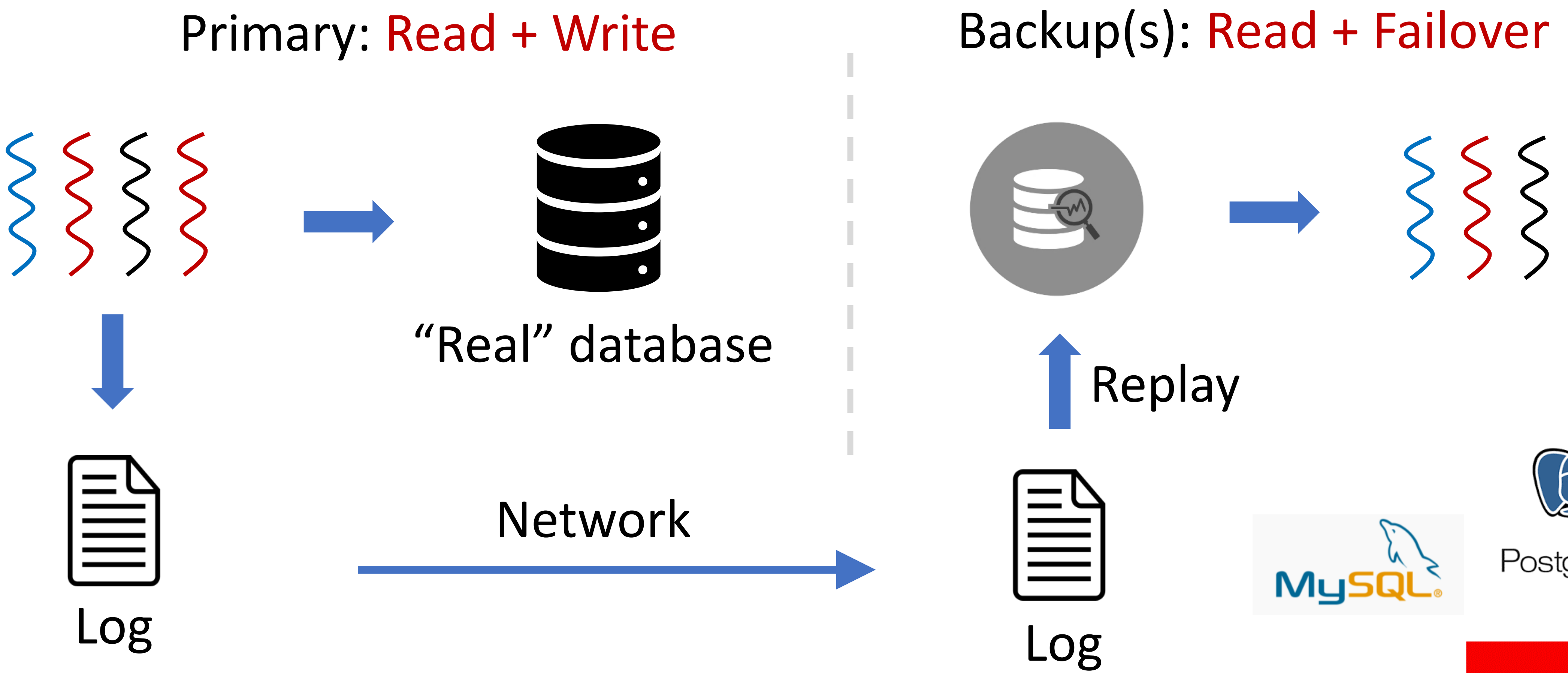
Ippokratis Pandis

UNIVERSITY OF TORONTO

amazon web services™

*Currently at Simon Fraser University

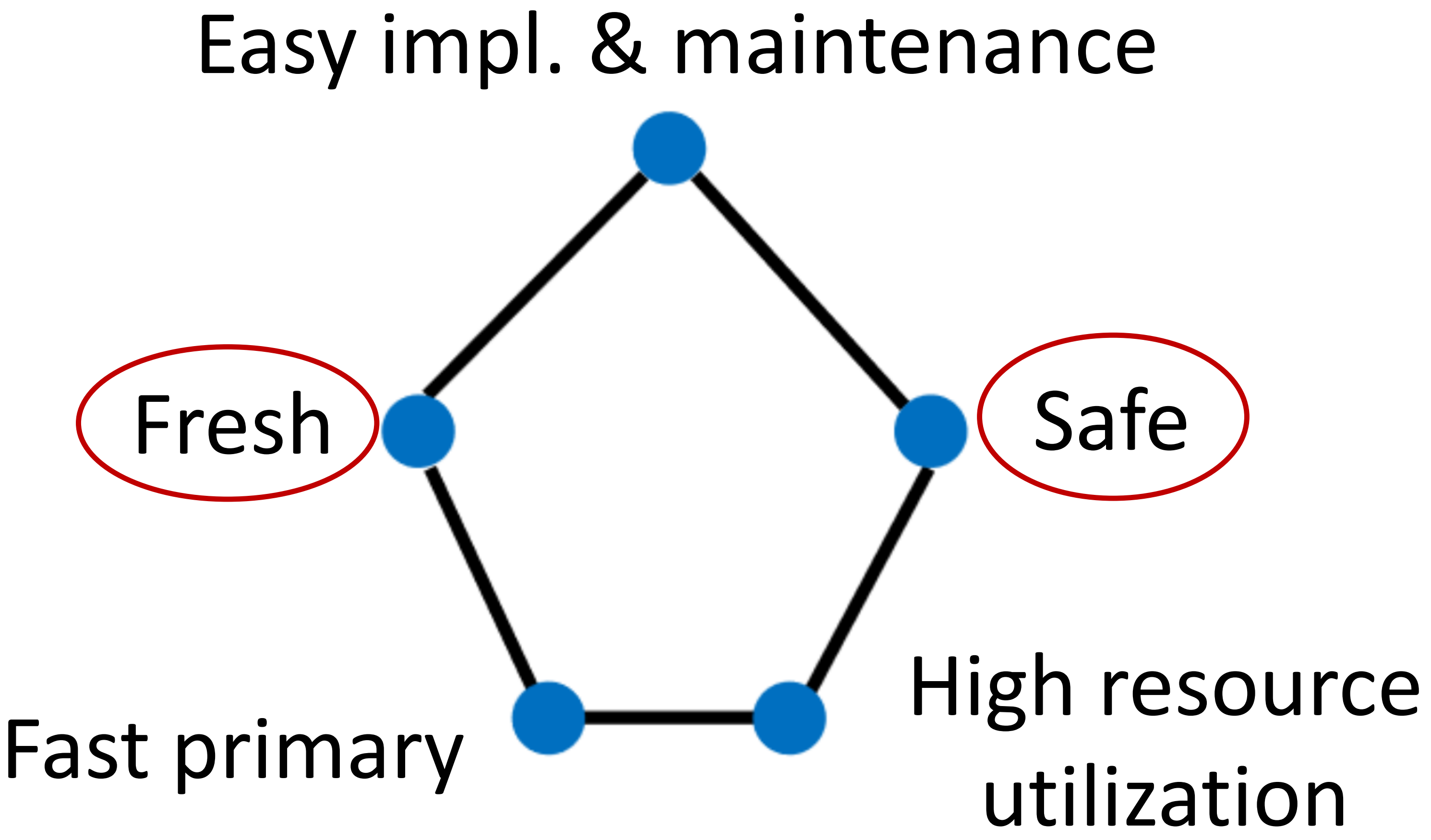
# High availability through log shipping



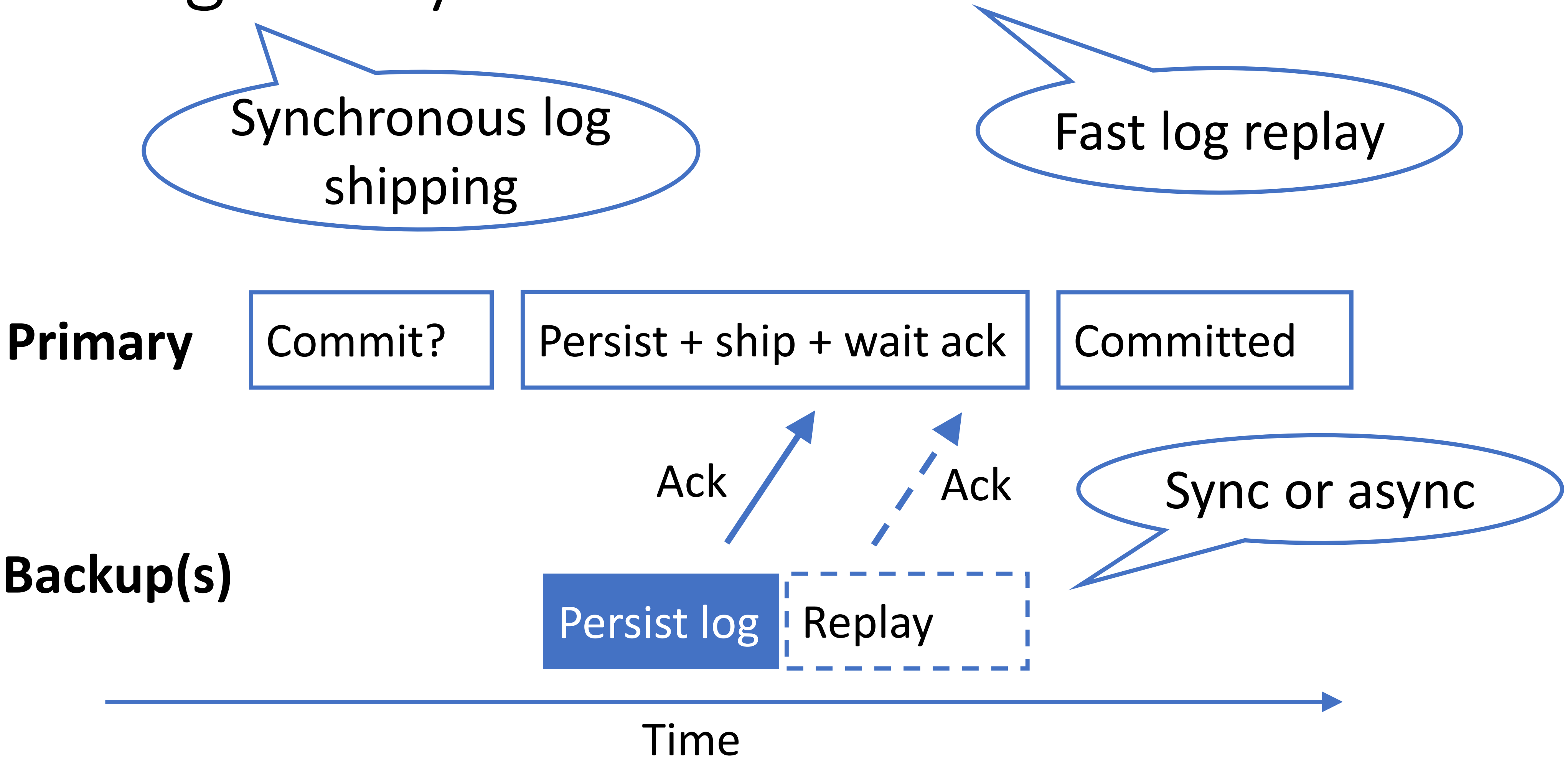

Widely used in practice

Microsoft SQL Server · MySQL · PostgreSQL · SAP · ORACLE

# Desirable properties



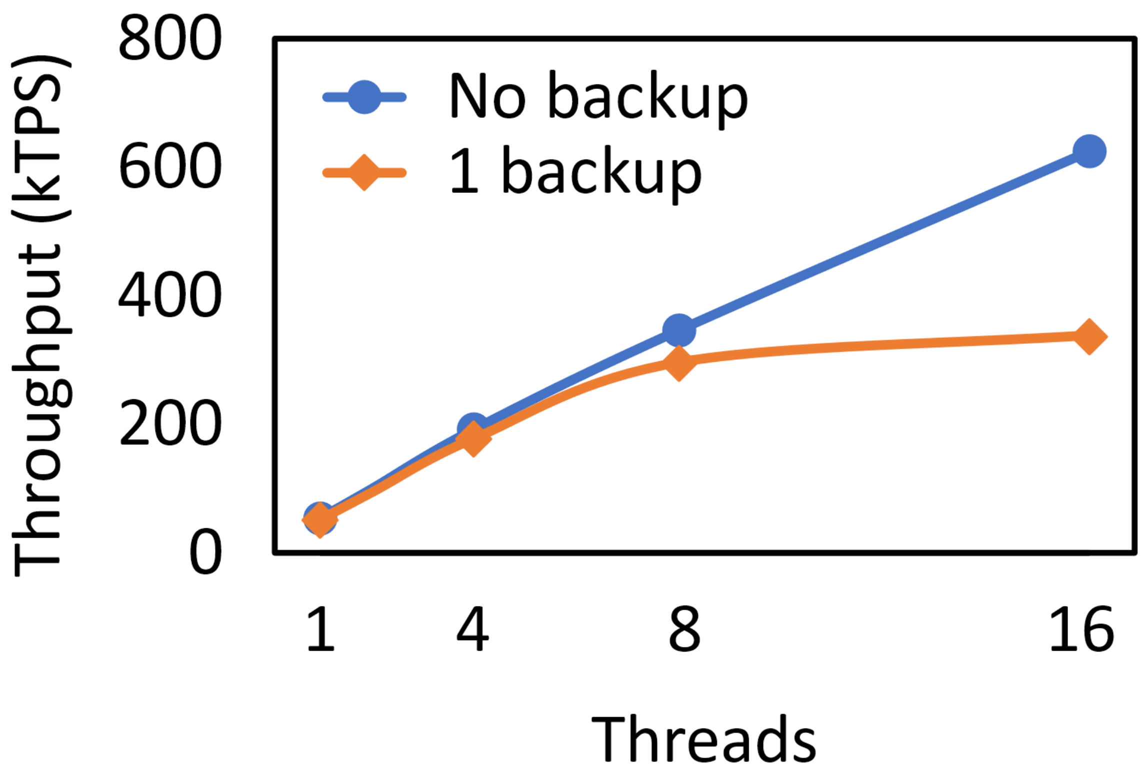
# Strong safety and freshness




I/O, network and/or replay on the critical path

# Synchronous log shipping: infeasible

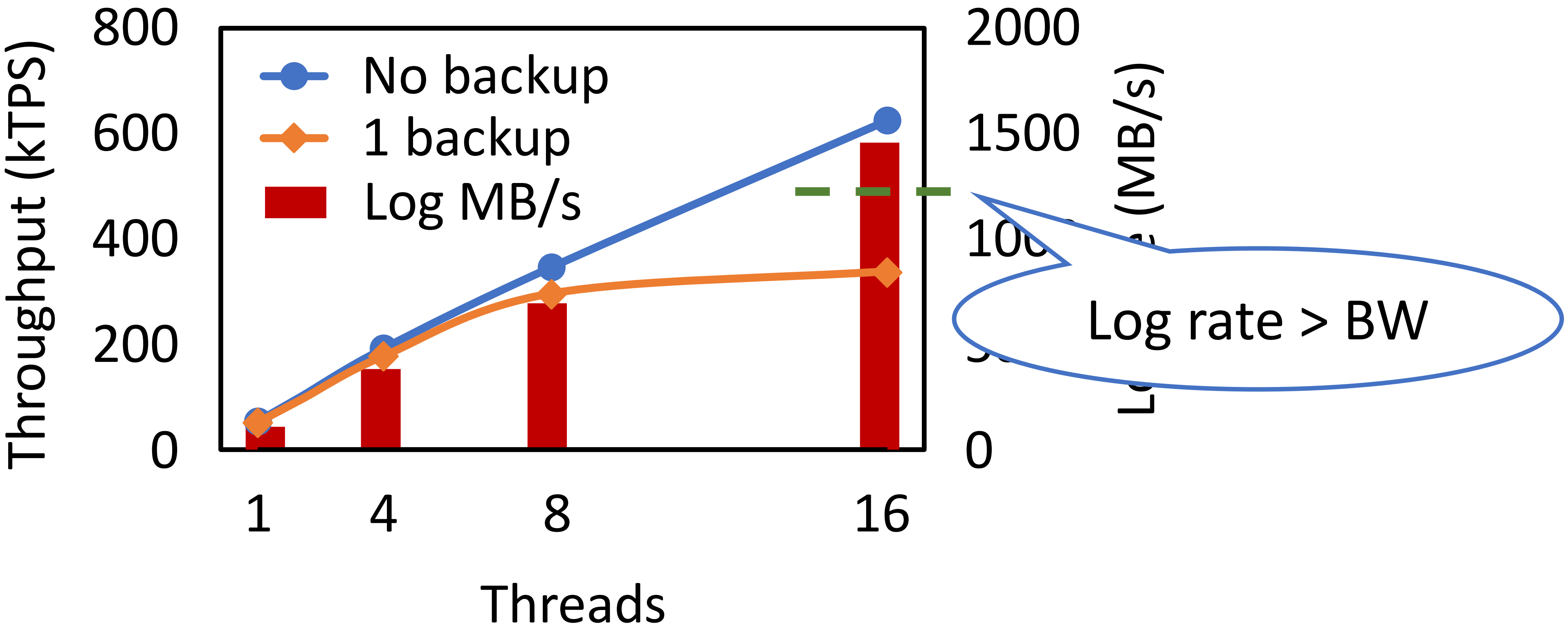
- ERMIA* TPC-C, 2-socket, 16 physical cores, **10Gbe**




* K. Kim, T. Wang, R. Johnson, I. Pandis, *ERMIA: Fast Memory-Optimized Database System for Heterogeneous Workloads*, SIGMOD 2016

# Synchronous log shipping: infeasible

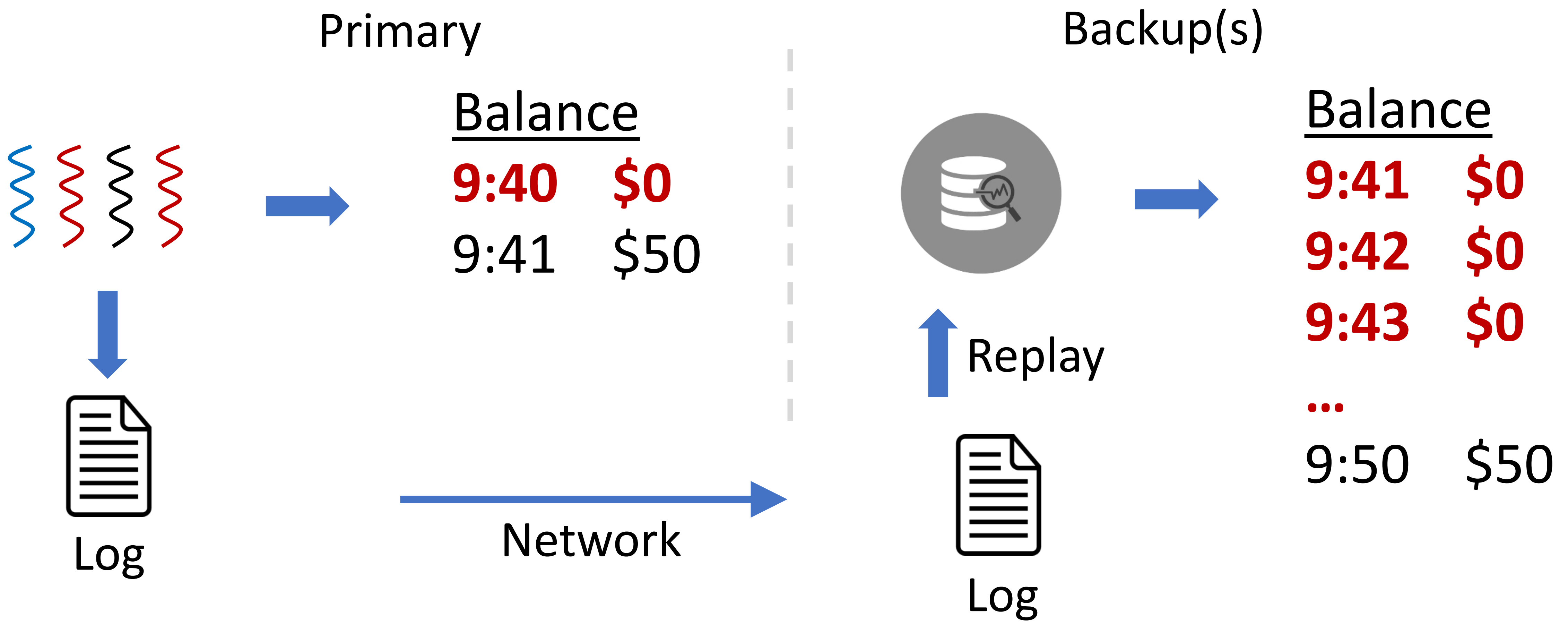
- ERMIA* TPC-C, 2-socket, 16 physical cores, **10Gbe**




Network + I/O: major bottleneck

* K. Kim, T. Wang, R. Johnson, I. Pandis, *ERMIA: Fast Memory-Optimized Database System for Heterogeneous Workloads*, SIGMOD 2016

# Reality: *asynchronous* log shipping → freshness gap



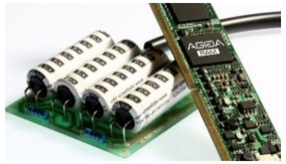

Safety and freshness traded for primary speed

# Query Fresh

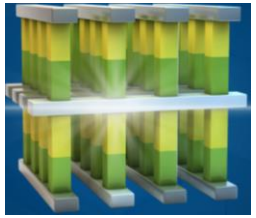
- Synchronous log shipping: leverage modern hardware
- Fast replay: append-only storage + indirection

# Modern HW: synchronous log shipping possible

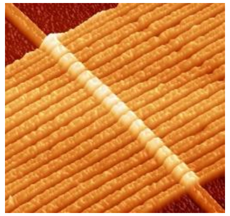
Non-volatile RAM
(NVRAM)



**NV-DIMM**

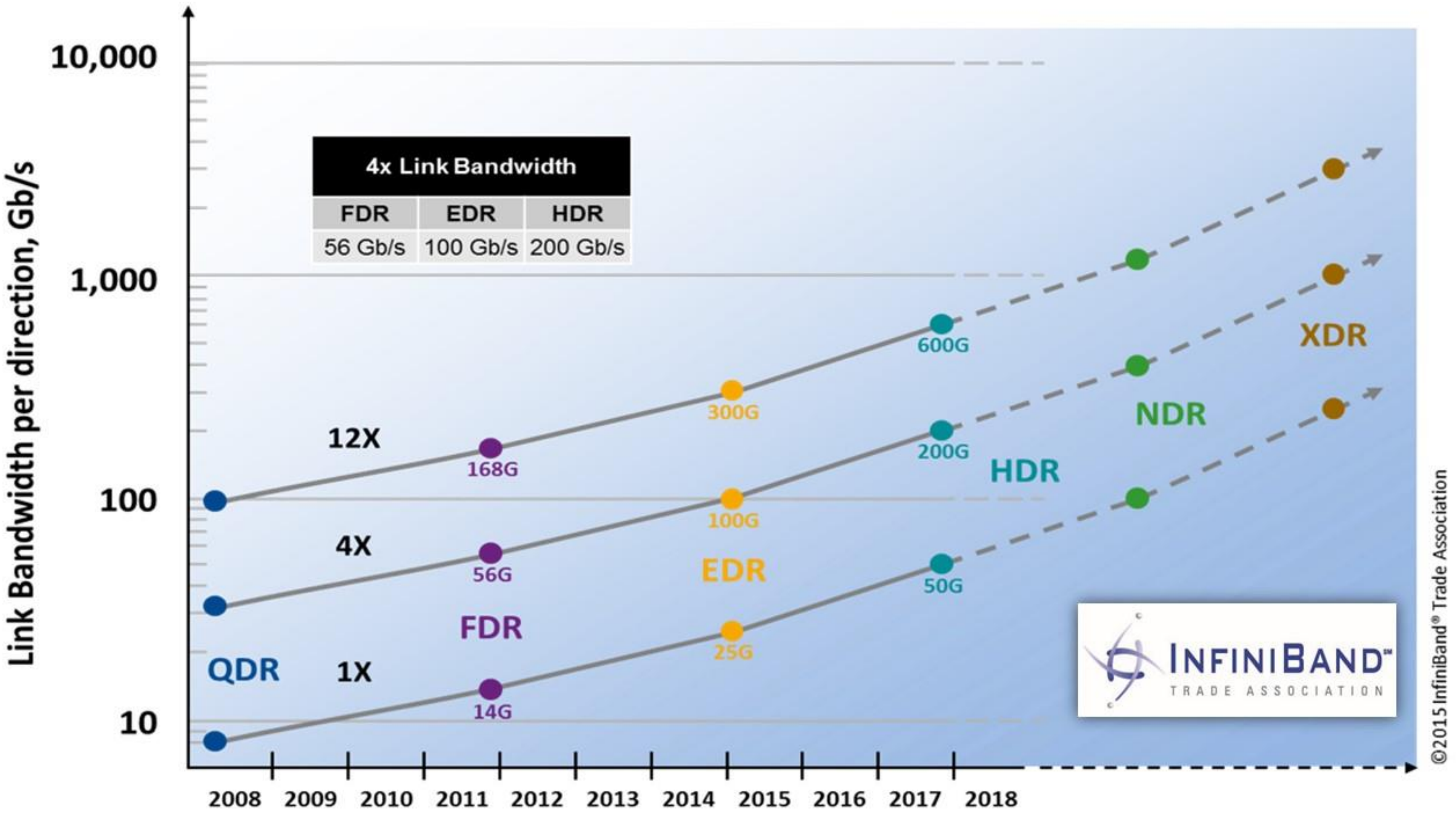


**3D XPoint**



**Memristor**

# Trend: network tracks memory speed

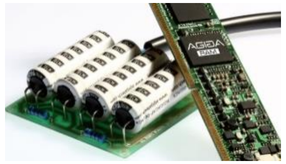



**Network no longer the biggest bottleneck**

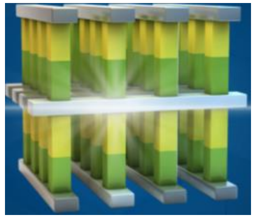
* https://www.infinibandta.org/infiniband-roadmap/

# Modern HW: synchronous log shipping possible

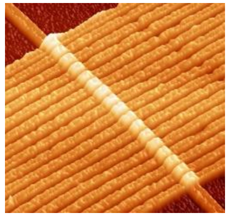
NVRAM
→ Fast persistence



NV-DIMM



3D XPoint



Memristor

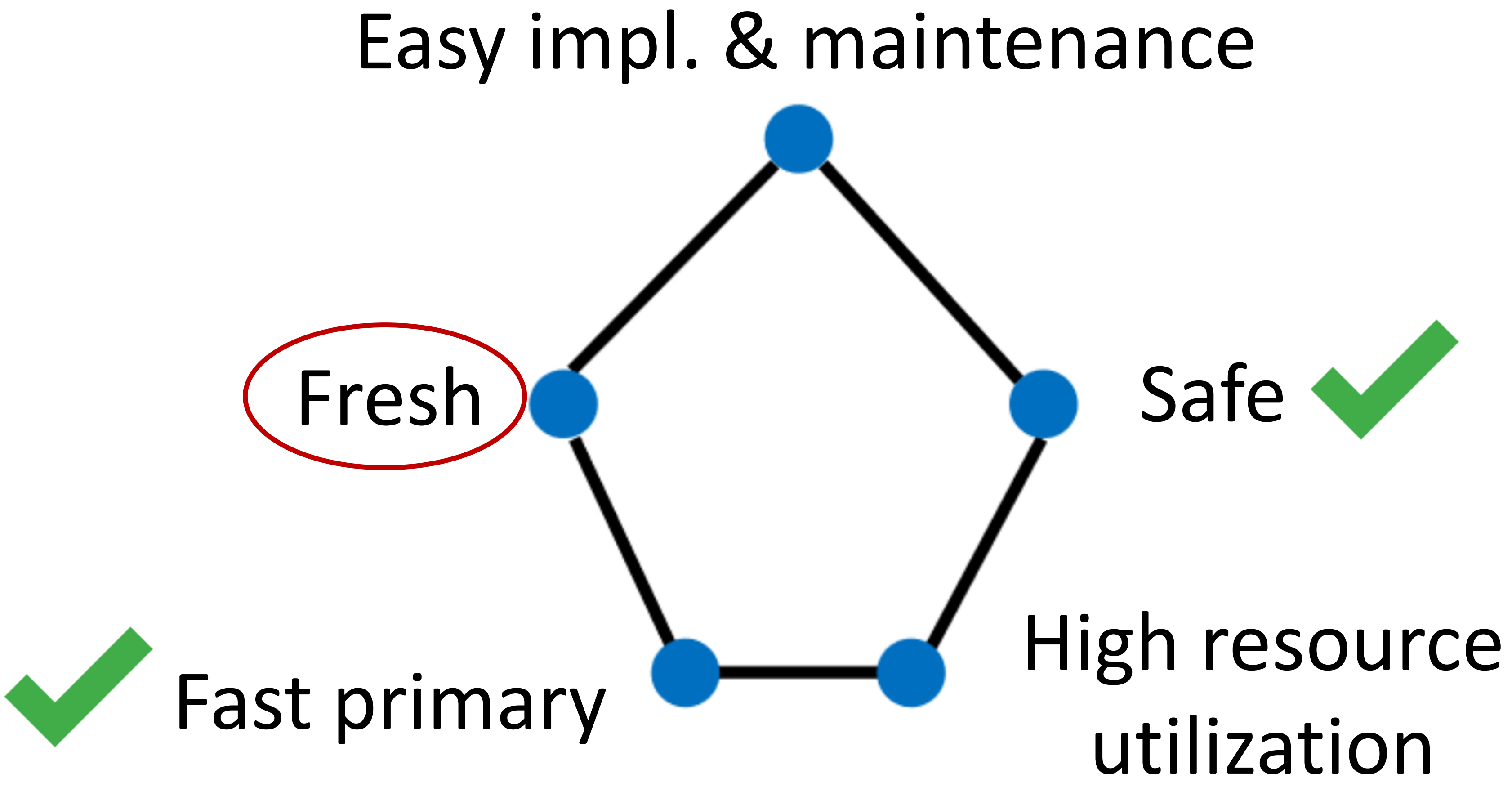
High BW network
→ Fast transfer

InfiniBand, Converged Ethernet
(56Gbps+)

See paper for challenges & soln.

RDMA over NVRAM: fast ***synchronous*** log shipping

# Desirable properties



# Sync. Shipping != Fresh Reads

The log

Replay

The “real” database

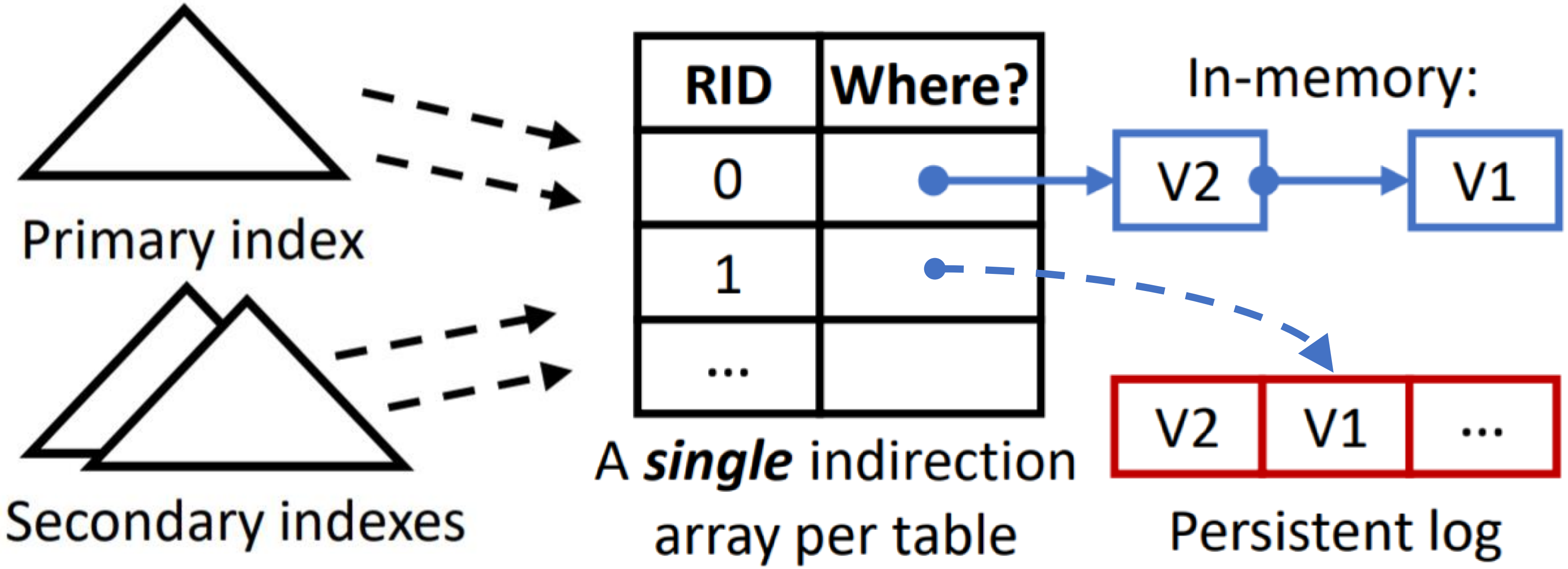
Often serial

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

Heavyweight record creation + serial replay = stale

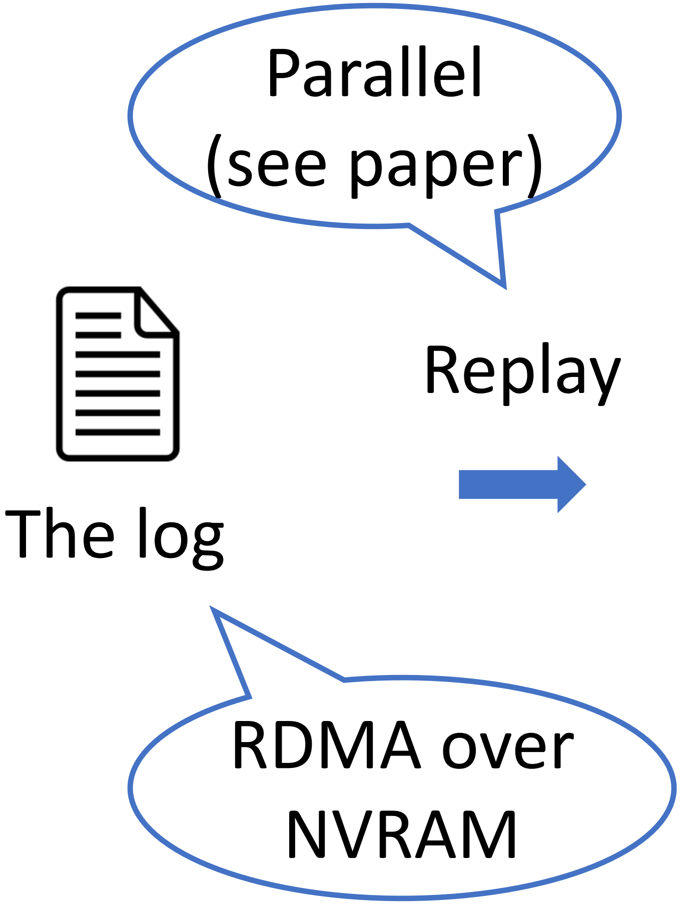
# Append-only storage: freshness possible

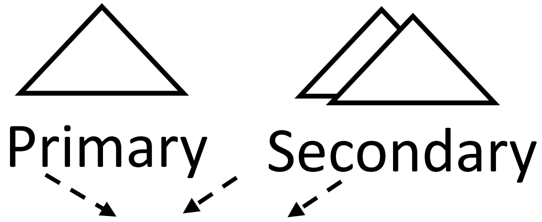
- Only keep one durable copy of data – ***the log***
- Redo-only logging, log record == data tuple
- LSN == position in the log, directly comparable




* K. Kim, T. Wang, R. Johnson, I. Pandis, *ERMIA: Fast Memory-Optimized Database System for Heterogeneous Workloads*, SIGMOD 2016

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



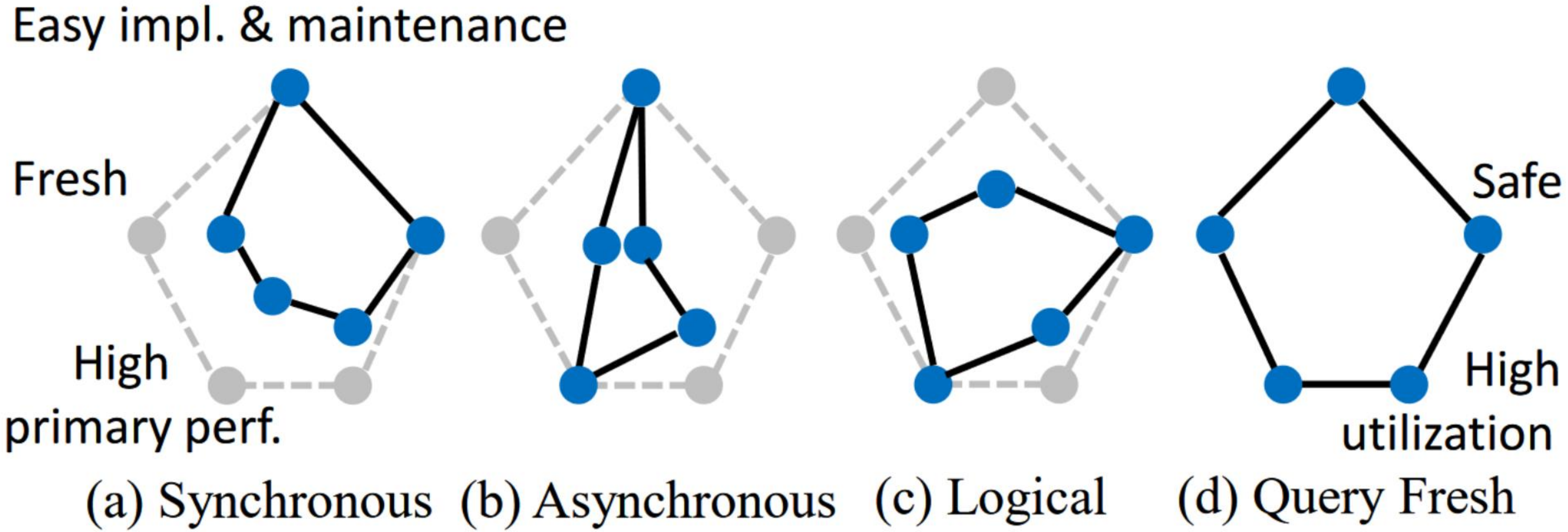

| RID | Where? |
|---|---|
| 0 | LSN 10 |
| 1 | LSN 20 |
| … | … |

(**New**) Per-table replay array

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

Fast sync log shipping + append-only = safe & fresh

# Query Fresh vs. Existing



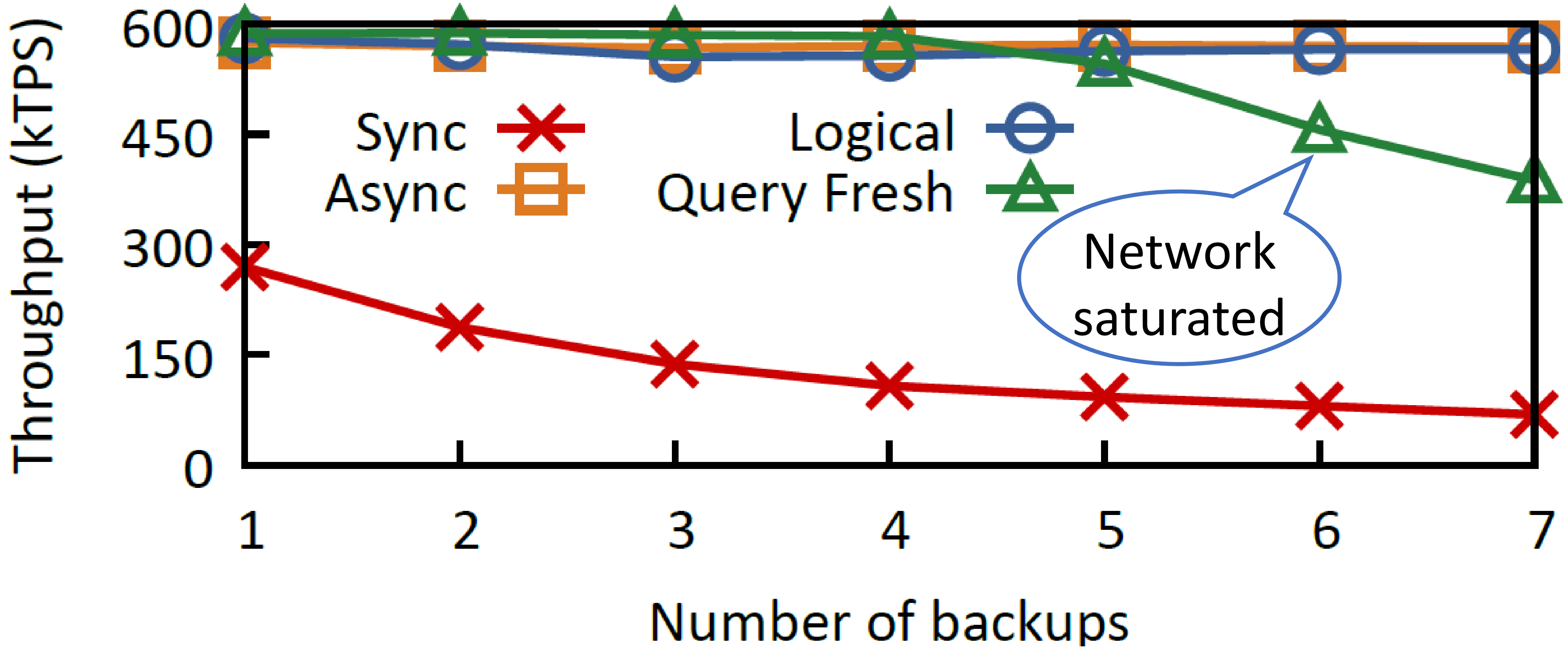

Query Fresh balances all aspects

# 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

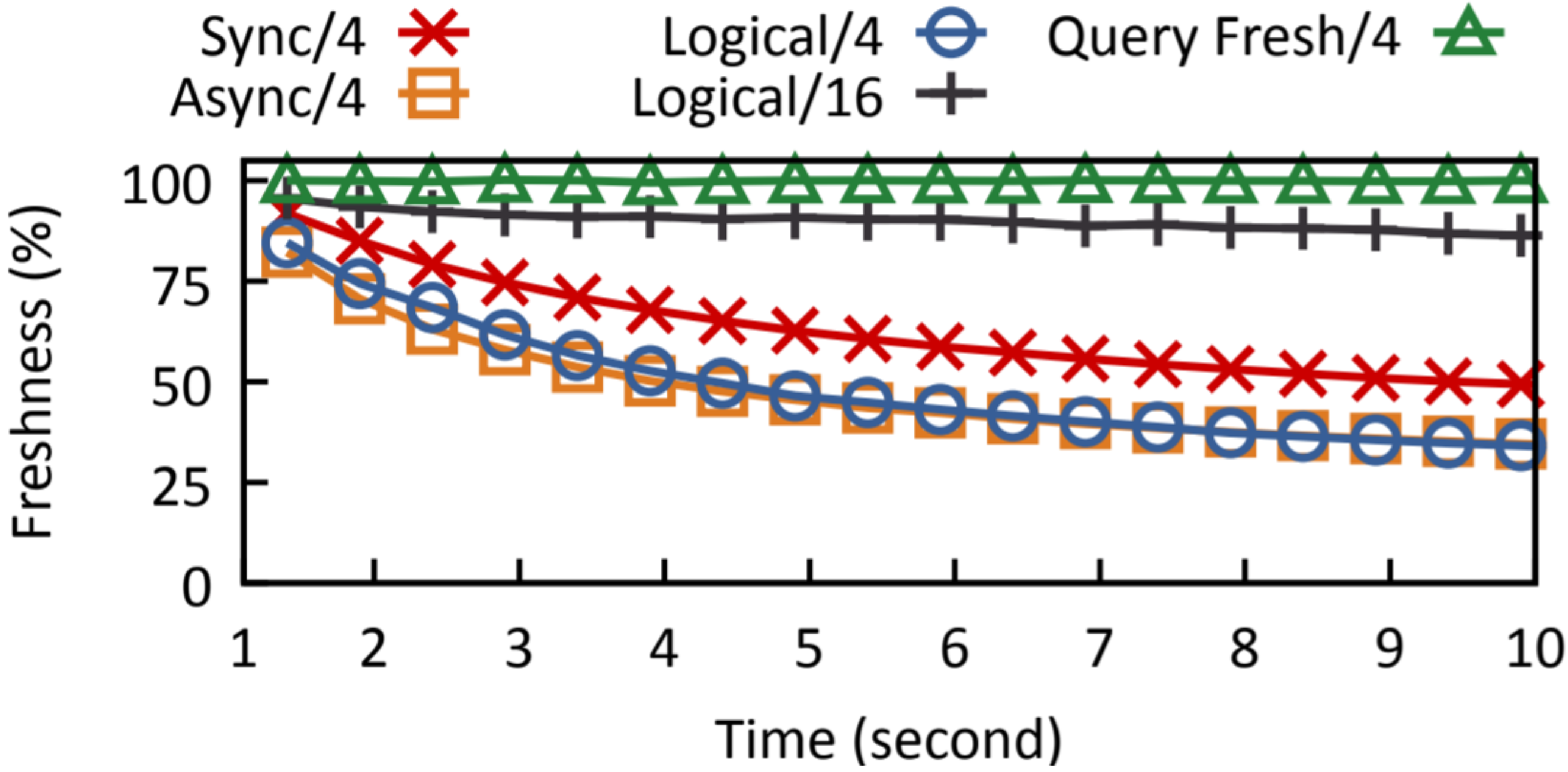
# Query Fresh: maintains fast primary

- 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

Fast, sync, safe

- **Query Fresh** = Fast network + NVRAM

  + Append-only storage with indirection

Fast replay → Fresh reads

***Find out more in our paper and code repo!***

[https://github.com/ermia-db](https://github.com/ermia-db)

*Thank you!*