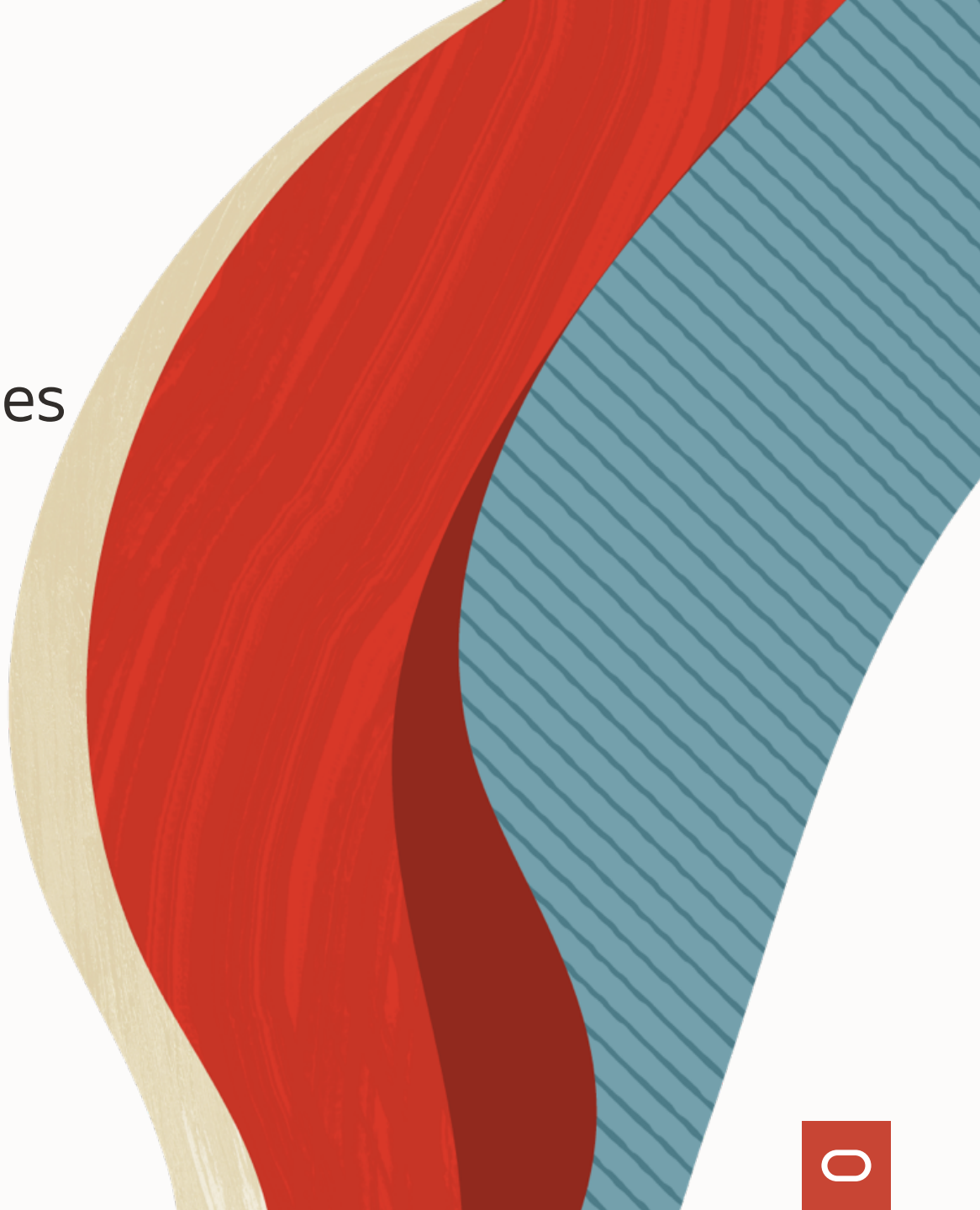
Accelerate your Applications 10x with True Cache

Dominic Giles
Distinguished Product Manager, Oracle Database

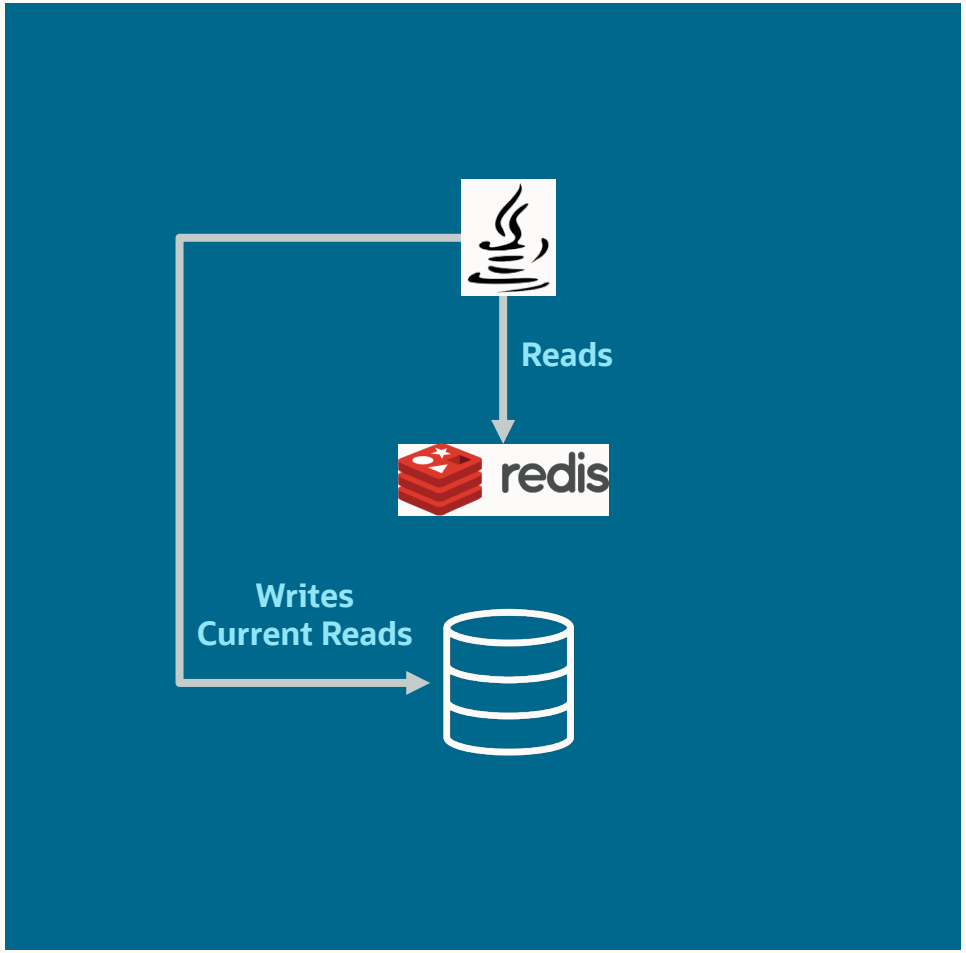
Agenda

- Cache use in Applications
- Problems with Conventional Caches
- True Cache
- Roadmap
- Performance Results



Cache use in Applications

- To accelerate performance, high throughput/ low latency applications use a separate caching tier
  - Examples include Redis & Memcached
- The primary source of data for the cache is an RDBMS like Oracle
- Some Reads done from the cache
  - Reads that can tolerate possibly stale data are directed to the cache
  - Reads that need the most current data are issued directly to the RDBMS
- Writes go to the backend RDBMS since that is the source of truth



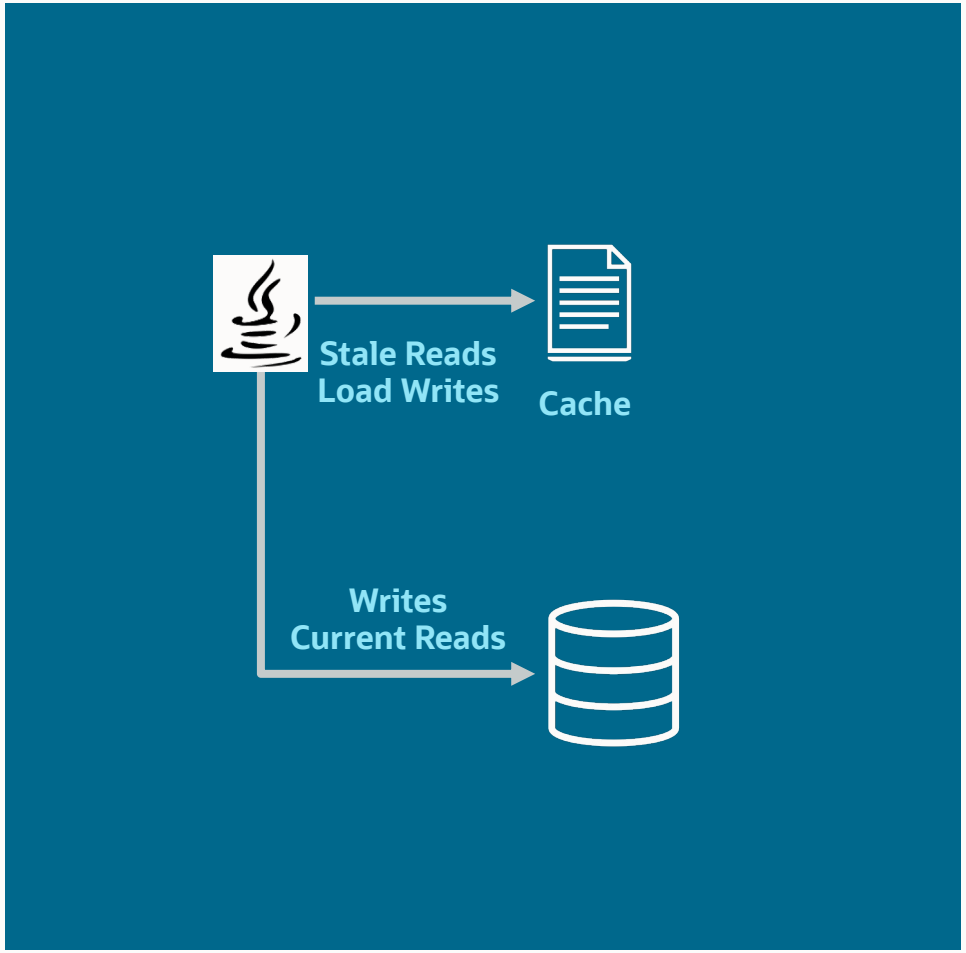
Shortcoming of Conventional Caches such as Redis or Memcached

"We had performance problem and now we also have a Caching Problem"

Developers are responsible for loading the cache

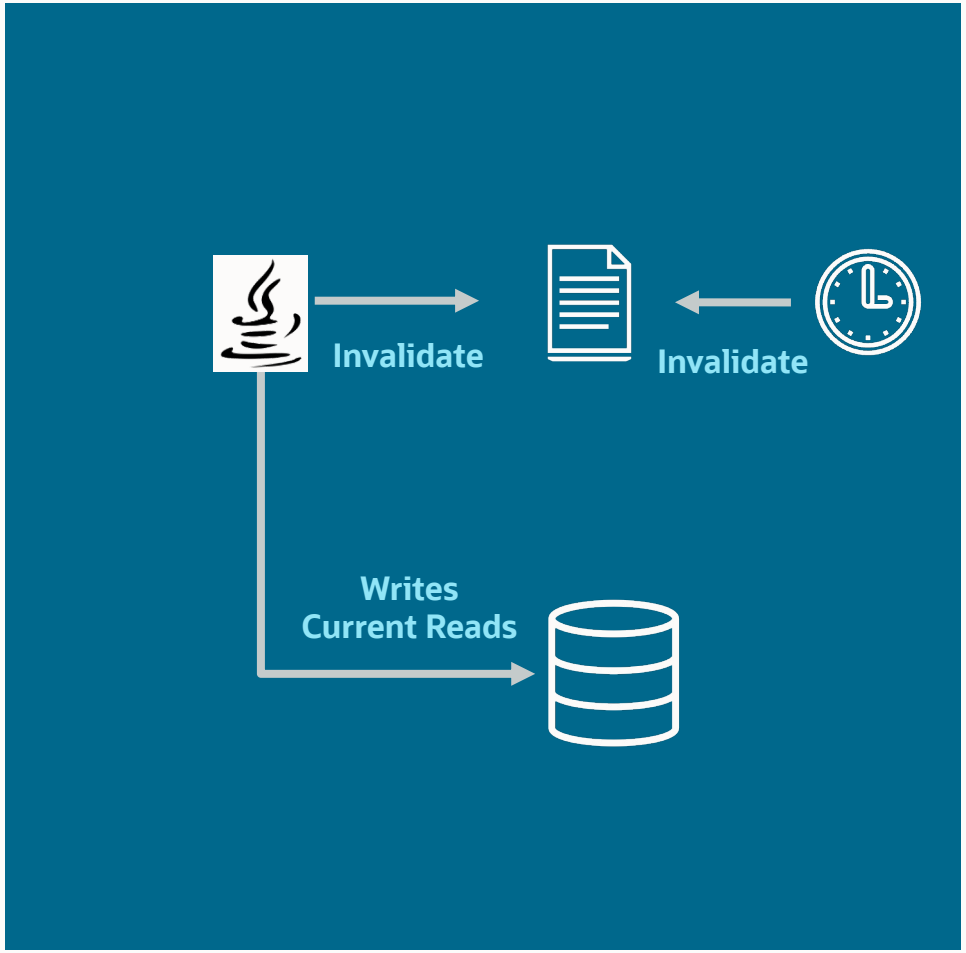
Developers are responsible for loading the cache, either

1. Load the cache at application startup
2. On each cache miss, read the value from the RDBMS, load it in the cache, and then return the data to the app



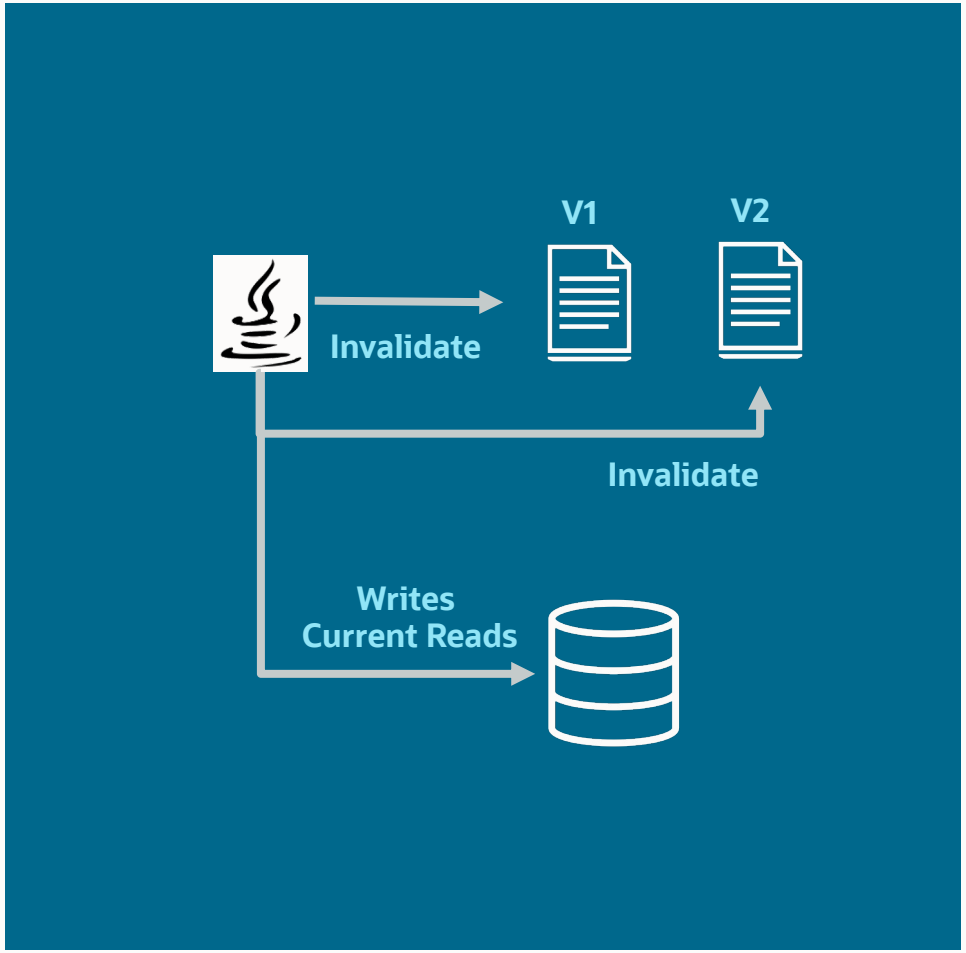
Developers are responsible for cache consistency with DB

- Cached content will become stale as the RDBMS is updated
  - Updates done directly against the database don't update the cache
  - Most apps are not written to manually maintain the cache for consistency and must therefore work directly against the database
- Some caches workaround this by allowing apps to configure Time to Live for different objects
  - Doesn't work well: frequent invalidations reduce cache effectiveness, less frequent invalidations result in a higher chance of getting stale data
  - Difficult to get this right, and keep it right as the application workload changes
- DDLs on RDBMS could invalidate entire cache contents (like purging a partition or dropping a column)



Developers are responsible for cache consistency with other caches

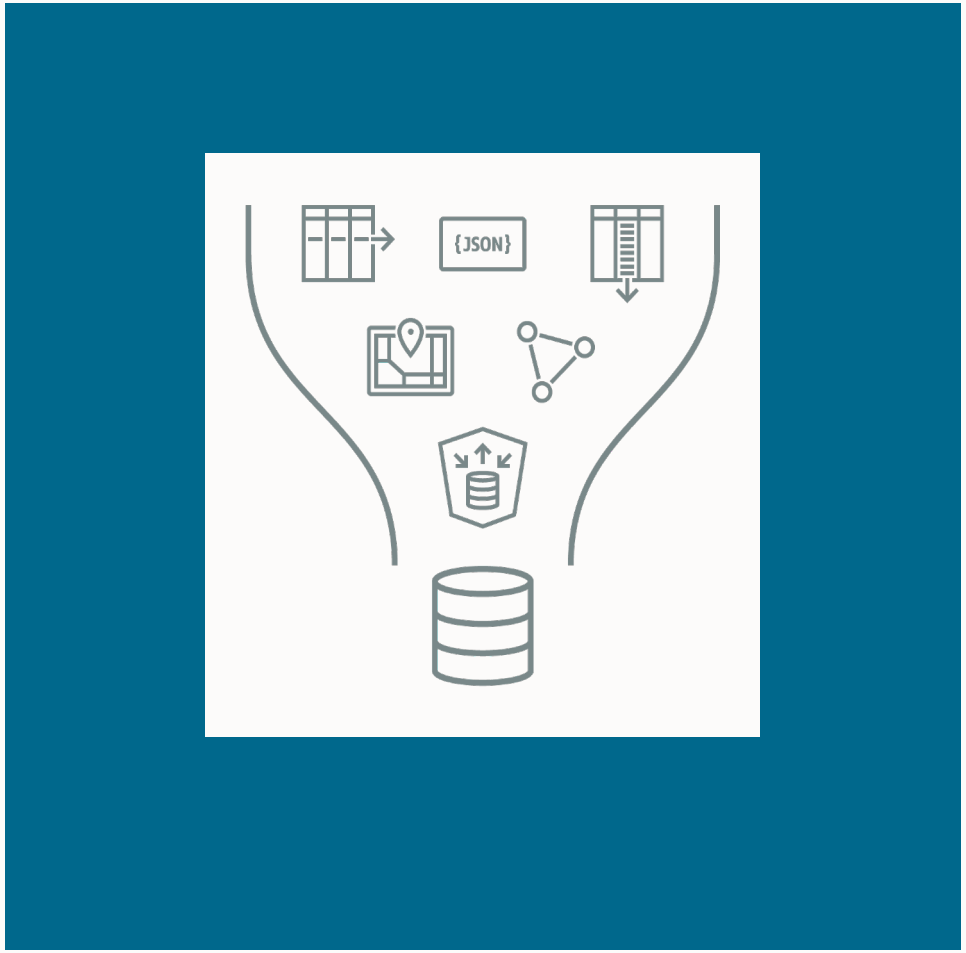
- When you read data from Oracle, we guarantee consistency among multiple values (Consistent Read)
  - Even if the data is updated by some other process
- Even if the application invalidates the cache on updates, it cannot update all the other caches [Cache inconsistent with other caches]
- For data stored in the cache, developers are responsible for enforcing data integrity and consistency [Cache inconsistent with itself]
  - Each object is independently maintained. This means if you read two objects, they are very likely to be from different points in time & hence inconsistent (your accounts won't balance)
- Limits cacheable data: Cannot cache complex nested objects with relationships



Limited data type and data format support

- Only scaler types like string, integer, and JSON supported
- Data cannot be accessed in Columnar formats for Analytics/Reporting
- Oracle database supports a variety of data types like Relational, JSON, Text, Spatial, and Graph and allows access in a row as well as columnar format

- Developers are responsible for handling additional data type conversion



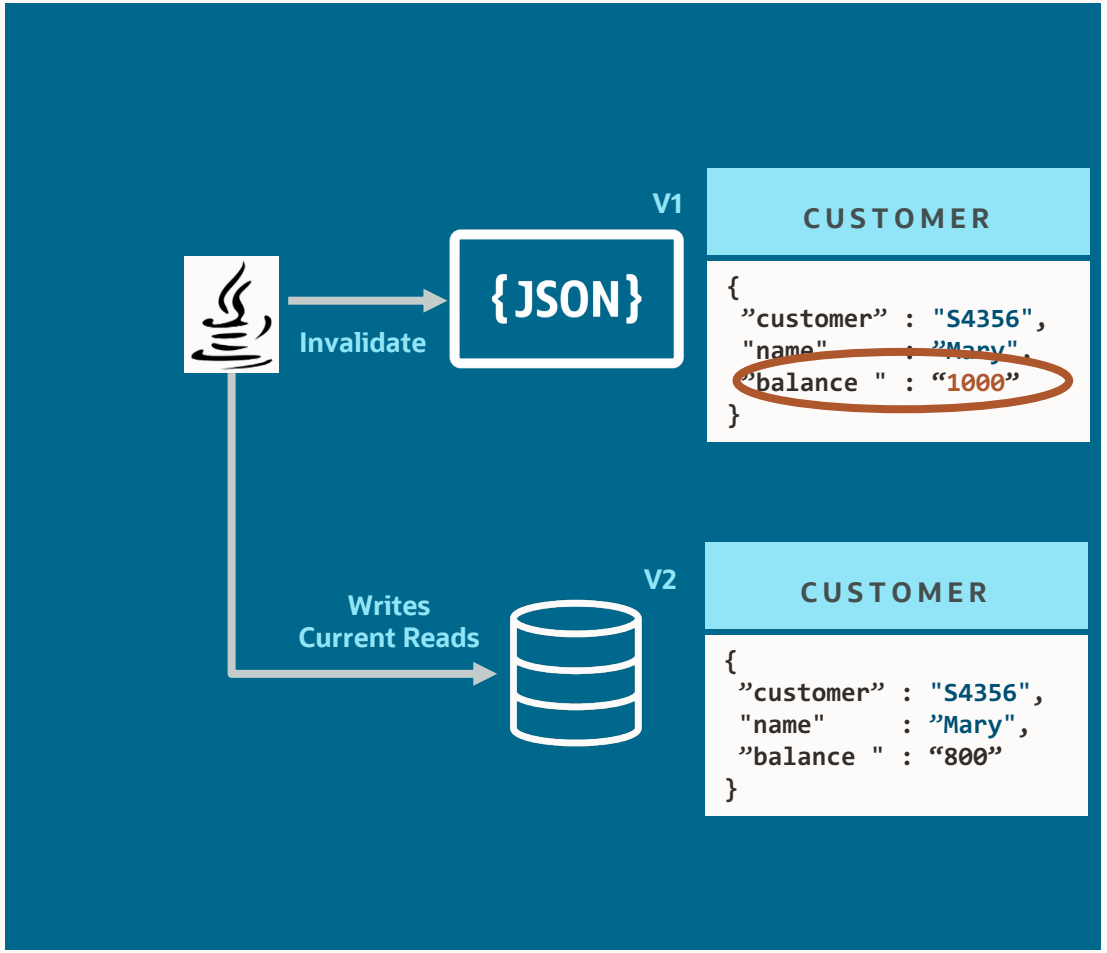
JSON Document exacerbates Caching Problem

When data is stored as JSON documents in an Object Cache, the caching problem gets exacerbated

- When a JSON object is stored in normalized tables, updates to the rows in the underlying tables should result in cache invalidation of the associated JSON attributes. This is hard to do
- Furthermore, need to handle DDL that changes the structure of the underlying tables (e.g., add column, or redefine type of existing column,…).

- No Document/Attribute level security in the cache
  - Cannot store sensitive data in cache

- Developers are responsible for updating and securing JSON documents in the cache

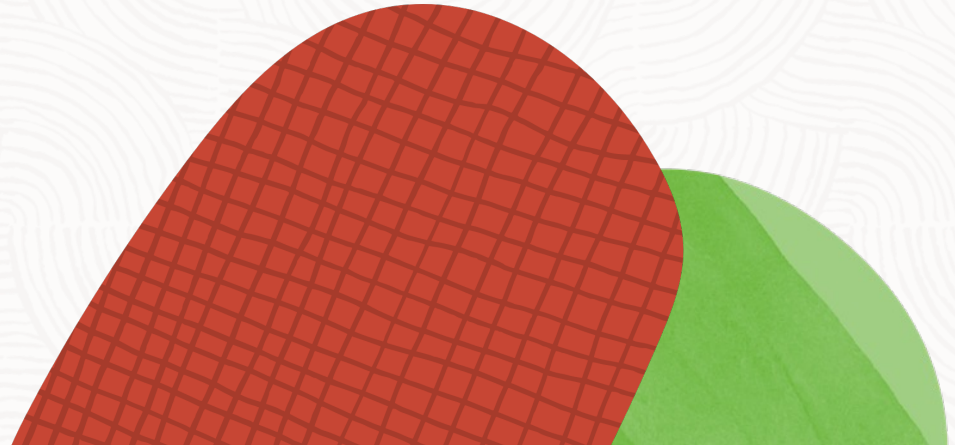


Missing Enterprise-Grade Features

- Suboptimal query performance for complex workloads
  - Missing features like sub-partitioning, secondary indices, parallel queries, multi-threading
- Less secure
  - Missing security features like user management, strong encryption, object-level security, row-level security, separation of admin role access
- Lower SLAs
  - Missing high availability features for multi-AZ/Region deployment
- Limited observability and missing enterprise-grade management

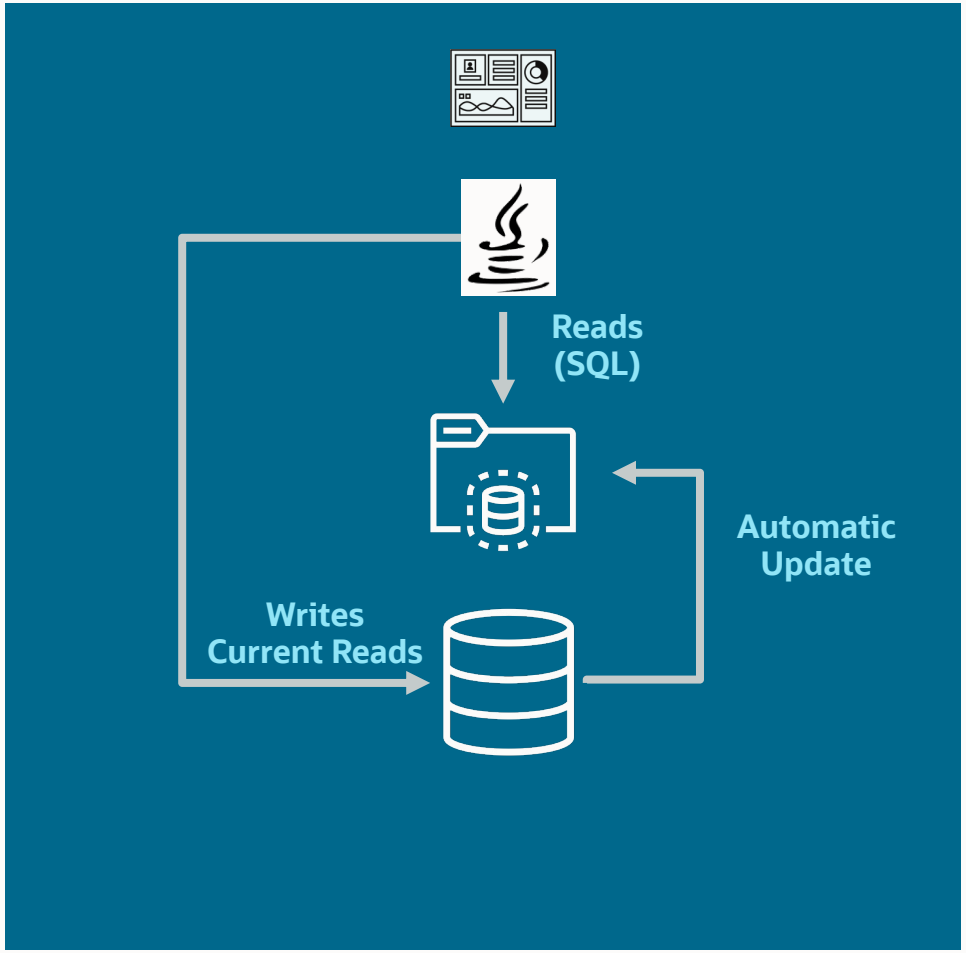
- Developers must write additional logic to compensate for the missing features

Oracle True Cache



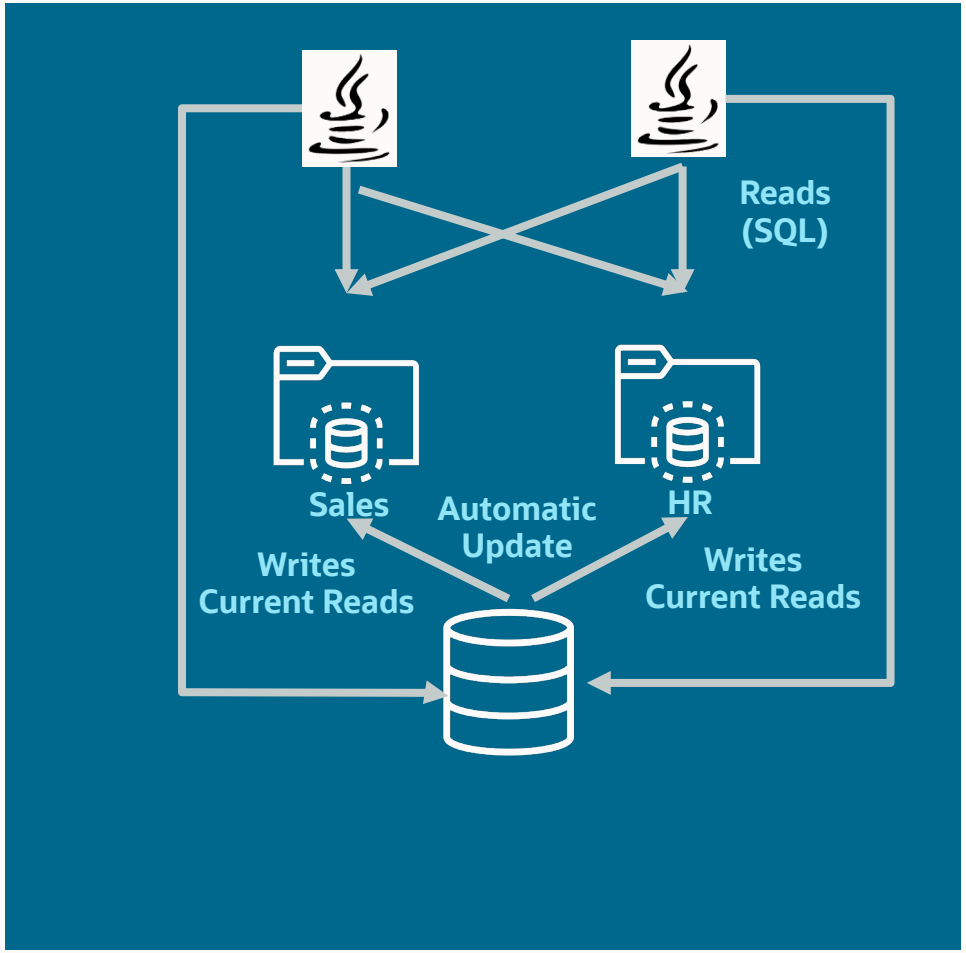
Oracle True Cache

- In-memory, consistent, and automatically managed SQL and key-value (object/JSON) cache
- Deployed as a read-only cache in front of an Oracle database
- Conceptually, a diskless Active Data Guard (ADG)
- True Cache instance is automatically updated
  - It uses Physical redo to apply updates, which is much more efficient than fetching data via SQL
- Queries that can use cached data can be issued to True Cache
  - True Cache can handle all the queries like the Oracle database
  - Data is always consistent across multiple tables and objects
  - DDLs automatically propagated to the cache



Scaling True Cache with partitioned configuration

- Different True Cache instances can support different database services
- Different True Cache instances cache different sets of data
- The aggregate cache size of all True Cache instances can be much bigger than that of the primary

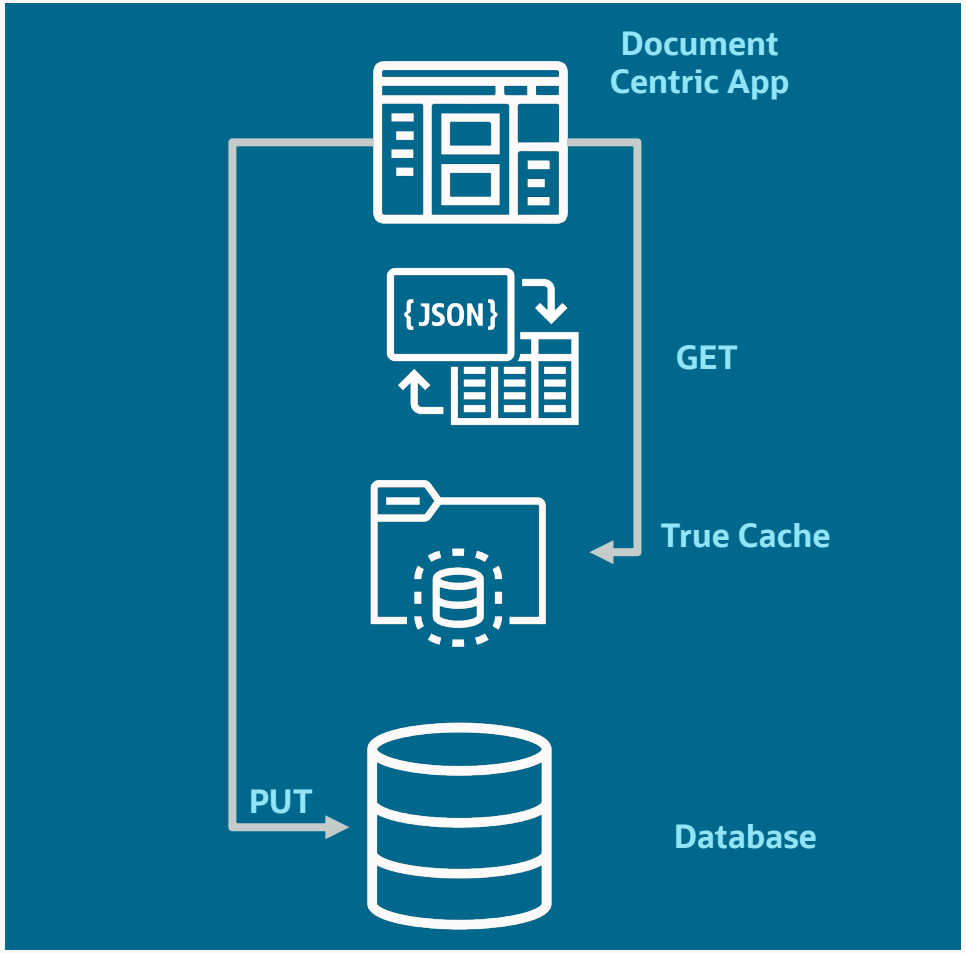


Global Proximity to Application

- Cache can be deployed closer to the Application if DB is in a remote location
- This is useful for Data Residency use cases
  - True Cache does not store any data on disk; hence is a data-processing tier
- This helps boost read performance significantly

True “Document and Relational” Cache

- With the new Document and Relational duality capabilities of Oracle database, developers can easily build document-centric apps
  - That stores new documents as relational data
  - Operate on existing relational data as documents
  - Create JSON microservice on top of a relational database
- JSON documents can be read directly from True Cache
  - GETs can be offloaded to the cache
  - PUTs go to the primary



True Cache mitigates deficiencies of conventional caches

Operation	Conventional Caches	True Cache
Loading the cache	Developer responsibility	**Automated**
Cache consistency with DB	Developer responsibility	**Automated**
Cache consistency with objects in the same cache	Developer responsibility	**Automated**
Cache consistency with other caches	Developer responsibility	**Automated**
Complex data type support	Developer responsibility	**Automated**
Full JSON support	Developer responsibility	**Automated**
Comprehensive security	Developer responsibility	**In-Built**
Parallel processing	Developer responsibility	**In-Built**
High Availability	Developer responsibility	**In-Built**

True Cache Use Cases

True Cache can be deployed as a:

Mid-Tier Cache

Edge Cache

Cross-Region Cache

Cross-Cloud Cache

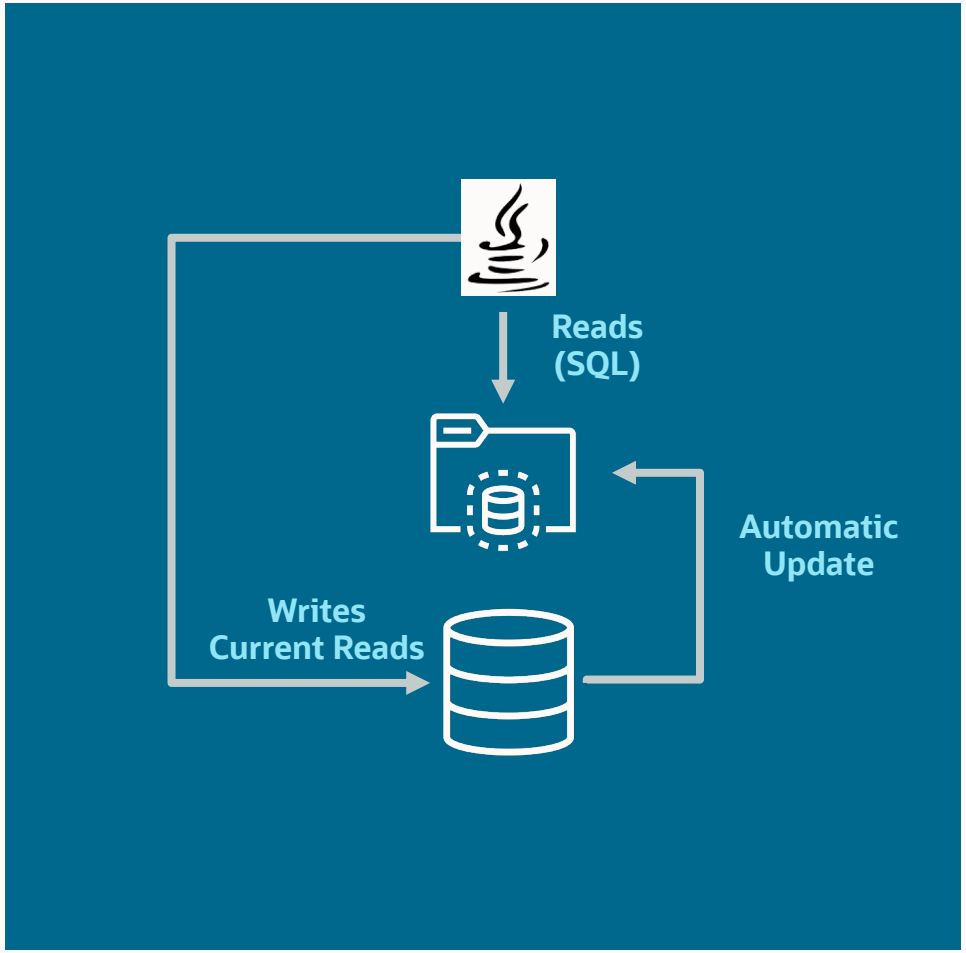
Benefit To The Business

- Cost Savings
  - Improve application performance without rewriting the applications,
  - No investment in additional caching products and skills
  - Single cache for different data types and formats (document, relational, columnar, row)
  - Offload caching to commodity hardware from the database (which might be on more expensive hardware)
- Stronger security, which comes with Oracle database
- Leverage the full power of Oracle database for newer application functionality (like indices, secondary indices, parallel query, compression, partitioning, security)

How To Use True Cache

The application can query data from True Cache in the following two modes

- The application maintains two connections: a read-only connection to True Cache and a read-write connection to the primary, and uses the read-only connection for offloading queries to True Cache
- The application maintains one connection. The JDBC driver maintains two connections and does the read-write split under application control
  - The application uses setReadOnly() calls to mark some sections of code as “read-only”
  - This is an existing JDBC API which MySQL already uses for similar purposes
  - JDBC driver will send read-only queries to True Cache and DMLs and DDLs to the primary



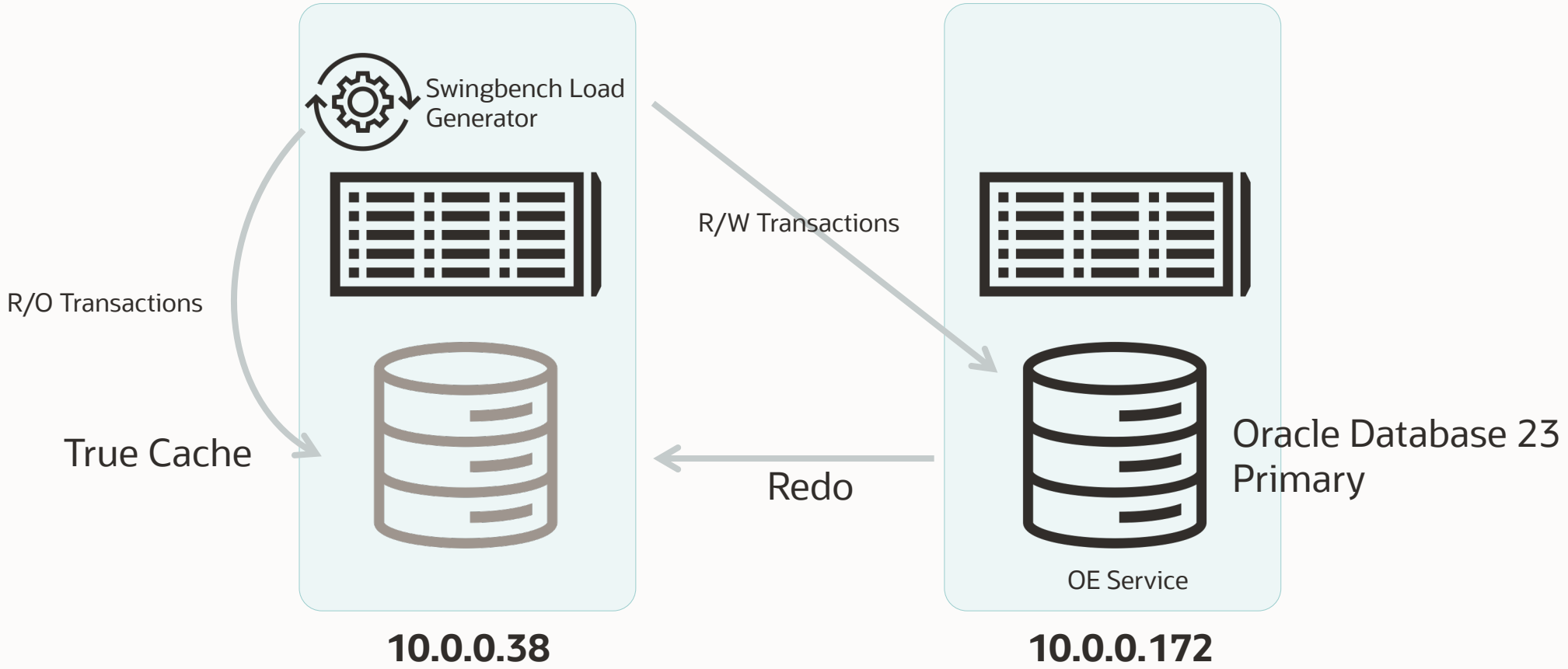
Customer Feedback

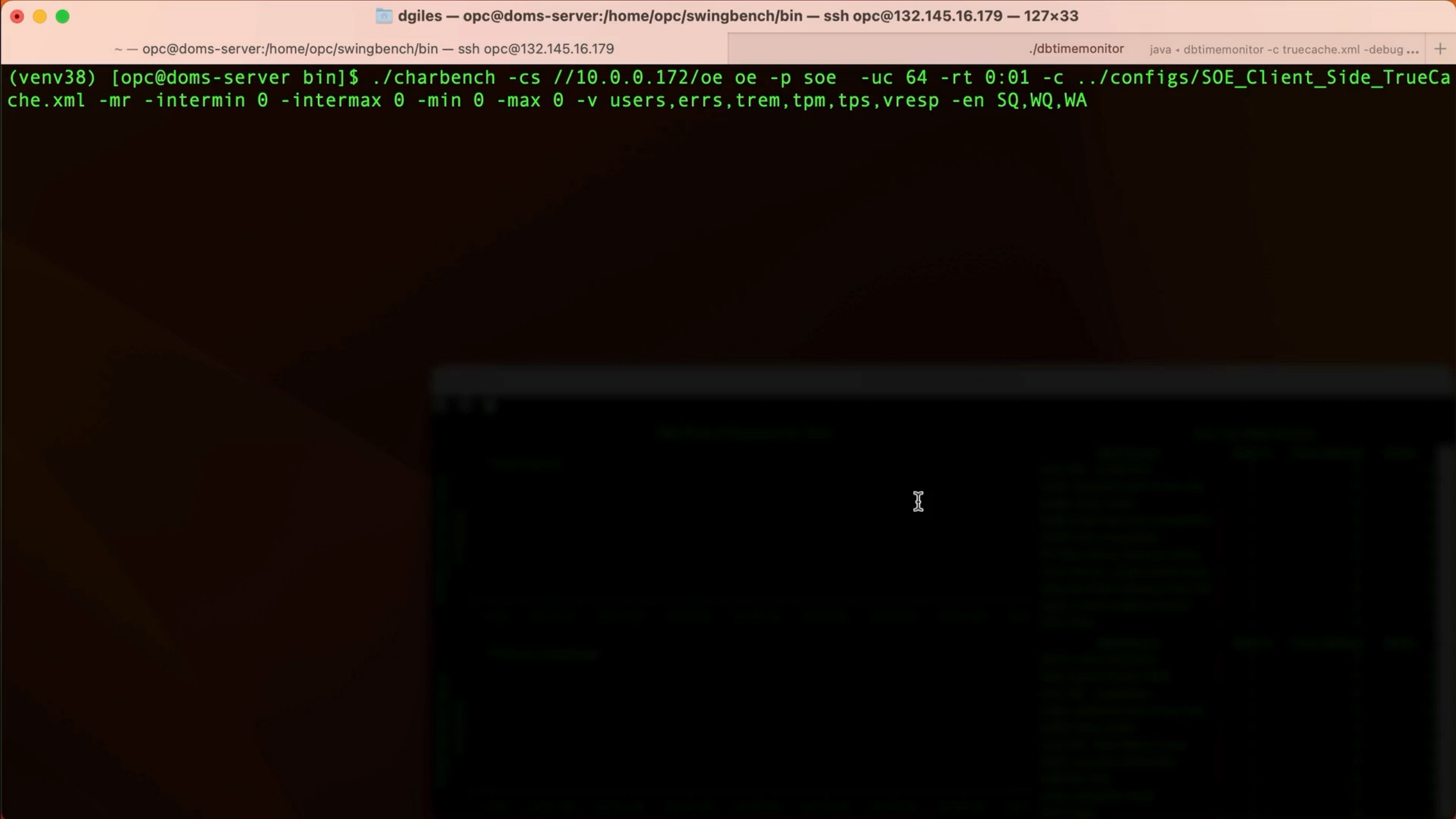
- Mobile Phone Manufacture – Offload read-only queries to True Cache as the primary is maxed out with vertical scaling
- Financial Institution – *"In the real-time fraud detection/blocking world, they are always trying to push the envelope for performance (+AH4-100ms SQL response time SLAs) and run frequent batches to try and keep the most current data in the fraud models. If we are able to shift their OLTP SELECT SQLs to the app server with True Cache, we can reduce the load and blocking SQLs on the database, which I think they would be very receptive to."*
- Marketing – Would like to use True Cache for real-time marketing campaigns as it will obviate another persistence store they have to manage, code against, and pay for.
- Oracle Fusion Applications – Cache for Business Object layer for objects that do not change as often
- Oracle Banking (Financial Services GBU) – Testing to offload Reads to Cache for core banking application, which is read intensive application

Demo

True Cache Demo

This demo runs an Order Entry workload against an Oracle Database 23c database. If we mark read only transactions in our application code, we can them run against True Cache. The cache is kept up to date and consistent with the the primary database. The application transaction latency benefits from the colocation with the True Cache instance




dgiles — opc@doms-server:/home/opc/swingbench/bin — ssh opc@132.145.16.179 — 127×33
~ — opc@doms-server:/home/opc/swingbench/bin — ssh opc@132.145.16.179
./dbtimemonitor
java ◂ dbtimemonitor -c truecache.xml -debug ...
(venv38) [opc@doms-server bin]$ ./charbench -cs //10.0.0.172/oe oe -p soe -uc 64 -rt 0:01 -c ../configs/SOE_Client_Side_TrueCache.xml -mr -intermin 0 -intermax 0 -min 0 -max 0 -v users,errs,trem,tpm,tps,vresp -en SQ,WQ,WA

Example Performance Test Results

Linear throughput improvement by adding cheaper True Cache nodes

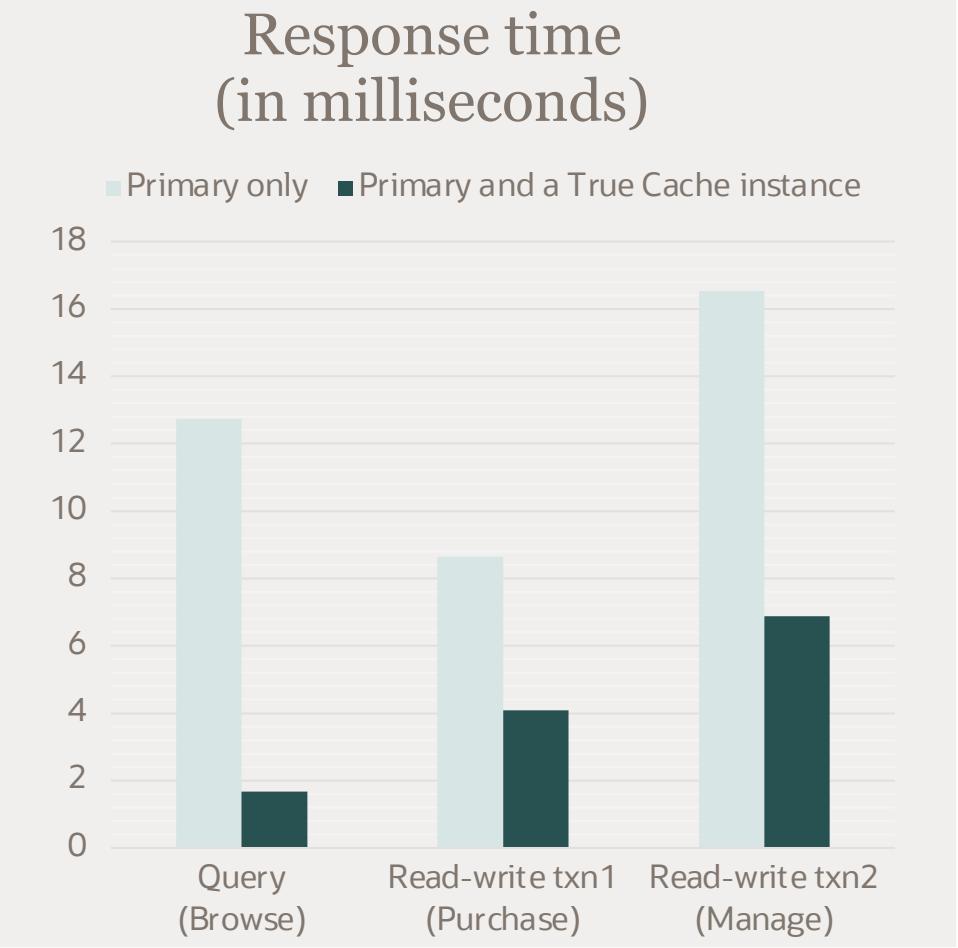
- Workload: 84% selects, 16% inserts/updates OLTP JDBC workload modeling online shopping and e-commerce
- The primary and each True Cache instance all have 48 cores
- Primary SGA: 236G; each True Cache instance SGA: 128G
- True Cache nodes are cheaper: diskless, less memory, can be commodity hardware
- Close to linear speedup

Linear Throughput Scalability

# of True Cache Instances	Throughput Improvement
1	1.8x
2	2.6x

Response Time Improvement by adding one True Cache instance

- Same OLTP JDBC workload
- Primary only: long response time due to queuing effect when CPU is maxed out
- Adding a True Cache instance reduces the queuing effect because the same workload is distributed to two instances
- Adding a True Cache instance can reduce response time significantly for the same workload



10X Throughput improvement with bigger aggregate cache

- With partitioned configuration, the aggregate cache size of all True Cache instances can be much larger than that of the primary
- Simulation:
  - Primary: 10G SGA
  - True Cache instance: 100G SGA
  - Database size: 116G
  - Workload: based on Swingbench

10X

True Cache speed up for cache hit Indexed OLTP query

Response time improvement due to cache proximity

- Primary: Mumbai, Application: Phoenix, SQL roundtrip: 600 ms due to distance
- Adding a True Cache instance in Phoenix reduces SQL round trip to 2 ms for cache hit queries, a 300x speedup
- One can eliminate cache miss for tables that fit in memory:
  - Allocate a keep buffer pool on True Cache
  - Assign the table to keep the pool on primary
  - Perform an initial table scan on True Cache

300X

True Cache speed up for cache hit for Indexed OLTP query

Scenario	Response Time
Primary only	600 ms
Primary + True cache, cache hits only	2 ms

Our mission is to help people see data in new ways, discover insights, unlock endless possibilities.

Thank you…


ORACLE