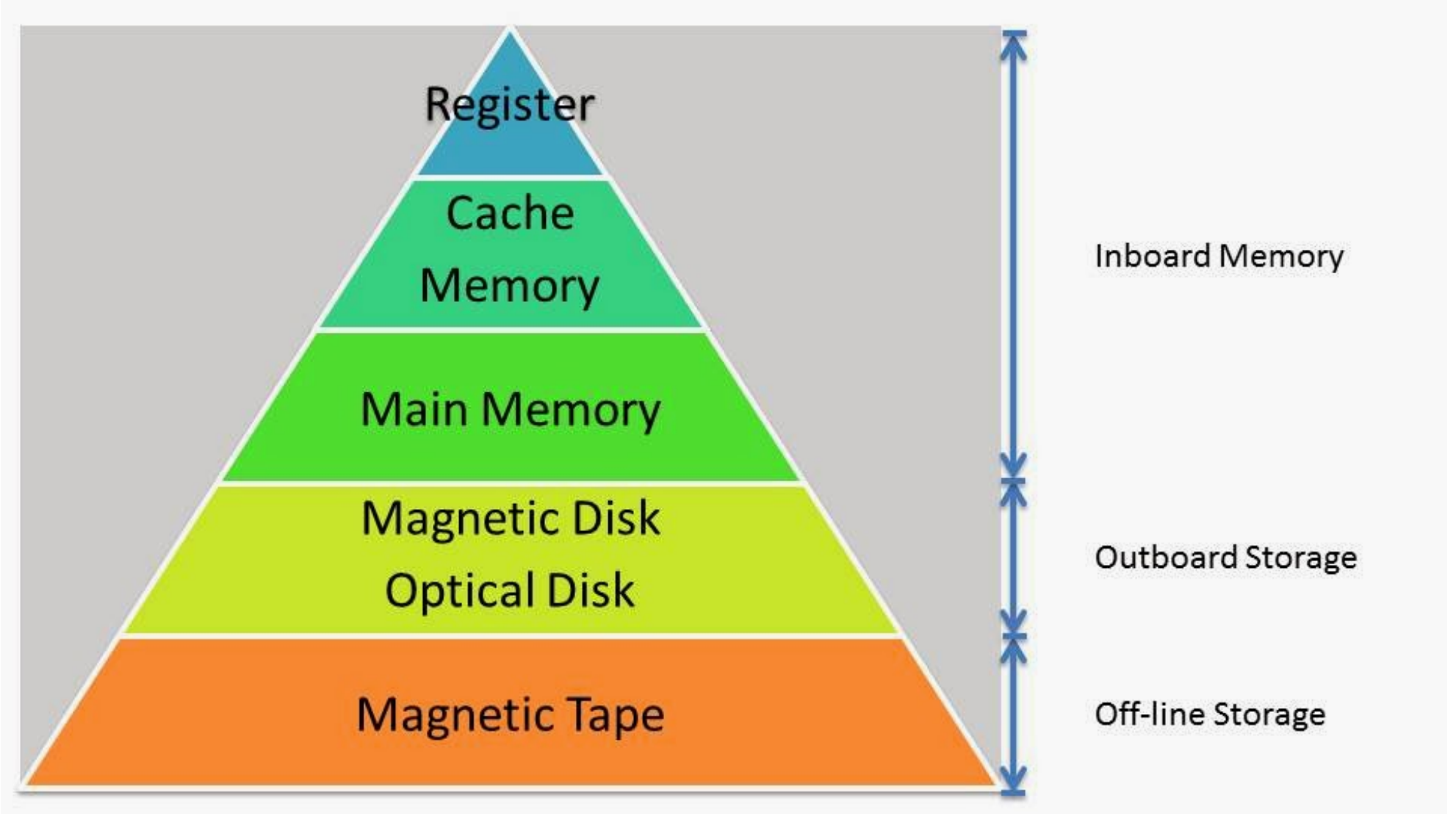
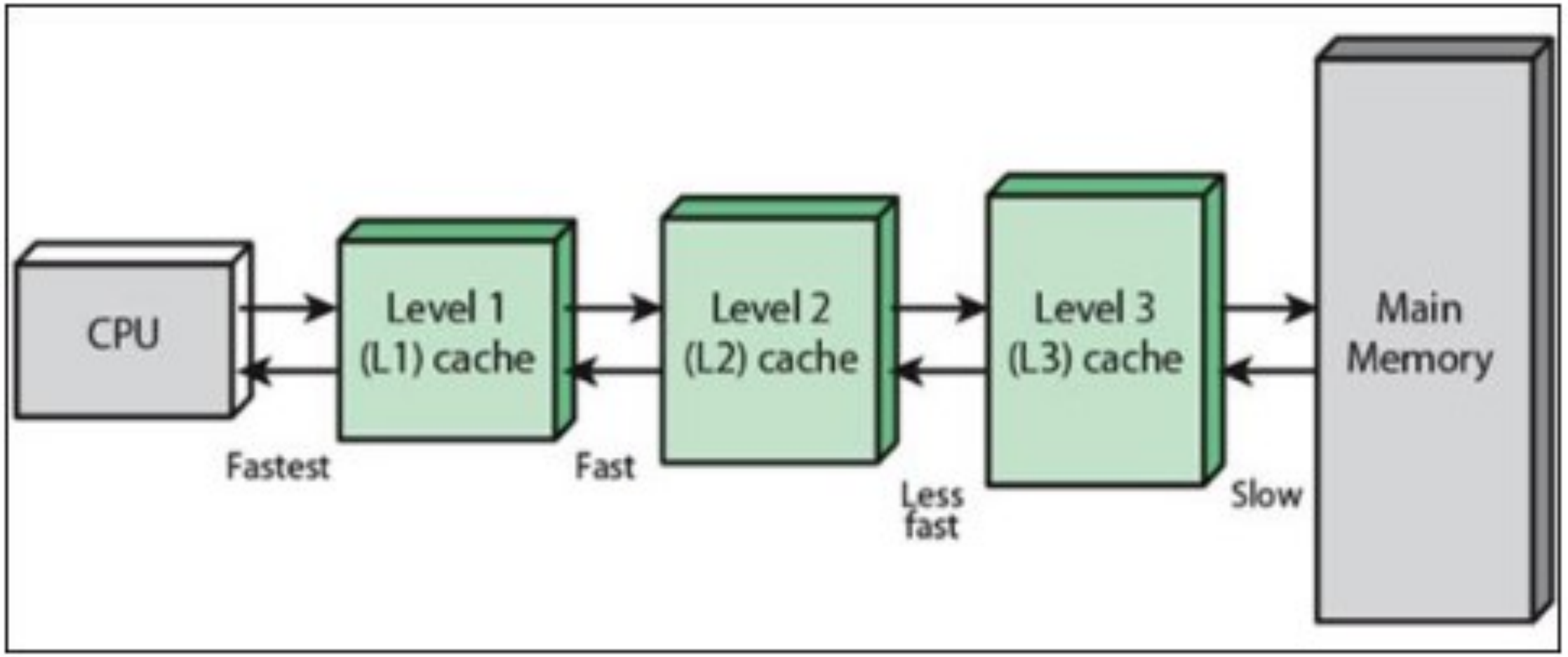
Using SQL and PLSQL for Mid-Tier Database Caching

Doug Hood
@ScalableDBDoug
Consulting Member of Technical Staff
Product Manager TimesTen In-Memory Database, Oracle In-Memory & Oracle NoSQL
May 16, 2019

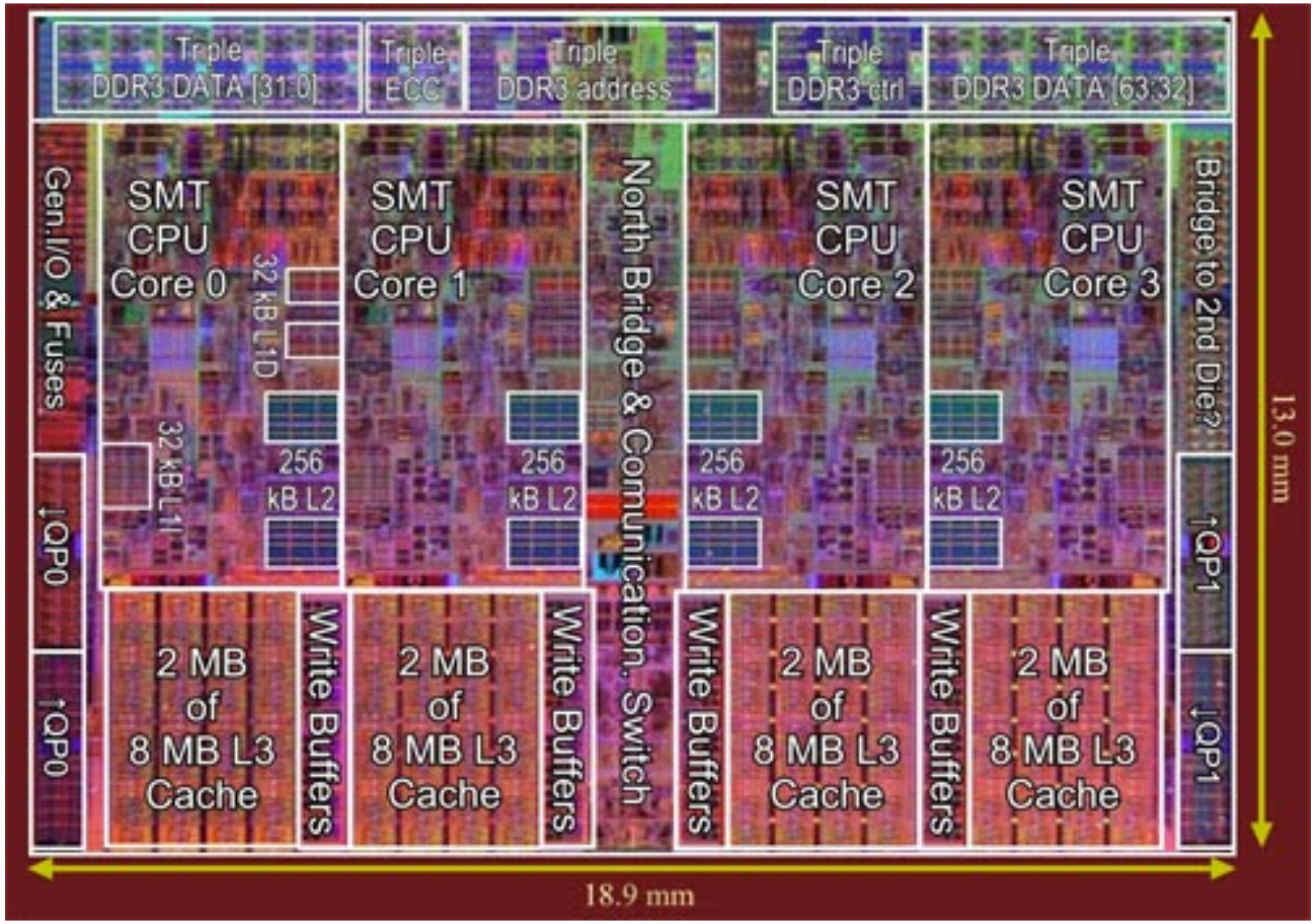
Memory Hierarchies (old school)



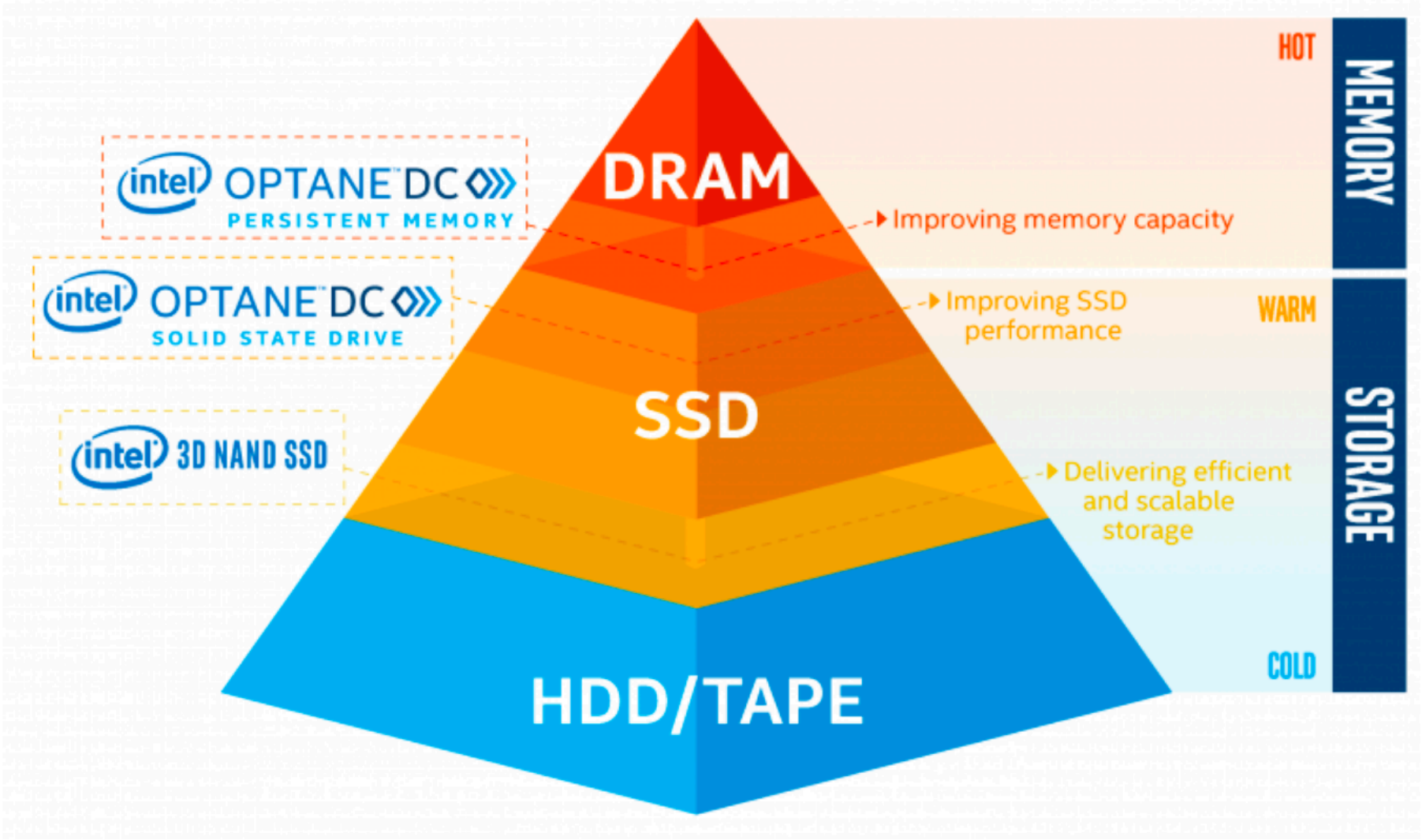
Memory Hierarchies (any modern CPU)



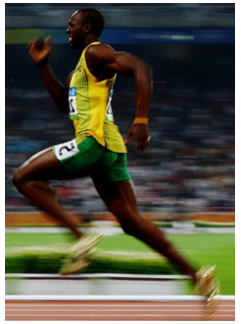
Memory Hierarchies (L1, L2 and L3 caches & 4 cores)



Memory Hierarchies (Intel's vision)



Latency, Throughput and Scalability

Latency		**How quickly can one operation complete** **One sprinter in 9.58 seconds** **~ 40 km/h for 100M [2009]**
Throughput		**How quickly can many operations complete** **Ten sprinters in under 11 seconds** **~ 40 km/h for 100m [2009]**
Scalability		**By adding more resources can throughput keep increasing** **33 cars on 2.5 mile oval track** **~250 km/h for 804 km [Indy 500, 2017]**

Latency Optimization

Before	After
43	3
69	6
105	8
121	20
140	18
163	19
231	18

ORACLE

RAC – Real Application Clusters

Oracle 11.2.0.4 RAC
Oracle Sun X7-2L
NVMe Storage
Over 50 Million Users

Latency is in Micro Seconds …

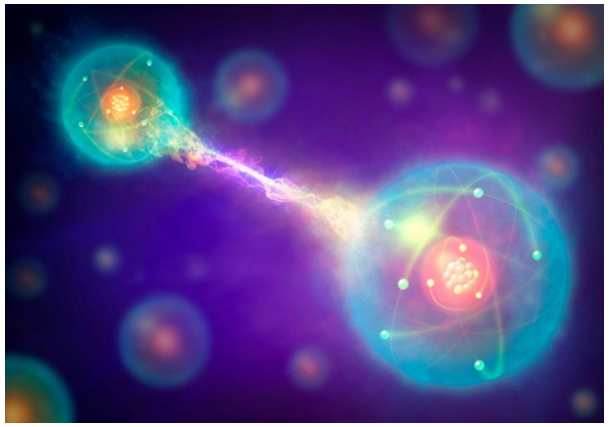
ORACLE

Why Cache Data?

Why Cache Data?

- To get lower latency for SQL statements
- To get more throughput
- To get more scalability

Caching Challenges

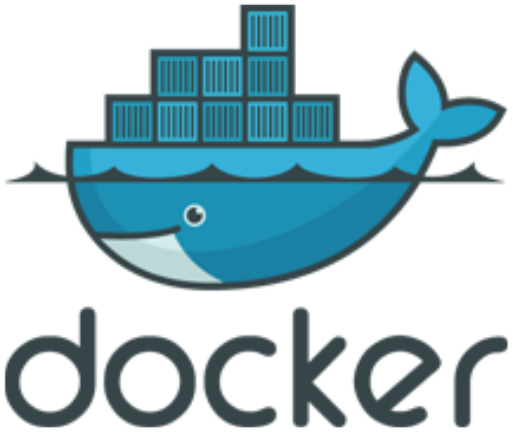


The Business Problem

- DB apps, make them faster and cheaper
- Do more with less

The DevOps Challenge

- Make it boring, no surprises
- Standardized, run in VMs or containers
- **Everything** is **automated**
- Figure out everything that can go wrong and **be well behaved**

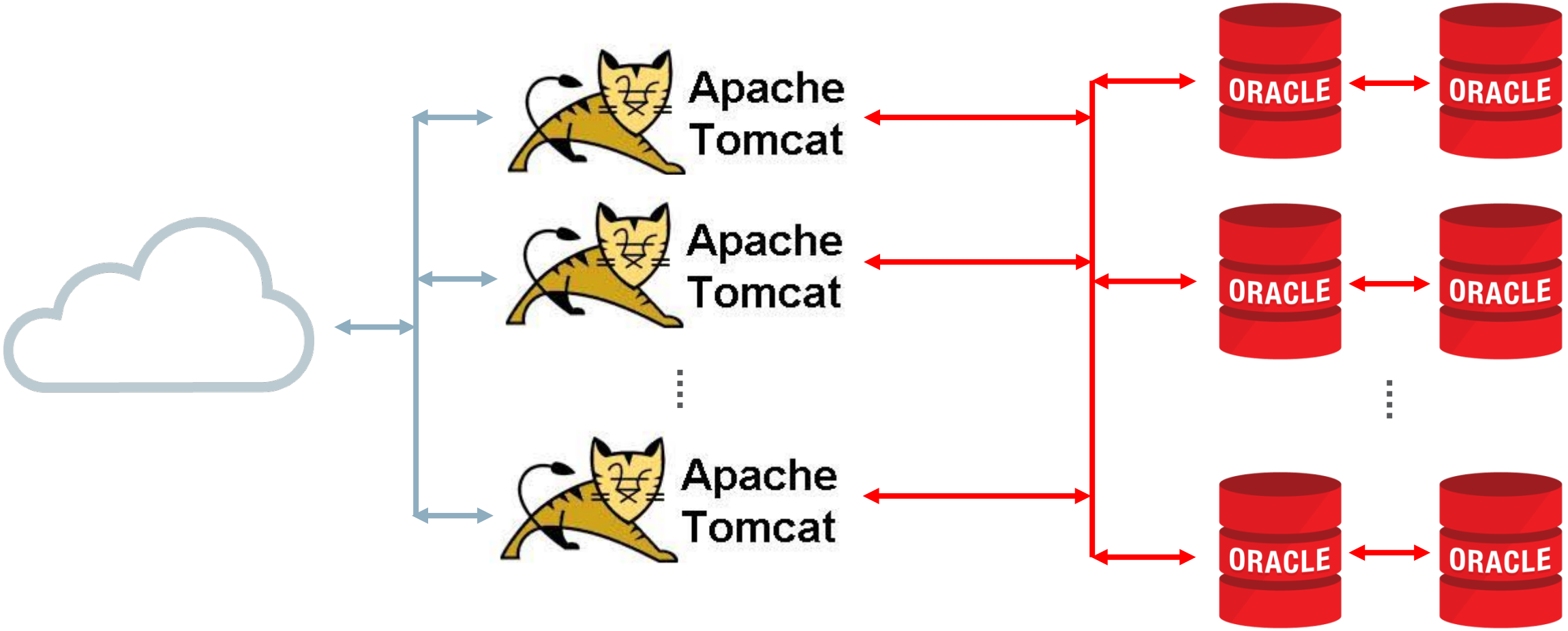


Linux KVM

Architectures and technical challenges

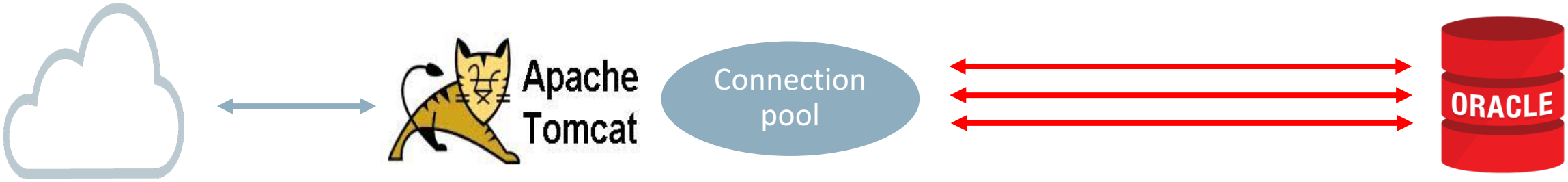
- How many users? How many concurrent users?
- How many concurrent Oracle DB connections?
- How many concurrent Application Server connections?
- What about HA? What about DR?
- What about latency?

Architectures and technical challenges



Can we do better?

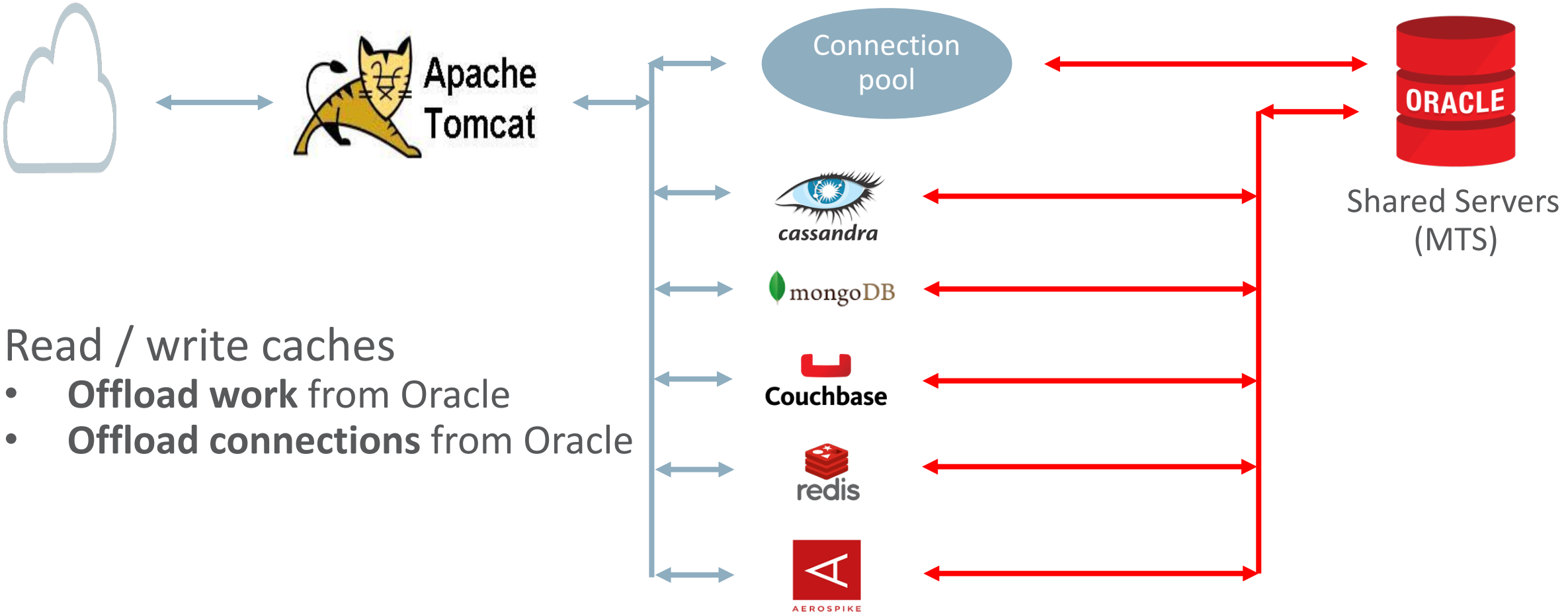
Architectures and technical challenges



Shared Servers (MTS)

Can we do better?

Architectures and technical challenges



Read / write caches

- **Offload work** from Oracle
- **Offload connections** from Oracle

Can we do better?

Architectures and technical challenges

- **Go faster** than NoSQL solutions & **minimize** the **hardware cost**
  - Lower latency at the **99th percentile**

- Simple and scalable is good,
  but how many machines do you need?
- What about the 95th and 99th percentile for latency?
- What about cache misses?
- How is the cache kept up to date?

99th percentile latency can be as high as 324 ms …

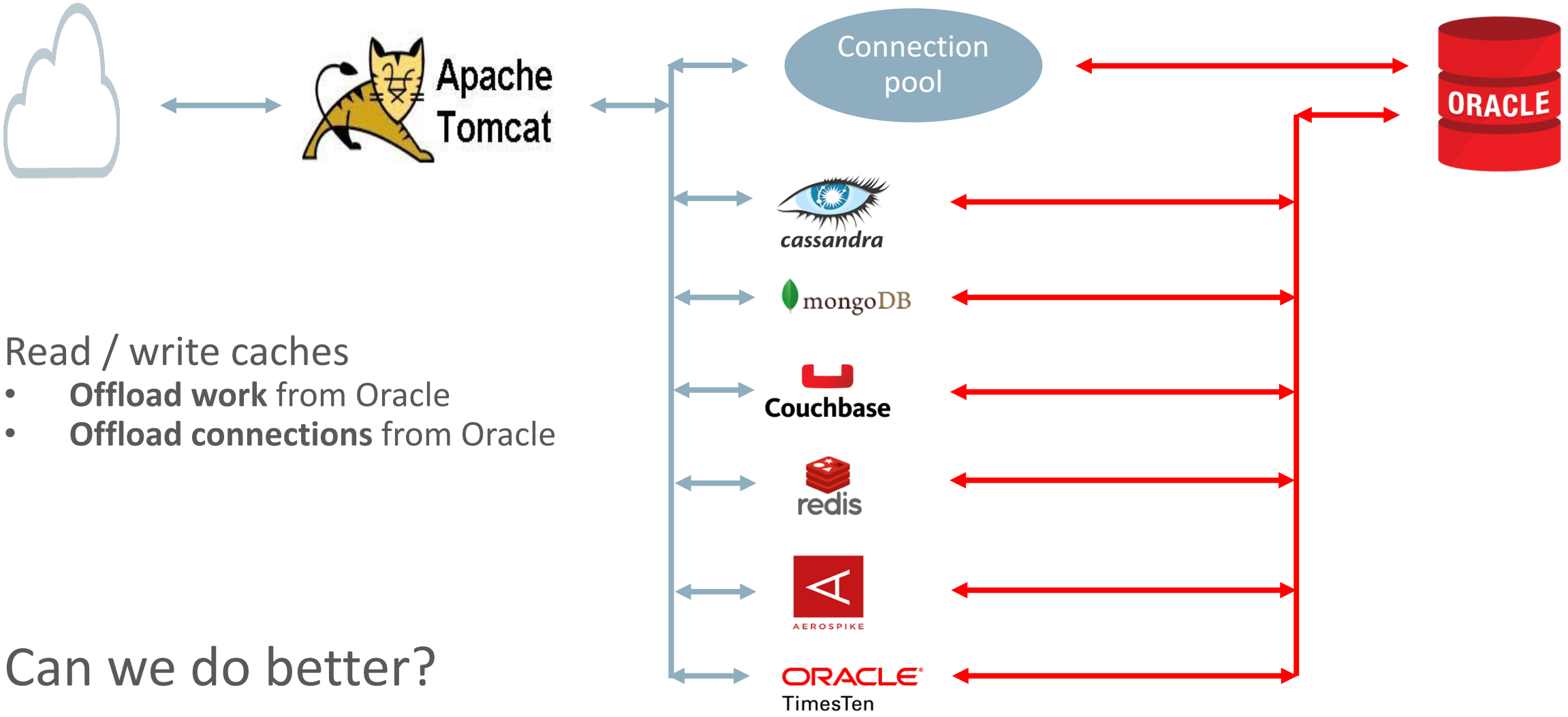
100th percentile latency is many seconds, timeout > 100 ms

YCSB Workload B (95% read, 5% write)

Database	TPS	Nodes
Cassandra	221K	32
MongoDB	260K	2
CouchBase	454K	9
Redis	1M	3
VoltDB	1.5M	6
AeroSpike	1.6M	1
TimesTen	2.8M	1

Lets talk

Architectures and technical challenges



Read / write caches

- **Offload work** from Oracle
- **Offload connections** from Oracle

Can we do better?

Customer measured TimesTen Cache latency

- 97% cache hit ratio
  - 3% of the time the data was not in the cache, so needed a round trip to Oracle
- 99th percentile latency = 1ms

Percentile	Latency in ms
87	0.016
98	0.125
99.7	1
99.99	8
100	423

NoSQL was 324 ms

Customer measured TimesTen Cache latency

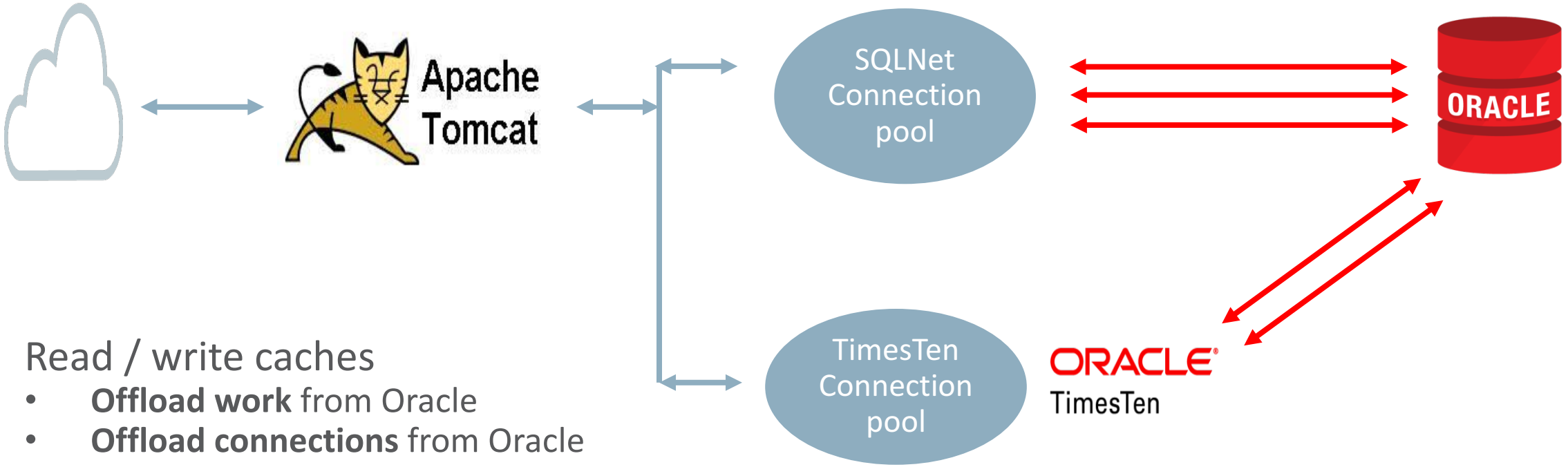
- 97% cache hit ratio
  - 3% of the time the data was not in the cache, so needed a round trip to Oracle
- 99th percentile latency = 1ms

Percentile	Latency in ms
87	0.016
98	0.125
99.7	1
99.99	8
100	423

Working to improve this

- Goal is < 50ms for 100th percentile
- Also define a 300 ms timeout

Improving the 100th percentile [max latency]



Read / write caches

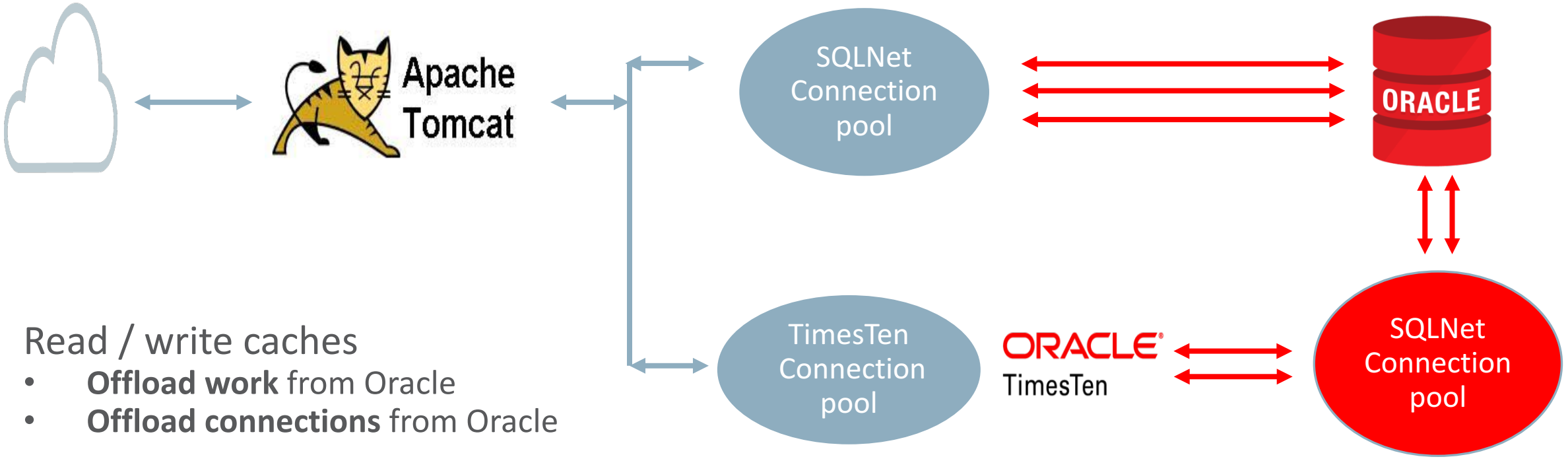
- **Offload work** from Oracle
- **Offload connections** from Oracle

For cache misses,
create a new SQLNet connection!

This can be **really slow …**

Can we do better?

Improving the 100th percentile [max latency]



Read / write caches

- **Offload work** from Oracle
- **Offload connections** from Oracle

For cache misses,
get connection from TT OCI CP
rather than the shared pool

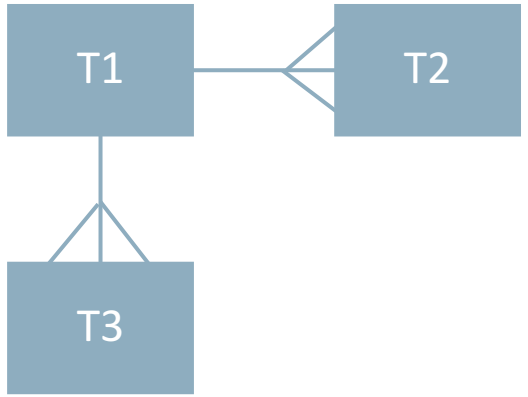
Tune the pool for the expected min/max

Can we do better?

How is it possible

- Use a **really fast** In-memory SQL RDBMS
- Use a **Cache DB** that just requires configuration
- Some hardware and software tuning

ORACLE
TimesTen



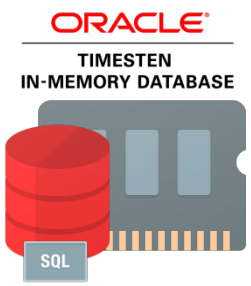
- Prepare & bind SQL statements
- Use the relevant indexes
- Do sensible joins
- Update statistics for SQL optimizer
- Check TimesTen SQL stats for SQL and IO bottlenecks

Most Widely Used Relational In-Memory Database

Deployed by Thousands of Companies

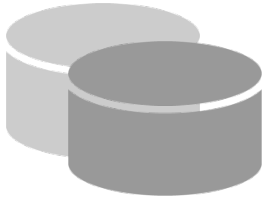
Oracle TimesTen In-Memory Database

Relational Database



- Pure in-memory
- ACID compliant
- Standard SQL / PLSQL
- Entire database in DRAM

Persistent and Recoverable

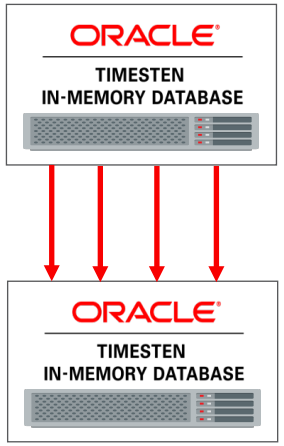


- Database and Transaction logs persisted on local disk or flash storage
- Replication to standby and DR systems

Extremely Fast

- Microseconds response time
- Very high throughput

Highly Available

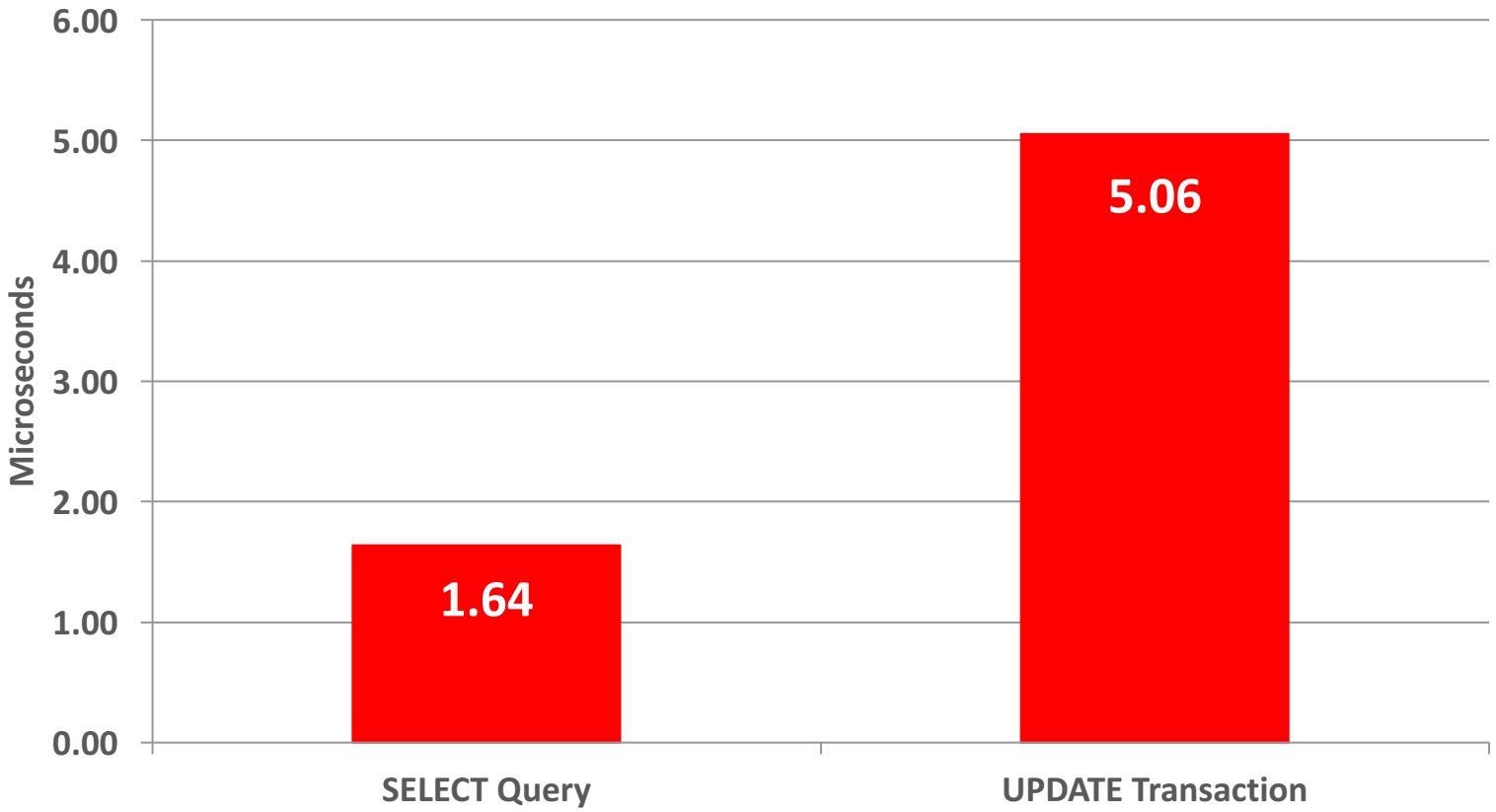


- Active-Standby and multi-master replication
- Very high performance parallel replication
- HA and Disaster Recovery

Performance – Response Time

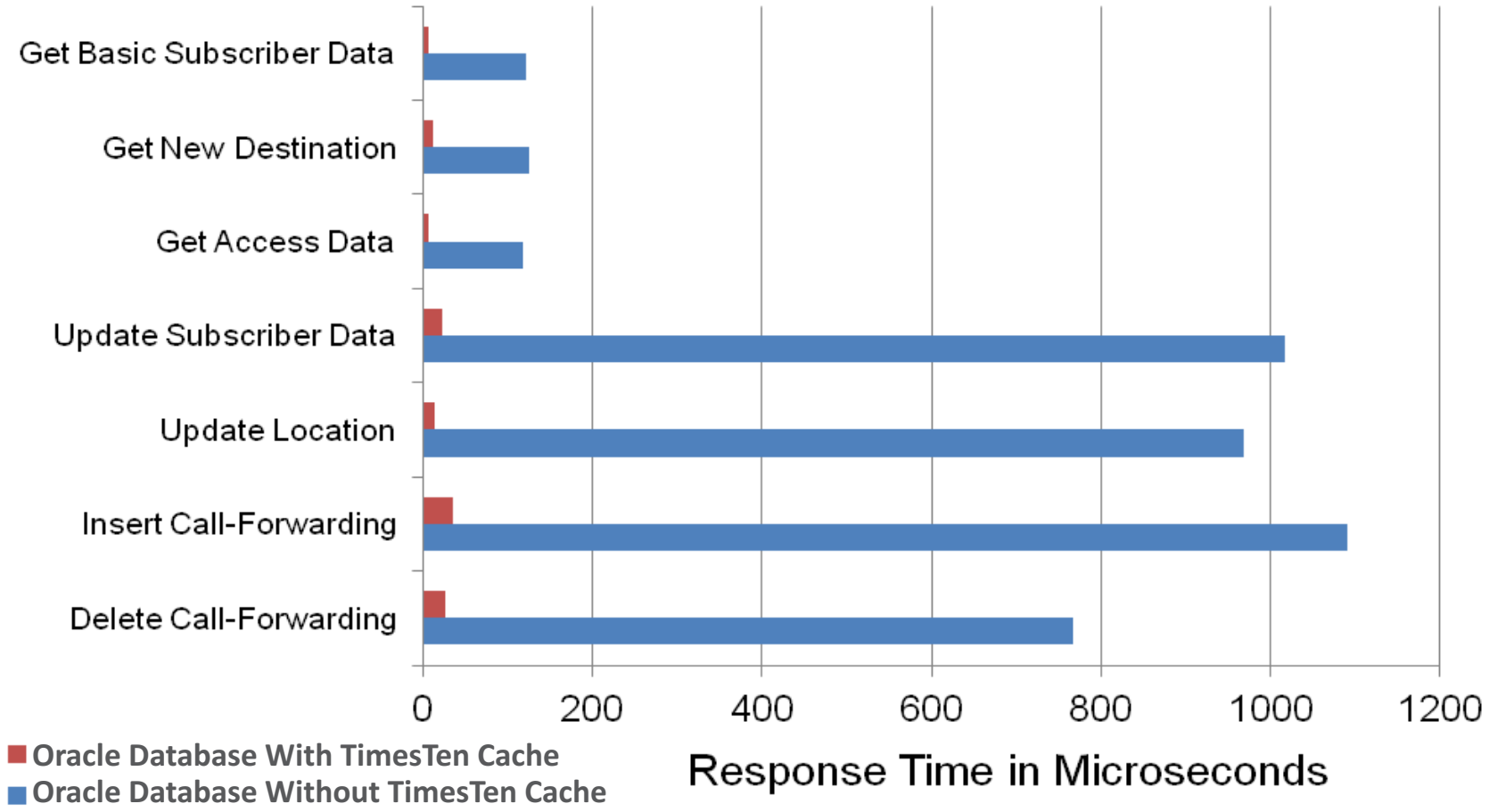
Low Latency - Microseconds Response Time

TPTBM Read and Update
E5-2699 v4 @ 2.20GHz
2 socket, 22 cores/socket,
2 threads/core
TimesTen 11.2.2.8.0
(100M rows, 17GB data)



HLR Mobile Transactions Response Time

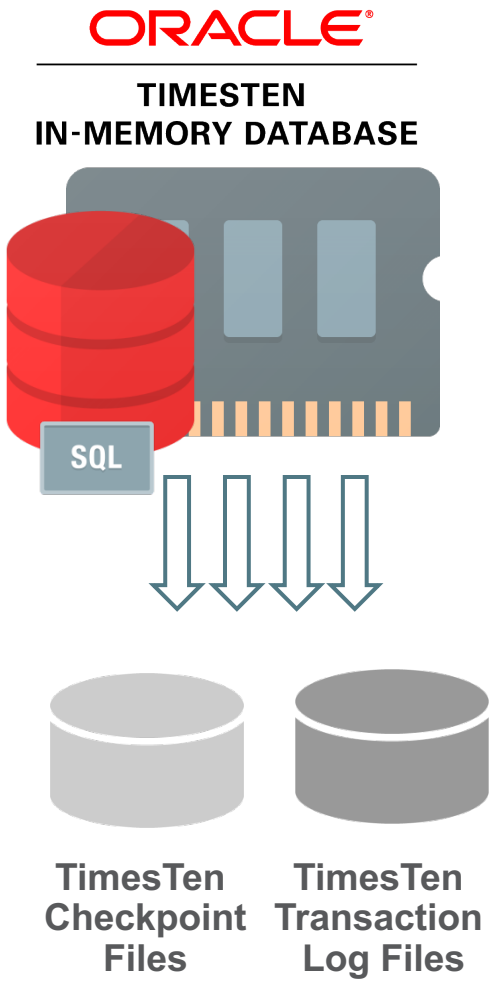
Response Time Improvement With TimesTen Application-Tier Database Cache



Speedup
17x
10x
17x
44x
69x
31x
29x

Intel® Xeon CPU E5-2680 @2.7GHZ 2 sockets 8 cores/socket 2 hyper-threads/core 32 vCPU

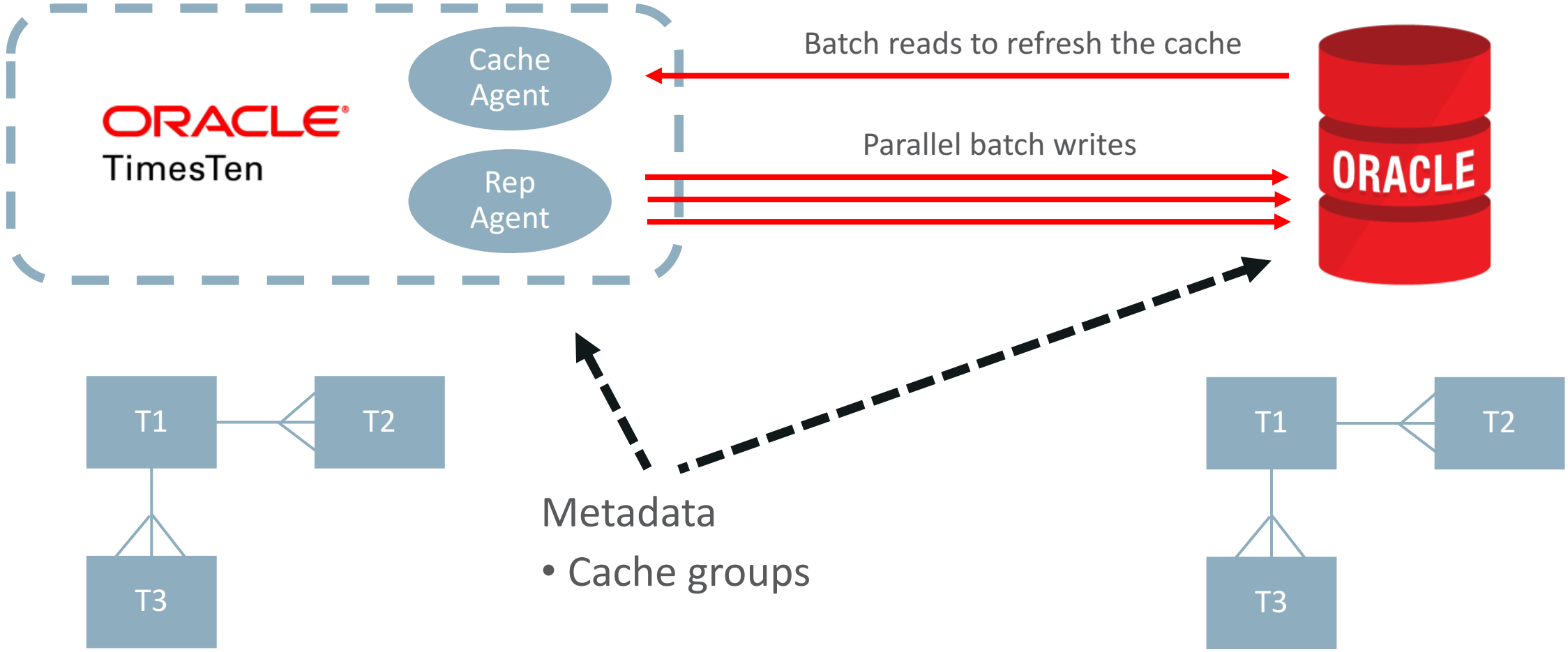
TimesTen In-Memory Database
Persistence and Recovery



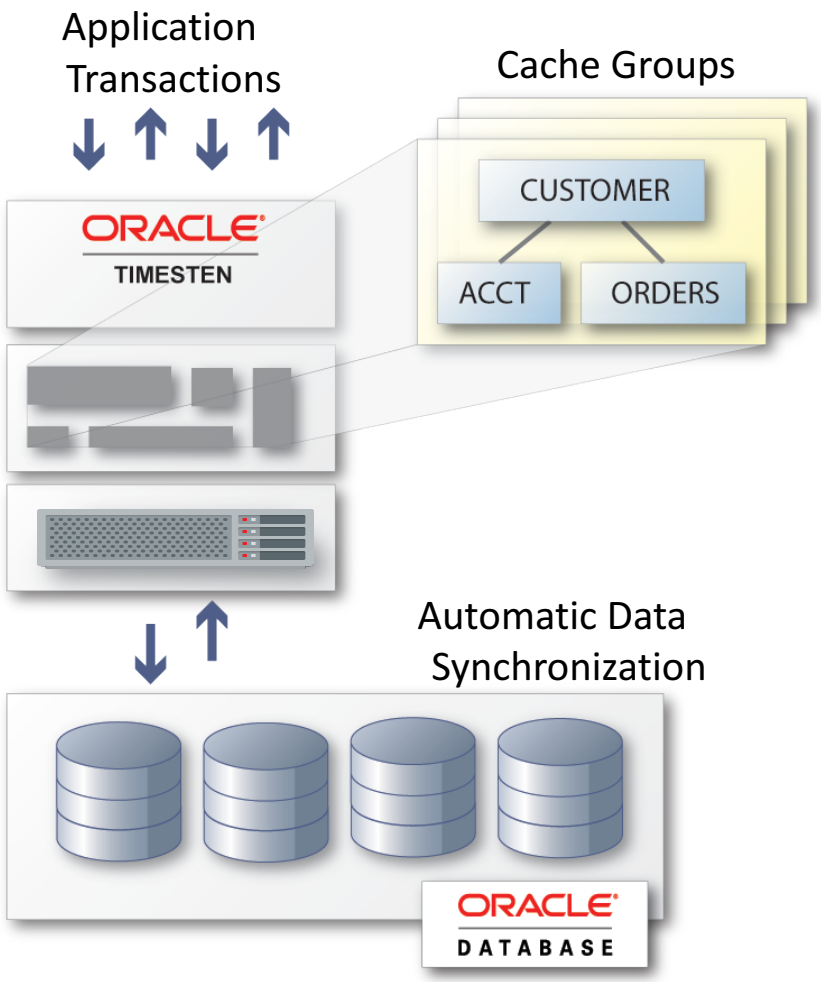
- TimesTen database persistence can be configured on
  - Flash, SSD, hard disk storage
- All transactions are logged and persisted
  - Redo, undo, and recovery
- Dual database Checkpoint files
  - Database restart
  - Database recovery after failures

Cache DB = Oracle Application Tier Database Cache

A feature of Oracle Enterprise Edition



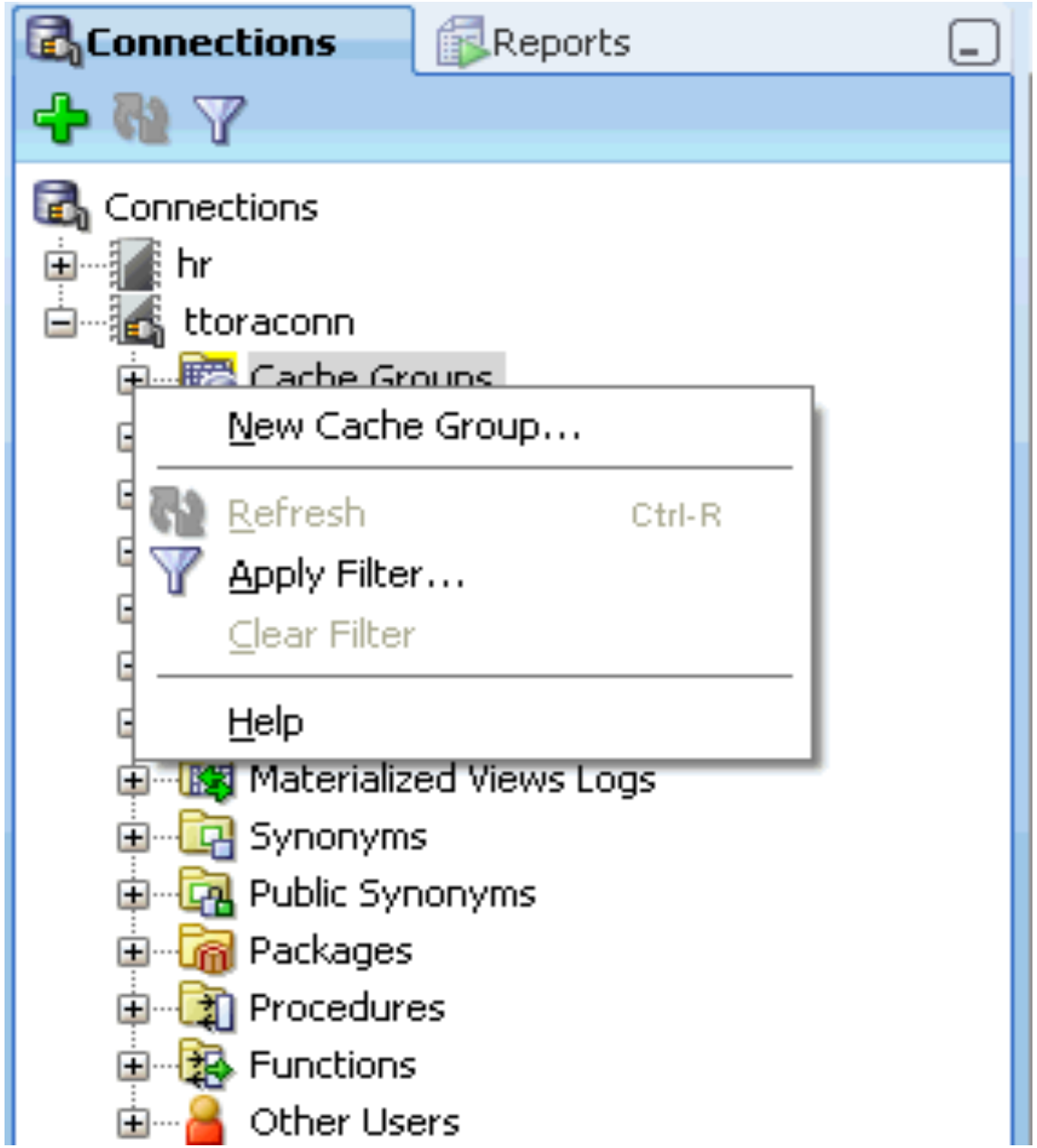
Flexible Cache Group Configurations

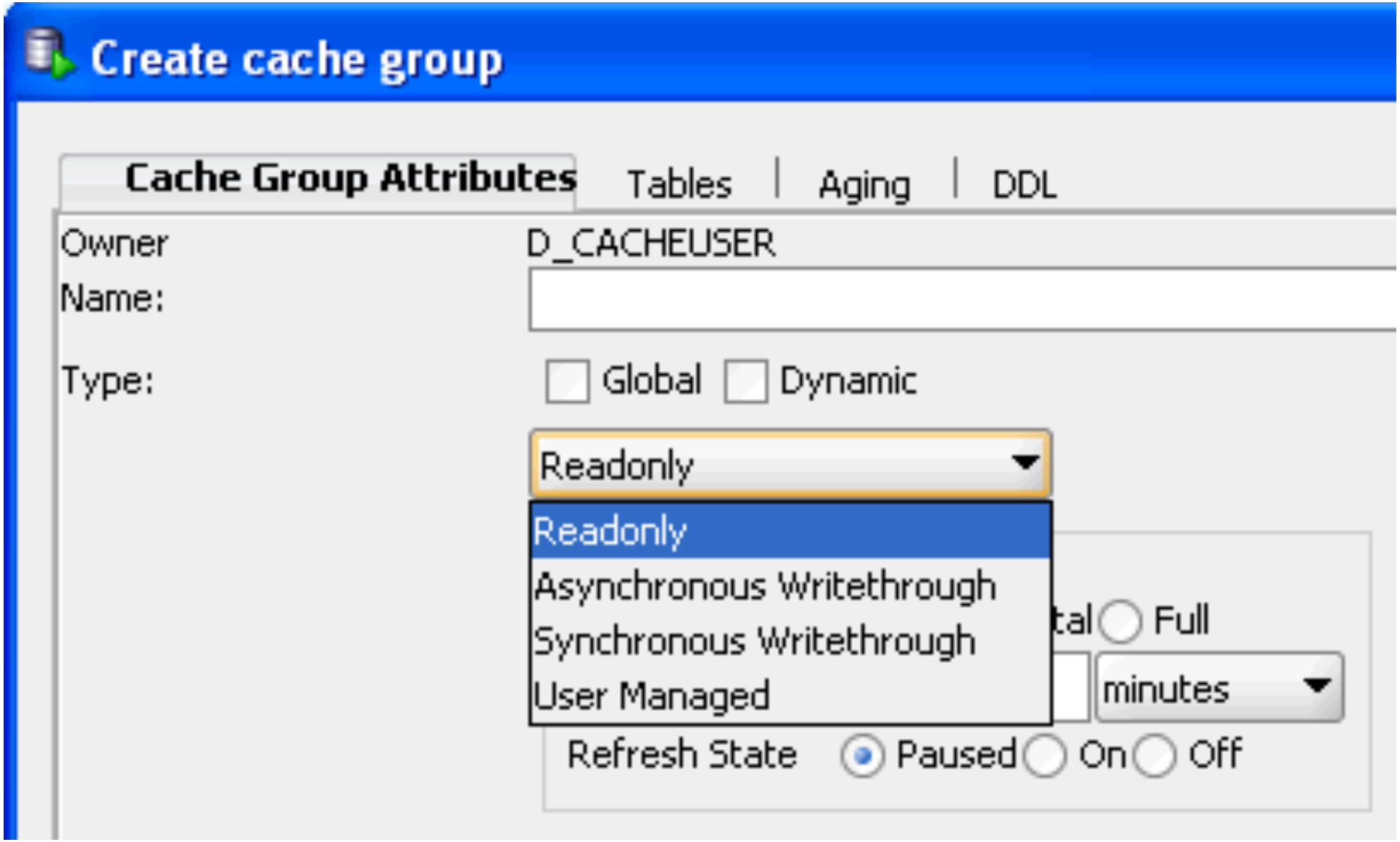


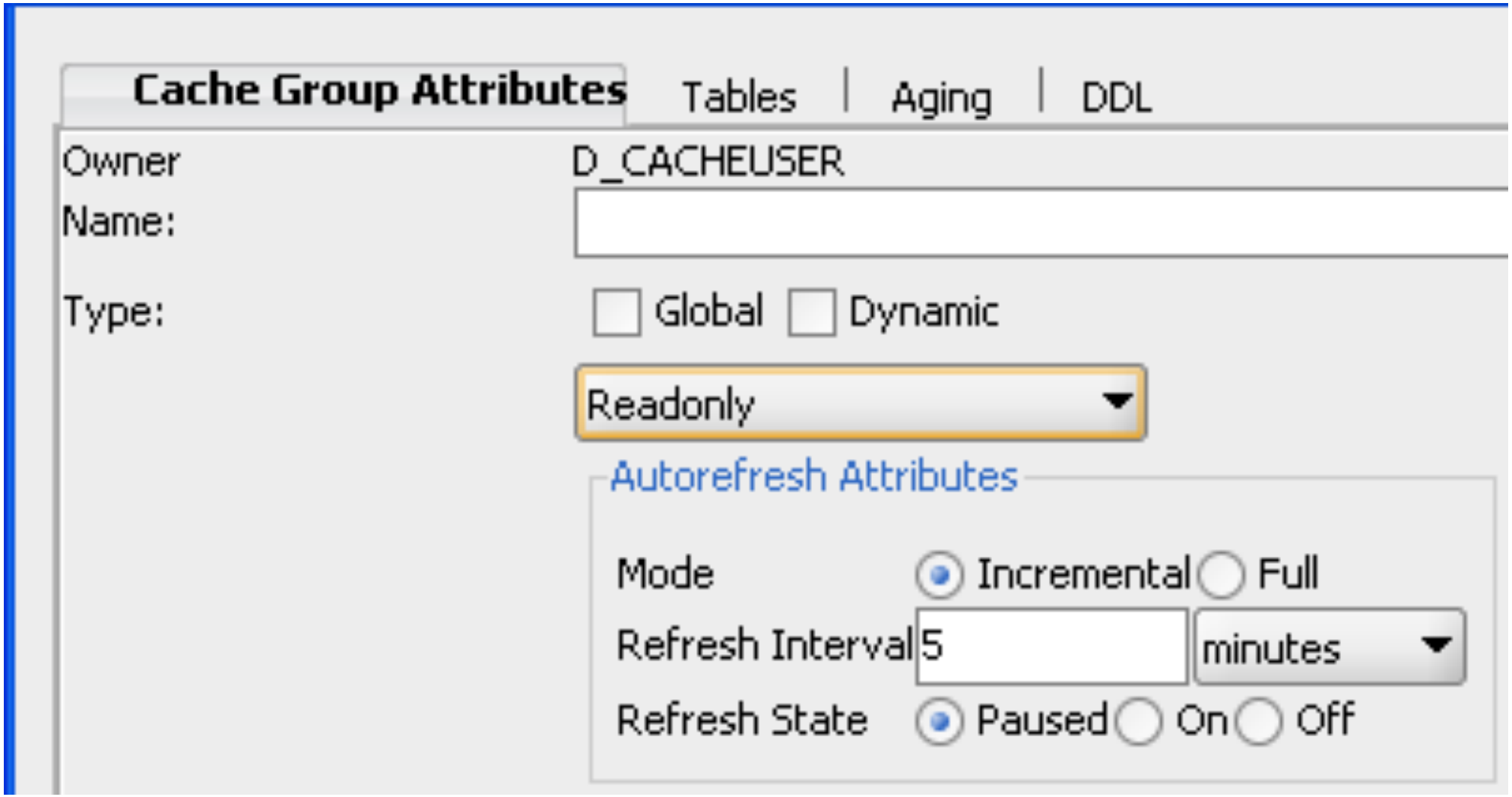
- Cache Group describes the Oracle Database tables to cache
  - All or subset of rows and columns
  - Defined using SQL

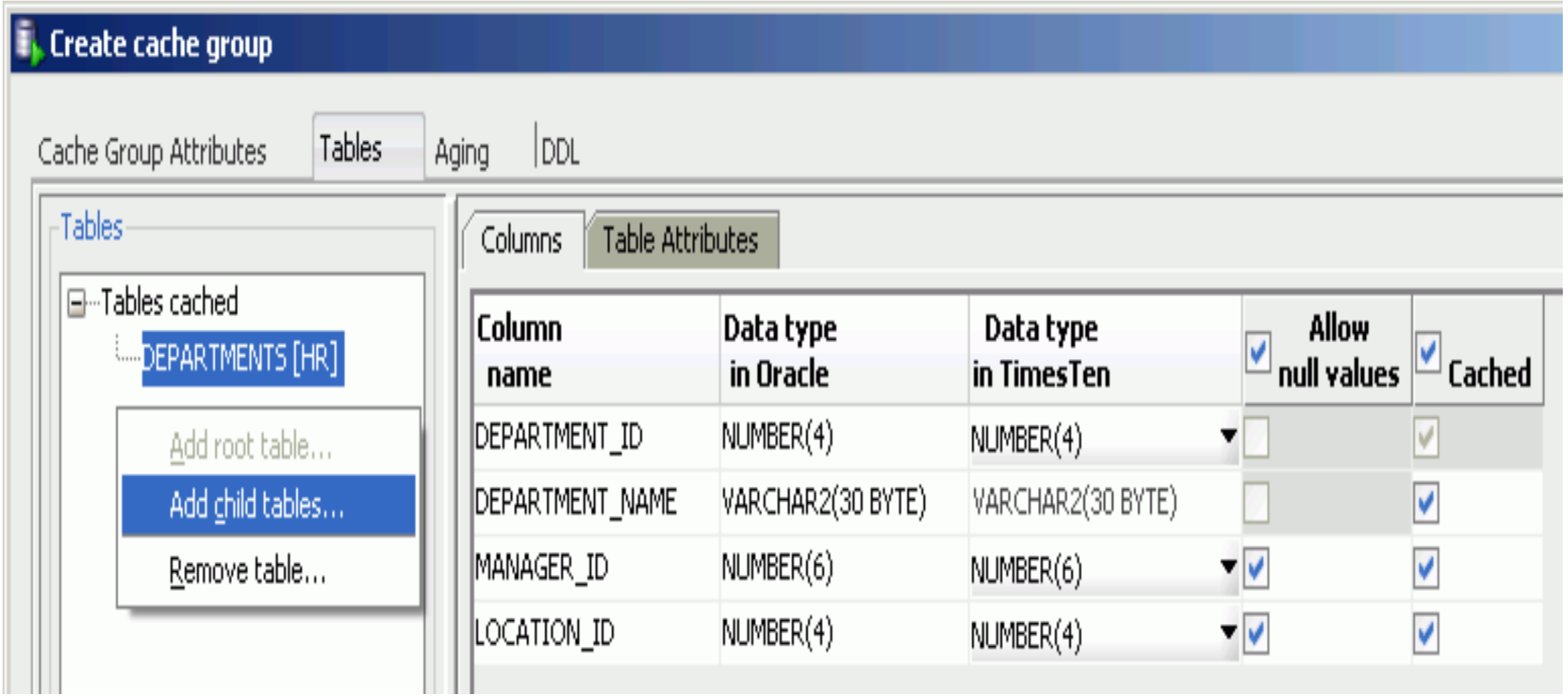
```
CREATE CACHE GROUP PremierUsers
   FROM OE.CUSTOMER (
      NAME VARCHAR2(100) NOT NULL,
      ADDR VARCHAR2(100)
 )
 WHERE OE.CUSTOMER.ORDER > 500;
```

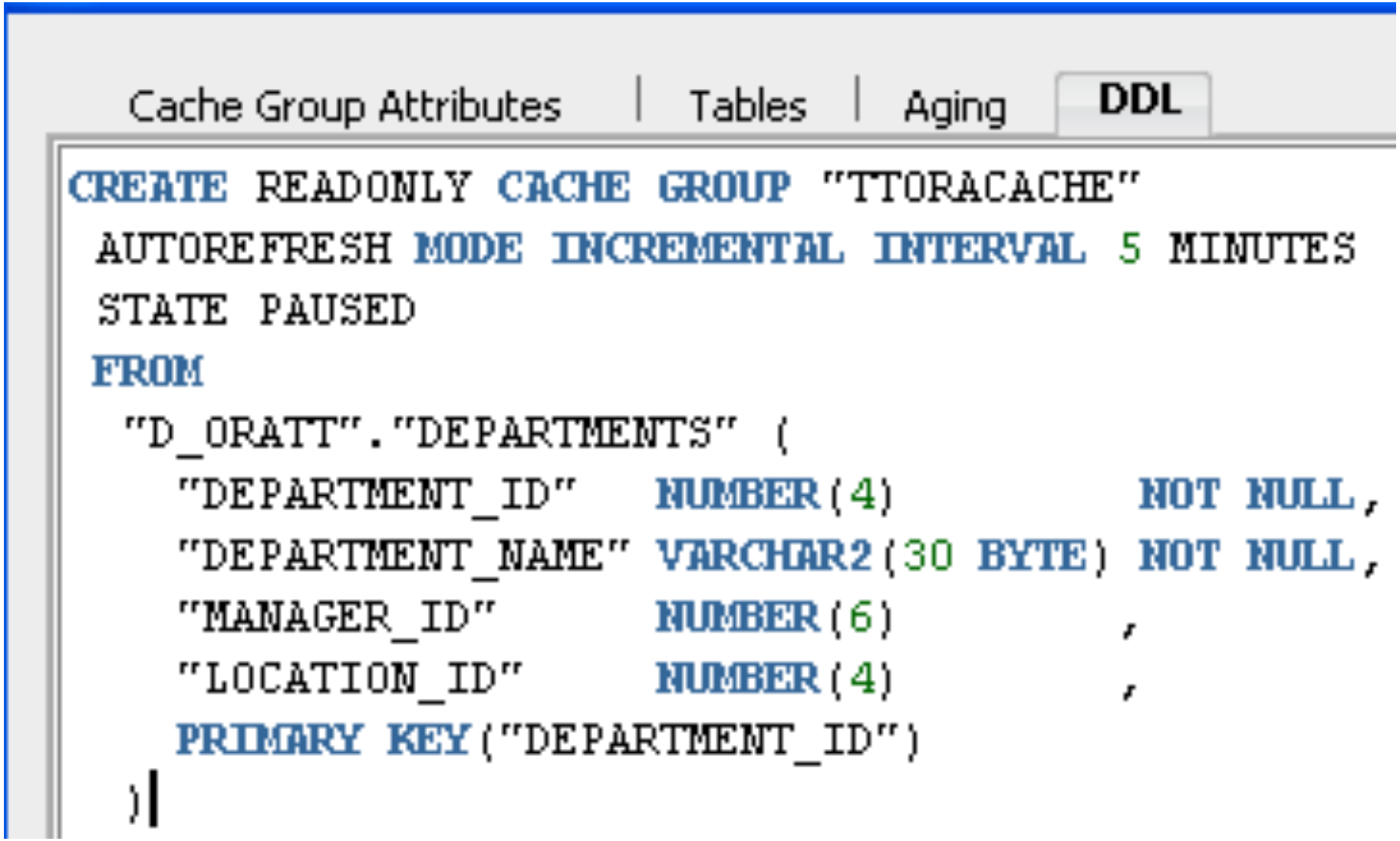
- Cache tables are regular tables in TimesTen
  - Joins/search, insert/update/delete











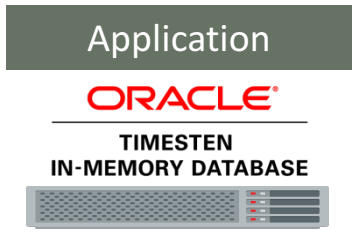
```
create dynamic read only cache group myReadCache1
autorefresh mode incremental interval  100 millseconds state on
from hr.employees ( employee_id   number (6) not null,
                    first_name              varchar2(20),
                    last_name               varchar2(25) not null,
                    hire_date                date not null,
                    job_id                   varchar2(10) not null,
                    salary                    number (8,2),
                    manager_id               number (6),
                    department_id           number(4),
                  primary key (employee_id)),
hr.job_history (employee_id number(6) not null,
                    start_date date not null,
                    end_date date not null,
                    job_id varchar2(10) not null,
                    department_id number(4),
  primary key (employee_id, start_date),
  foreign key (employee_id)
  references hr.employees (employee_id));
```

Oracle TimesTen

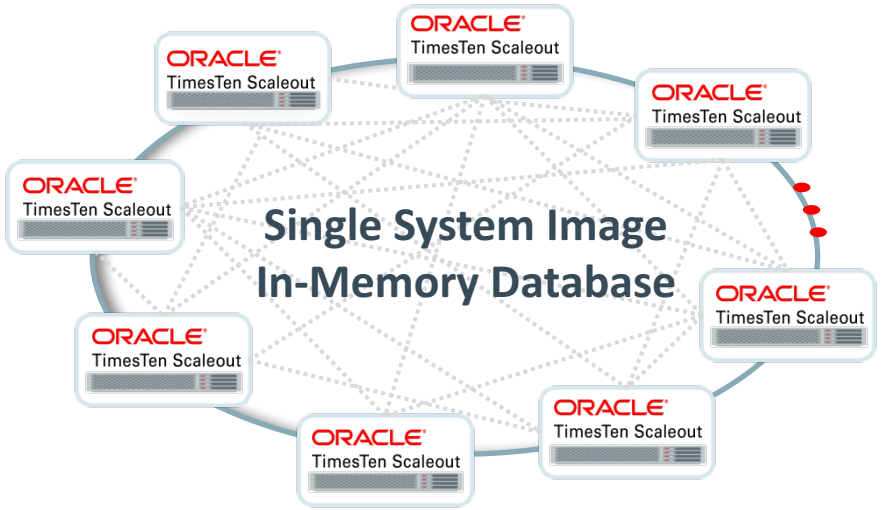
- One technology, two products, three deployment modes

Oracle TimesTen In-Memory Database

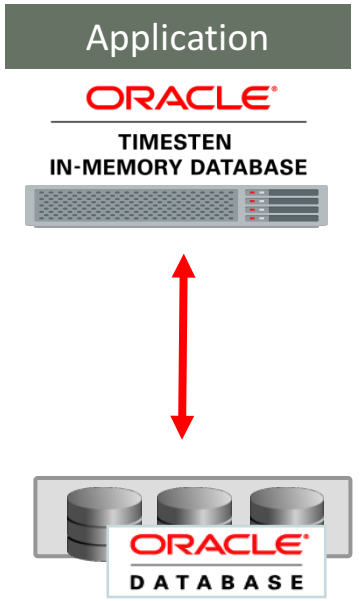
Standalone In-Memory Database for OLTP and analytics



TimesTen Scaleout



Application Tier Database Cache



Summary

- Cloud Scale read/write caching with 1ms latency at 99th percentile
- Oracle TimesTen is faster than any NoSQL DB
- Caching is configuration, not coding
- Use TimesTen PLSQL stored procedures to minimize network hops

- Write TimesTen apps as if it were an Oracle DB*
  - SQL, JDBC, PLSQL, OCI, ODBC, ODP.Net, Pro*C, Pro*COBOL
  - R, Python, Node.js, Go, Ruby & PHP

Subset of the Oracle SQL and PLSQL