Beyond Manual Caching

Offloading MySQL and Postgres Workloads Automatically

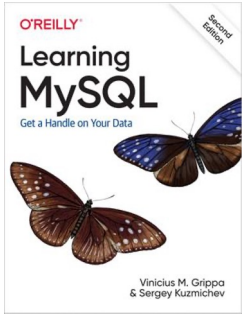
Vinicius Grippa

Lead Database Engineer @ Readyset

About Me

SELECT * FROM person WHERE name LIKE ‘Vini’;

- Lead Database Engineer at **Readyset**
- Ex-Percona working with MySQL and MongoDB
- Co-author of the book Learning MySQL
- Organizer of the MySQL community in Brazil (MUG)



COMUNIDADE MySQL BR

Agenda

Agenda

01

THE LANDSCAPE

Defining In-Memory Databases and the raw speed vs. budget trade-off.

02

MARKET REALITY

Challenges of scale: DDR5 price volatility and physical capacity caps.

03

THE READYSET ENGINE

Architectural overview of automated SQL result caching and auto-maintenance.

04

CACHING MODES

Shallow (TTL) vs. Deep (Incremental View) caching strategies.

05

Performance

Performance analysis: 60x throughput gains and sub-millisecond latency.

06

Q&A

Open discussion, technical resources, and project roadmap.

WHAT IS AN IN-MEMORY DATABASE?

Core Concept: Data lives natively in system MEMORY rather than experiencing execution bottlenecks on a traditional Disk.

Sub-ms Speed

Completely eliminates standard Disk I/O storage bottlenecks to unlock true "instantaneous" performance vectors.

High Concurrency

Effortlessly scales to process millions of concurrent operational request structures simultaneously.

Simplified Logic

Bypasses complex kernel storage layout translation steps via direct CPU-to-Memory channel communication pathways.

POPULAR INDUSTRY EXAMPLES:

Readyset | Redis & Memcached | SAP HANA

CHALLENGES & MARKET REALITY

Economic Hurdle

High Cost Per GB makes In-Memory storage significantly more expensive than enterprise-grade SSD or HDD solutions.

Price Volatility

Market shifts have led to recent DDR5 price spikes of up to[3], making infrastructure scaling unpredictable.

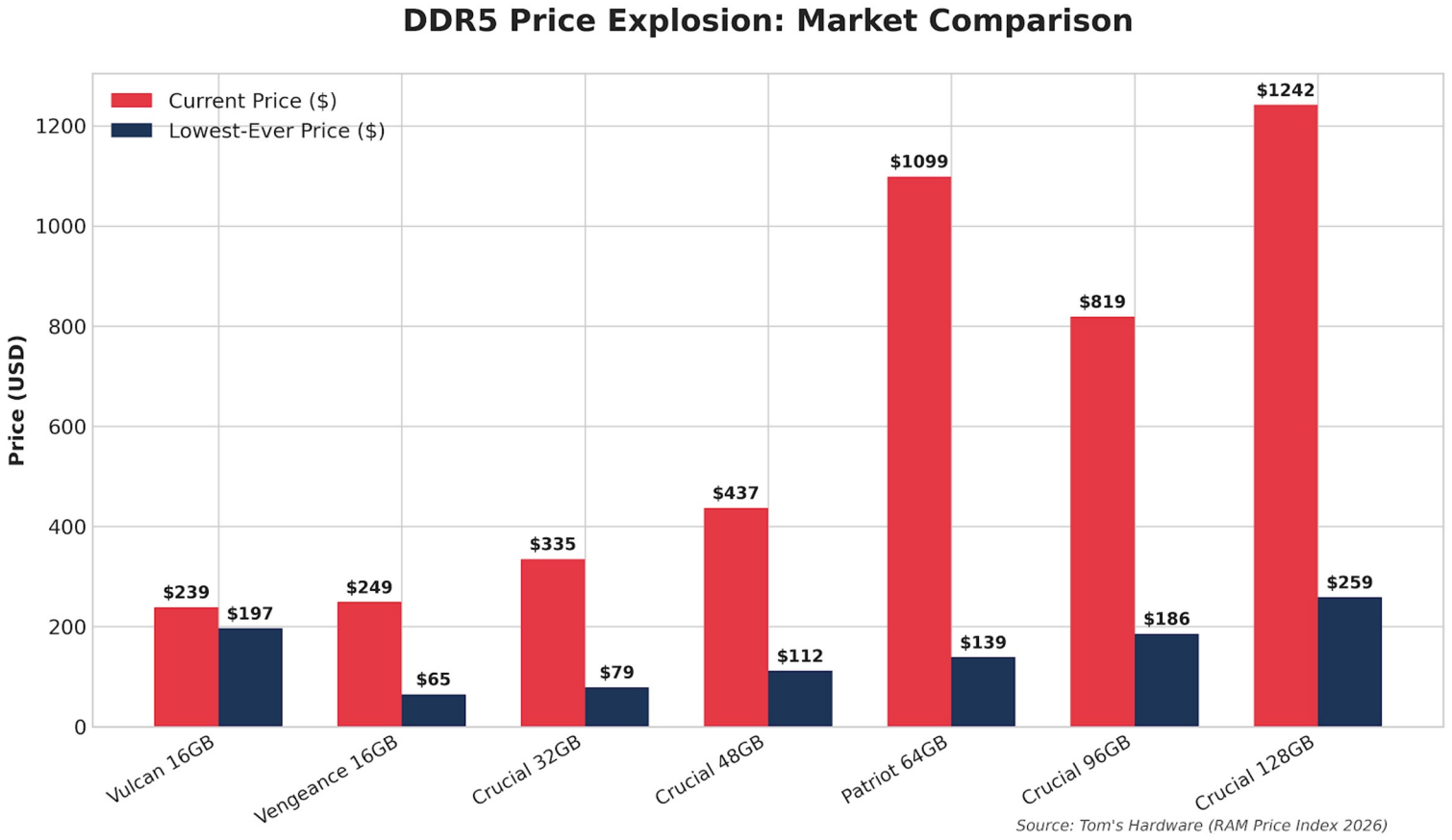
Physical Limits

Unlike cloud storage, total data volume is strictly capped by the physical RAM slots available on the server motherboard.

KEY TAKEAWAY

Trading Budget for Raw Operational Speed

Challenges & Market Reality



https://www.tomshardware.com/pc-components/ram/ram-price-index-2026-lowest-price-on-ddr5-and-ddr4-memory-of-all-capacities

“The hard problem is not *storing* results in memory; it is knowing when they are no *longer* valid.”

Readyset

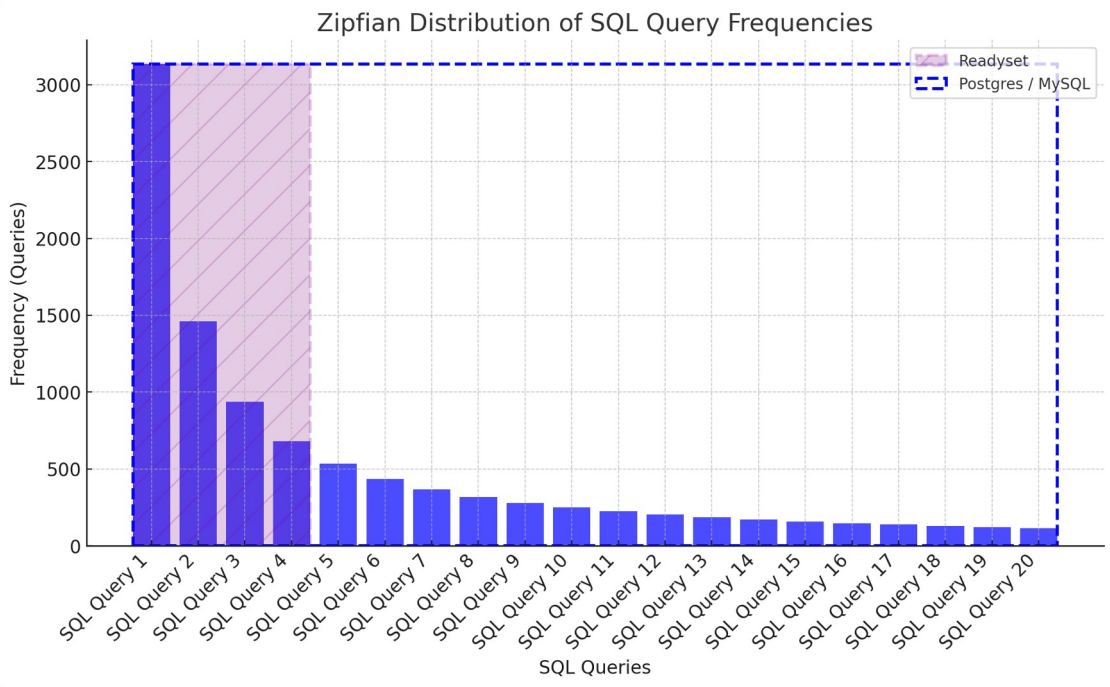
SQL caching engine

Readyset

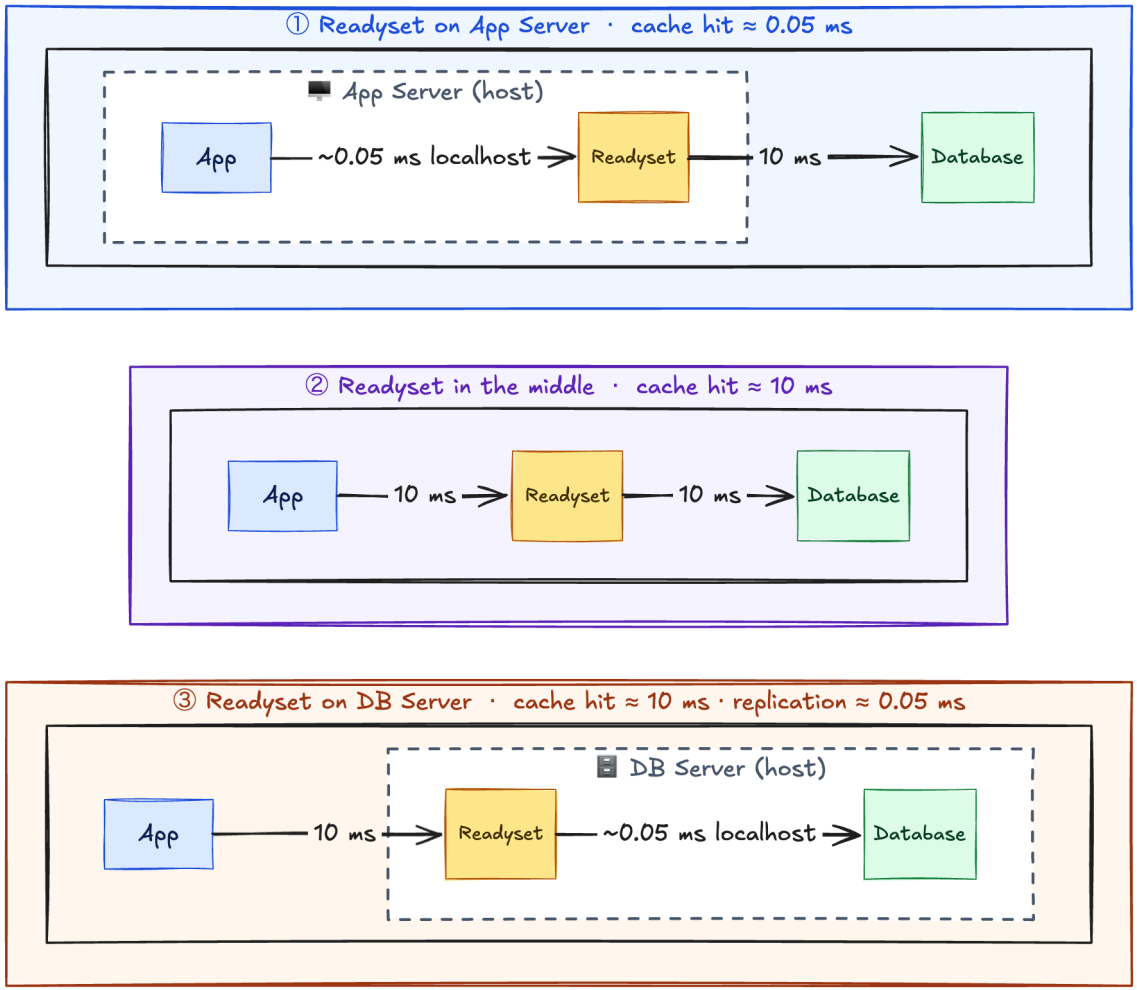
- **Based on MIT PHD Thesis:** Partial State in Dataflow-Based Materialized Views
- **Drop-in SQL Caching:** Sits between app and DB; no architectural changes required.
- **Sub-millisecond Performance:** Serves repeated queries from RAM to offload primary DB.
- **Auto-Maintenance:** Tracks dependencies and refreshes data automatically as it changes.
- **Simplified Scaling:** Eliminates manual cache invalidation logic across services.

Readyset

- Queries often follow a Zipfian distribution
- Read heavy workloads. Infrequent changes. Eventual consistency is okay.



Architecture Examples

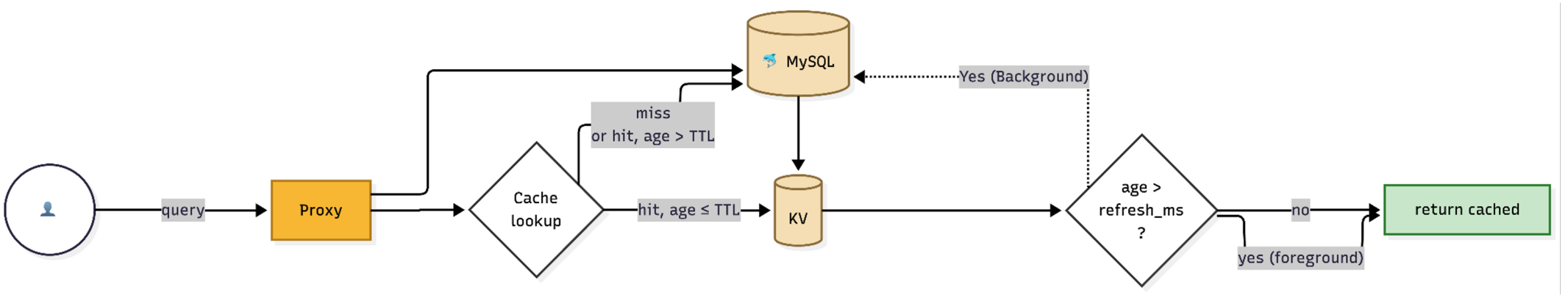


Shallow Caching

Shallow Caching

- **TTL-based query result cache:** Shallow caching stores query results in memory for a configured time window, then refreshes when entries become stale.
- **No dataflow materialization required:** It does not build a full incremental view graph, which makes setup lighter and faster.
- **Broader query compatibility:** It can cache many queries that deep/dataflow caching may not support yet.
- **Freshness is policy-driven:** Data is refreshed by TTL and optional refresh settings (on-demand or scheduled), not by per-row CDC dependency tracking.
- **Good fit for pragmatic acceleration:** Useful when you need quick read latency improvements with simpler operational overhead, while accepting TTL-style freshness tradeoffs.

Shallow Caching

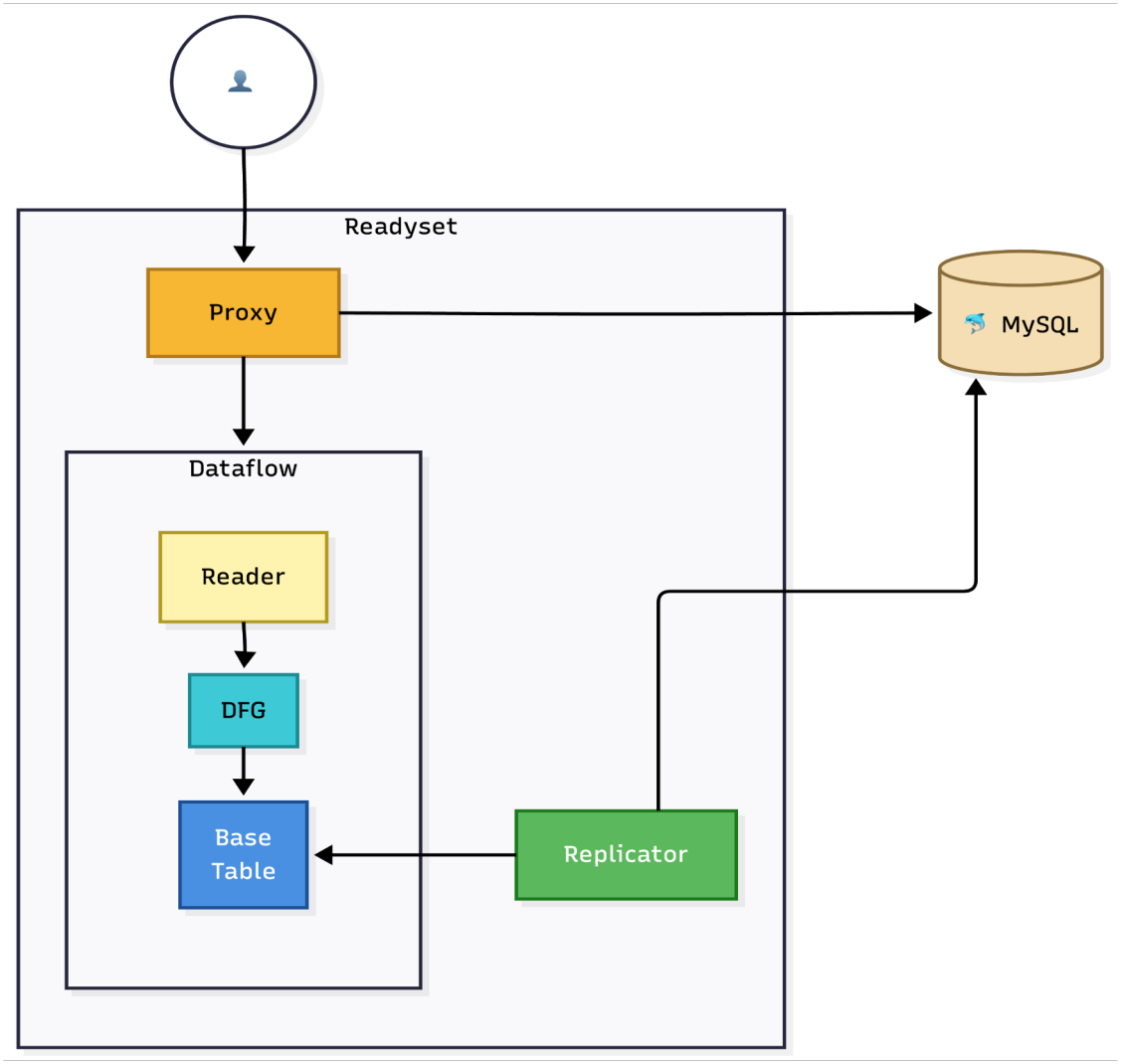



```
CREATE SHALLOW CACHE  POLICY TTL 10 SECONDS REFRESH 5 SECONDS
COALESCE 5 SECONDS FROM SELECT a,b FROM …;
```

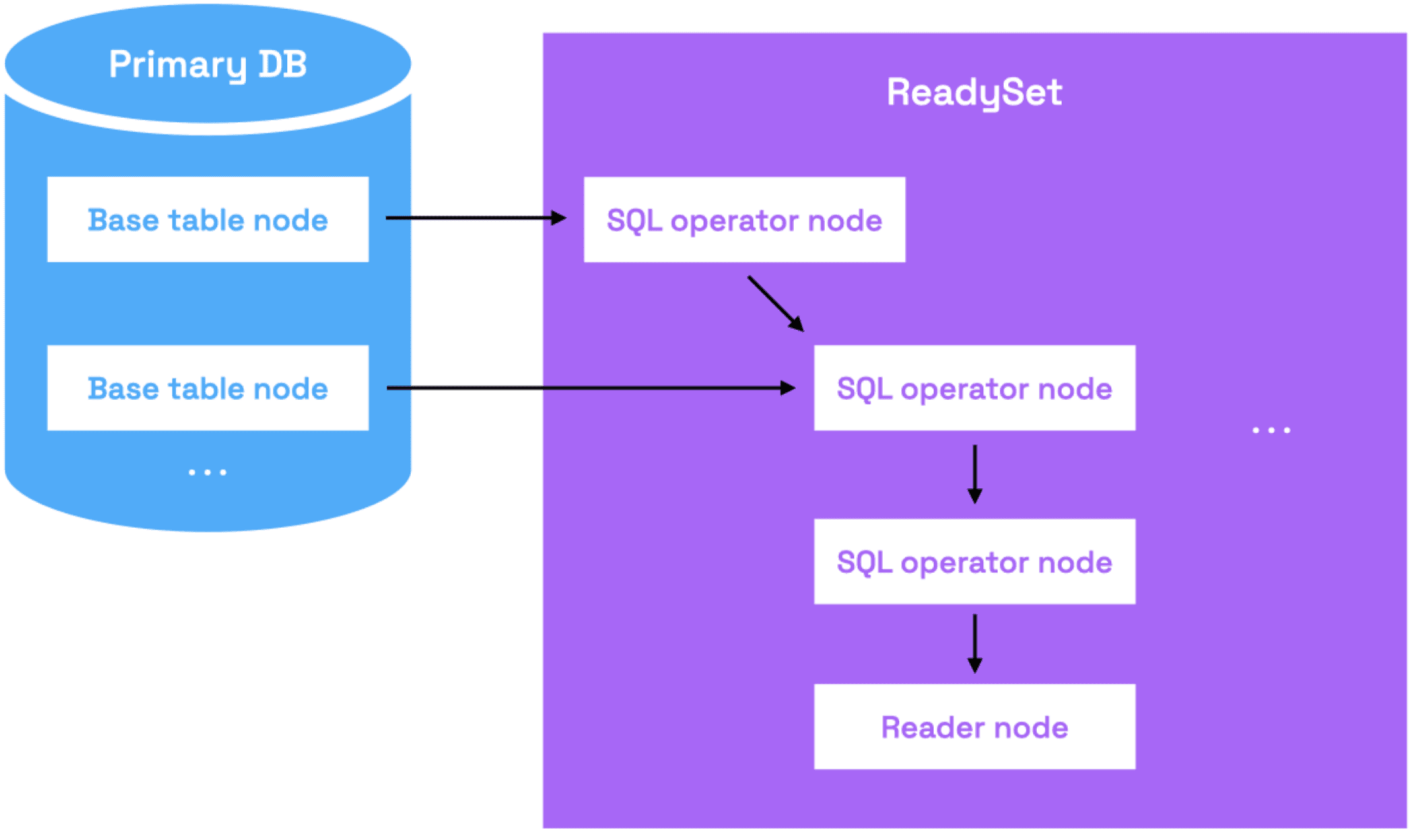
Deep Caching

Deep Caching

- Snapshot
- Replication / CDC
- Proxy
- Dataflow
- Partial Materialization
- Cache Miss
- Incremental View Maintenance (IVM)



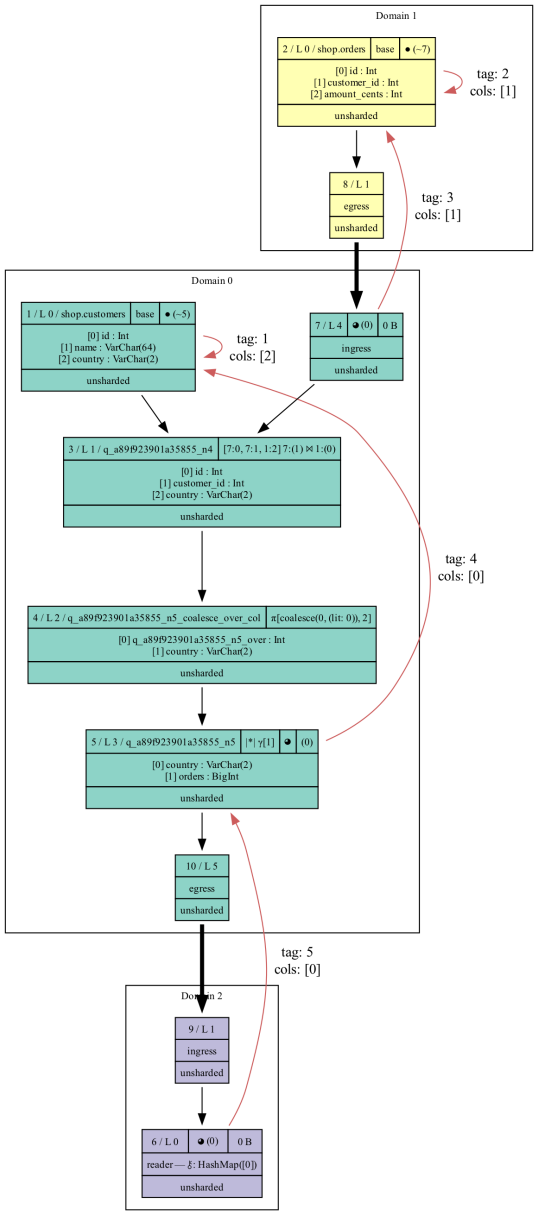
Deep Caching



Deep Caching

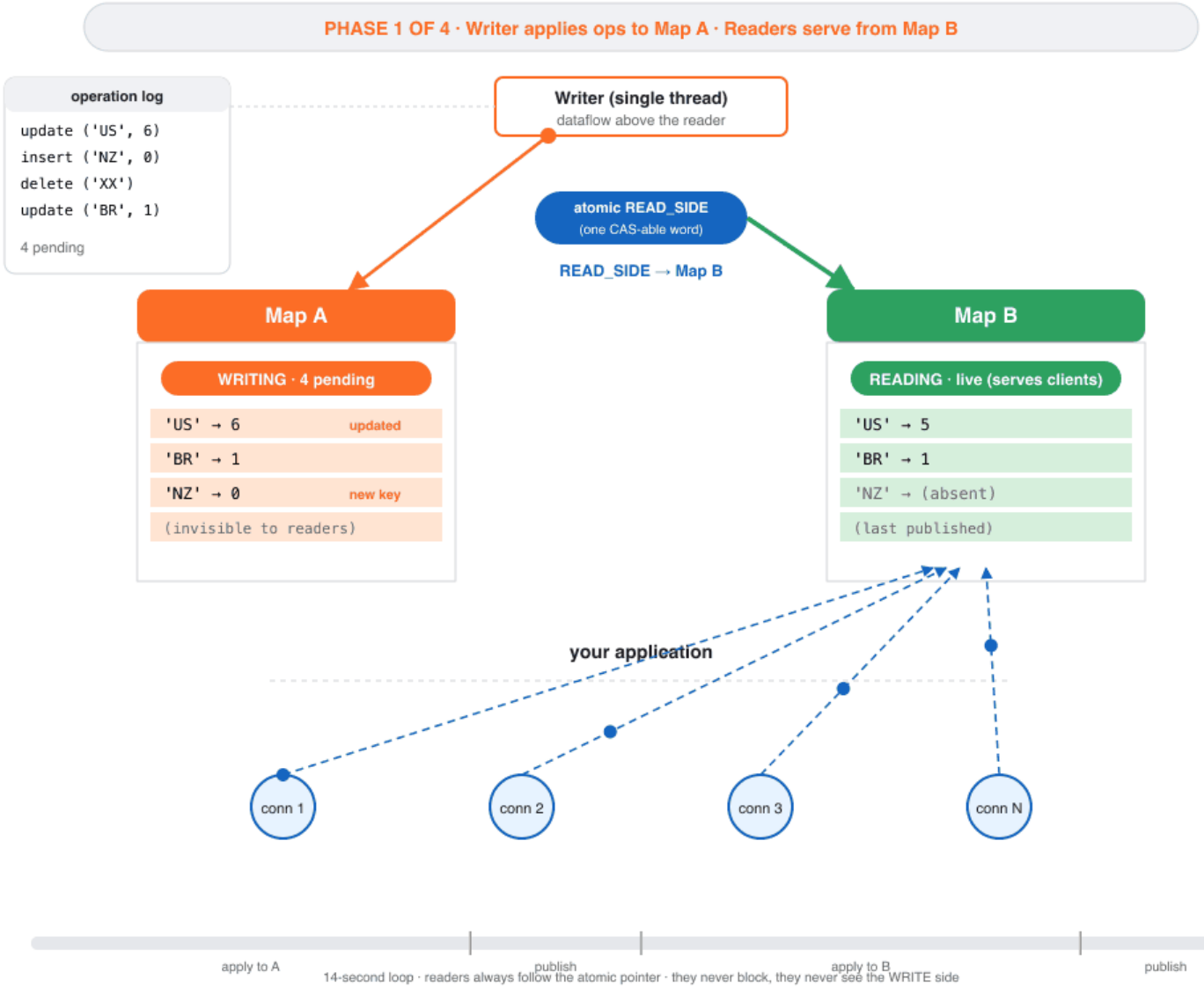
```
SELECT c.country, COUNT(*) AS orders
FROM orders o
INNER JOIN customers c ON o.customer_id = c.id
WHERE c.country = ?
GROUP BY c.country;
```

Deep Caching

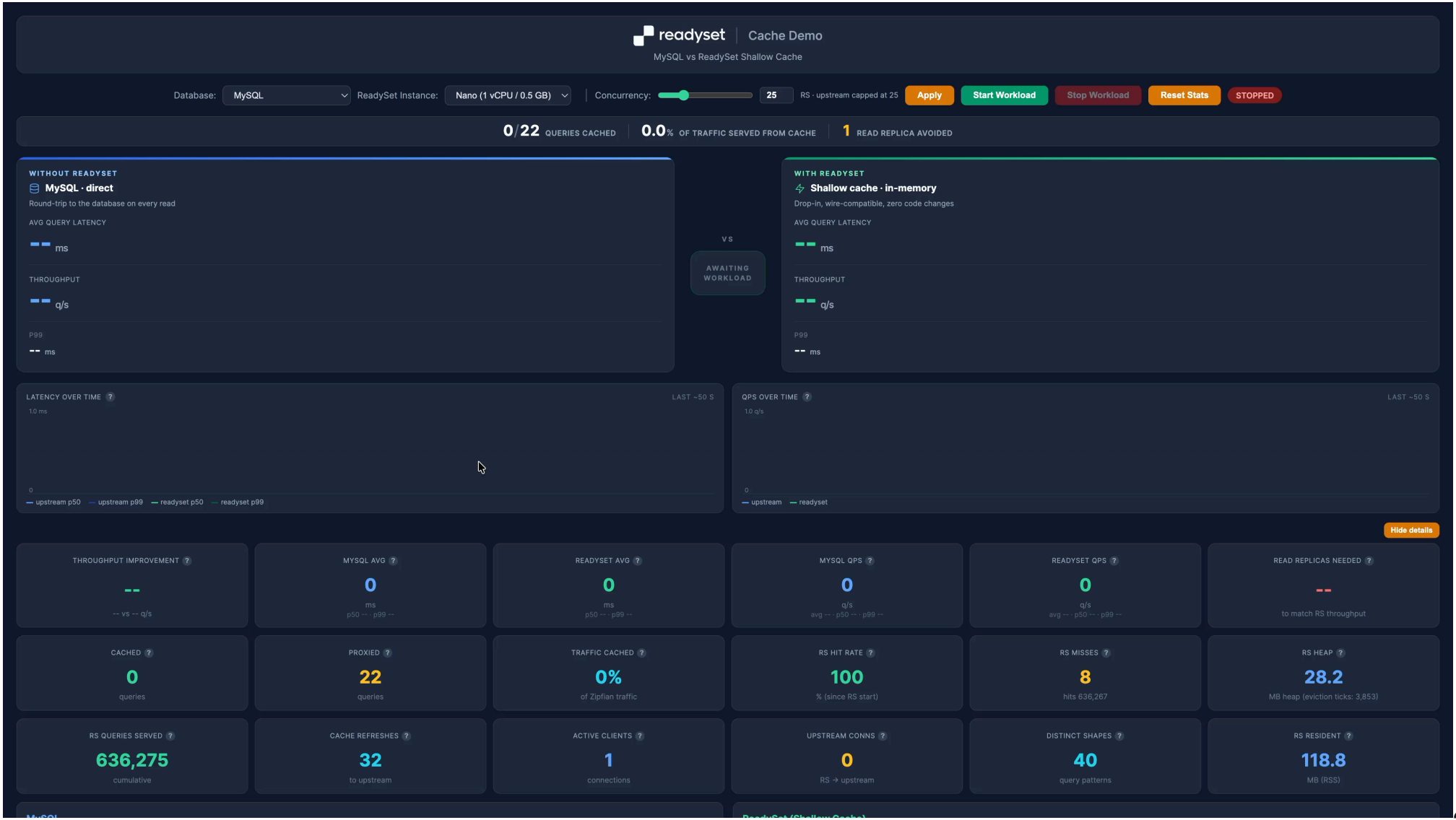


Readyset reader: the left-right pattern

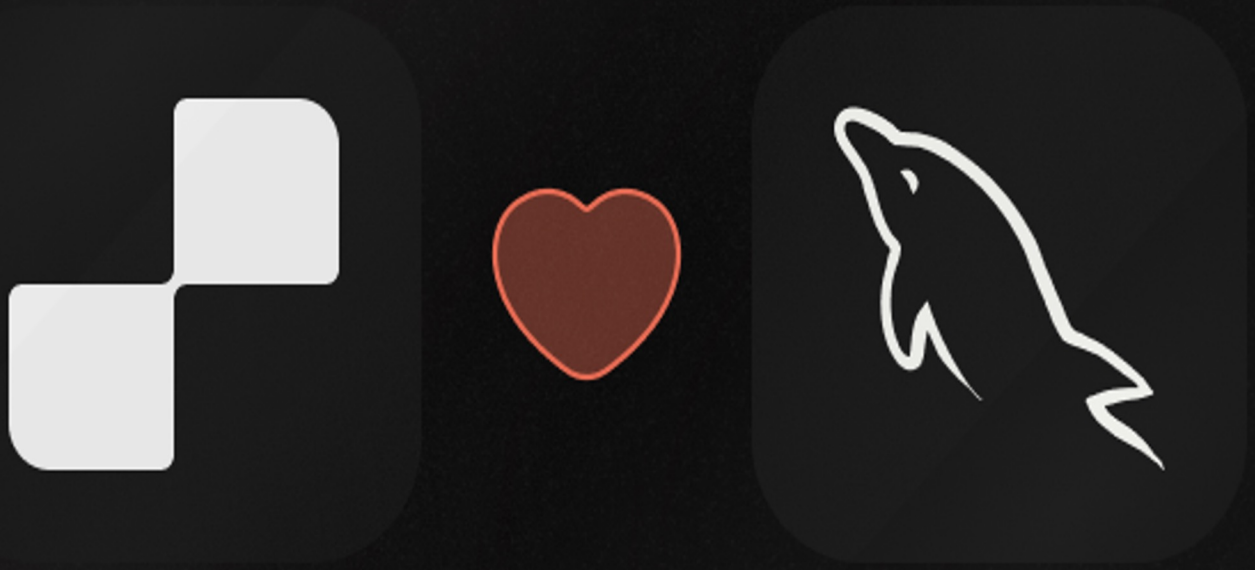
one writer · two map copies · one atomic pointer · readers stay lock-free

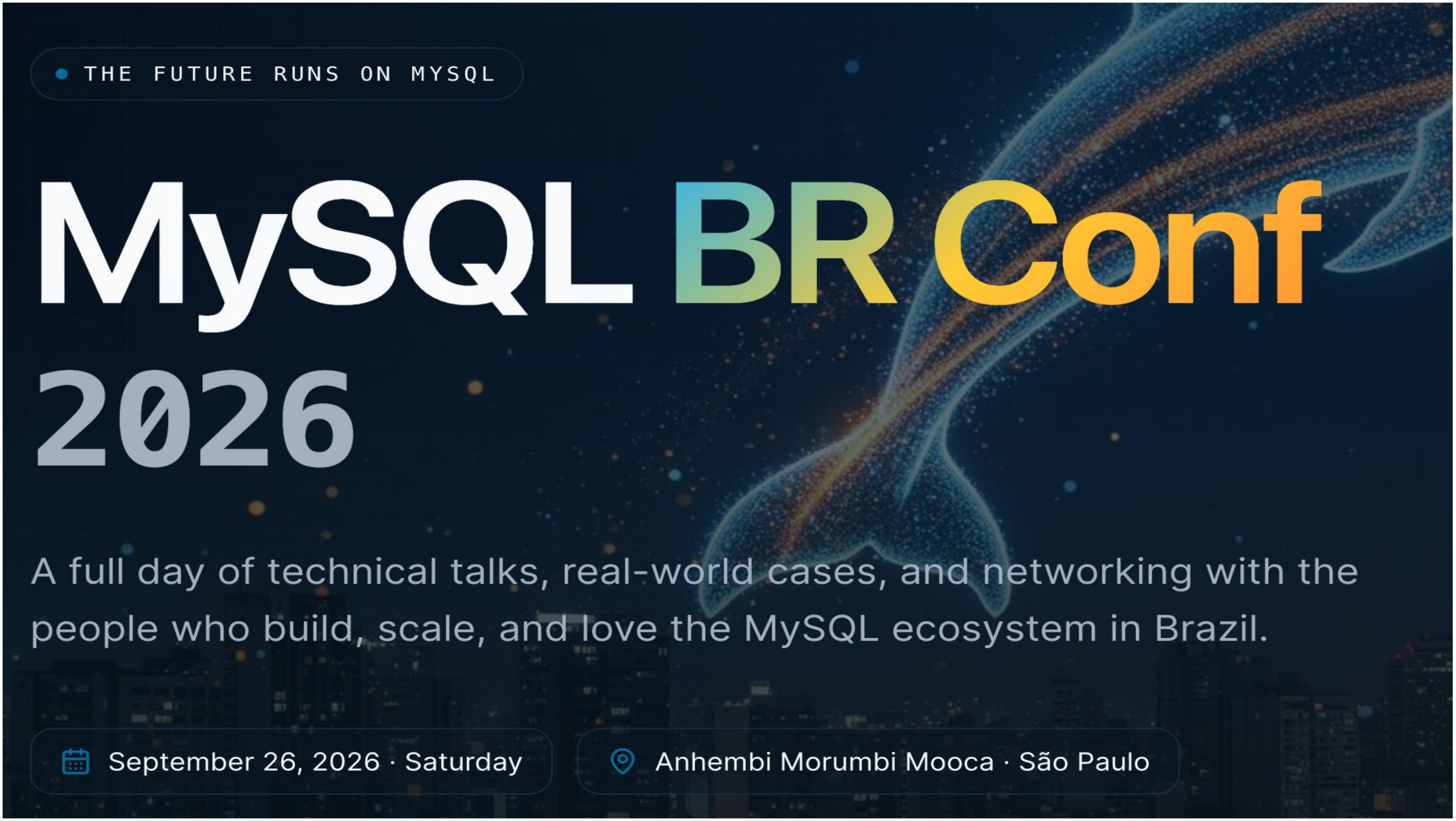


Demo


readyset | Cache Demo
MySQL vs ReadySet Shallow Cache
Database: MySQL
ReadySet Instance: Nano (1 vCPU / 0.5 GB)
Concurrency: 25
RS · upstream capped at 25
Apply
Start Workload
Stop Workload
Reset Stats
STOPPED
0/22 QUERIES CACHED
0.0% OF TRAFFIC SERVED FROM CACHE
1 READ REPLICA AVOIDED
WITHOUT READYSET
MySQL · direct
Round-trip to the database on every read
AVG QUERY LATENCY
-- ms
THROUGHPUT
-- q/s
P99
-- ms
VS
AWAITING WORKLOAD
WITH READYSET
Shallow cache · in-memory
Drop-in, wire-compatible, zero code changes
AVG QUERY LATENCY
-- ms
THROUGHPUT
-- q/s
P99
-- ms
LATENCY OVER TIME
LAST ~50 S
1.0 ms
0
upstream p50
upstream p99
readyset p50
readyset p99
QPS OVER TIME
LAST ~50 S
1.0 q/s
0
upstream
readyset
Hide details
THROUGHPUT IMPROVEMENT
--
-- vs -- q/s
MYSQL AVG
0
ms
READYSET AVG
0
ms
MYSQL QPS
0
q/s
READYSET QPS
0
q/s
READ REPLICAS NEEDED
--
to match RS throughput
CACHED
0
queries
PROXIED
22
queries
TRAFFIC CACHED
0%
of Zipfian traffic
RS HIT RATE
100
% (since RS start)
RS MISSES
8
hits 636,267
RS HEAP
28.2
MB heap (eviction ticks: 3,853)
RS QUERIES SERVED
636,275
cumulative
CACHE REFRESHES
32
to upstream
ACTIVE CLIENTS
1
connections
UPSTREAM CONNS
0
RS → upstream
DISTINCT SHAPES
40
query patterns
RS RESIDENT
118.8
MB (RSS)

Questions?




THE FUTURE RUNS ON MYSQL
MySQL BR Conf
2026
A full day of technical talks, real-world cases, and networking with the people who build, scale, and love the MySQL ecosystem in Brazil.
September 26, 2026 · Saturday
Anhembi Morumbi Mooca · São Paulo

Thank You!