Cache is Not Always King
A Tale of Two Caches

Tirthankar Lahiri
Vice President
Data and In-Memory Technologies
Oracle Database

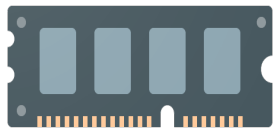
ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Adding External Caches on top of Databases

- **Adding a Cache to your Database Architecture has many benefits**
- **As advertised …**

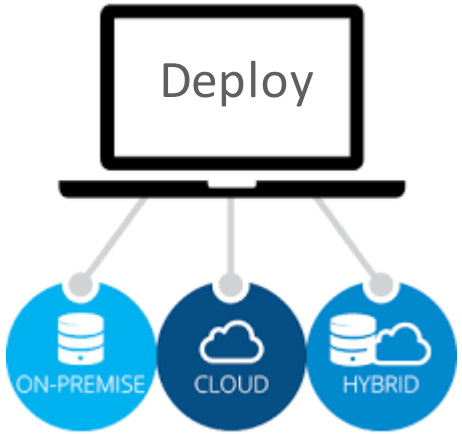


In-Memory

HIGH PERFORMANCE

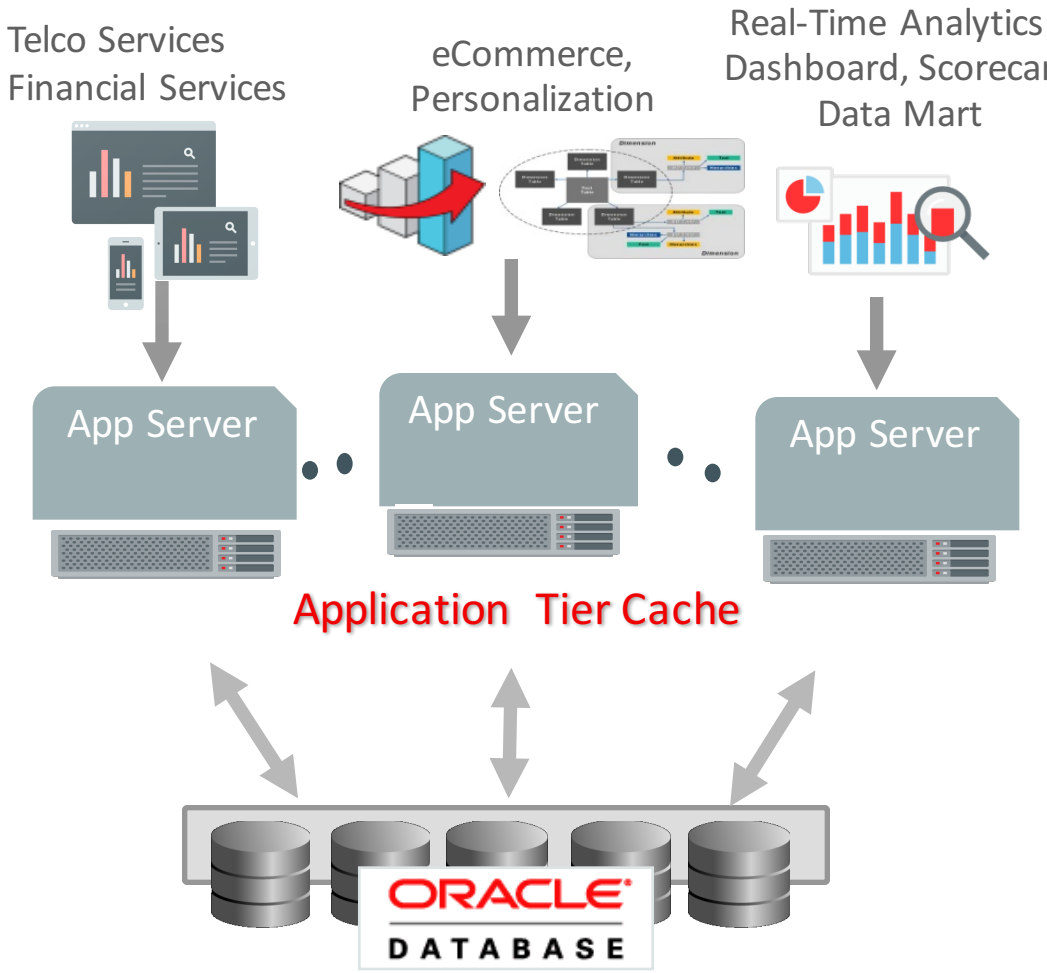
Ultra Low Latency

Transactions



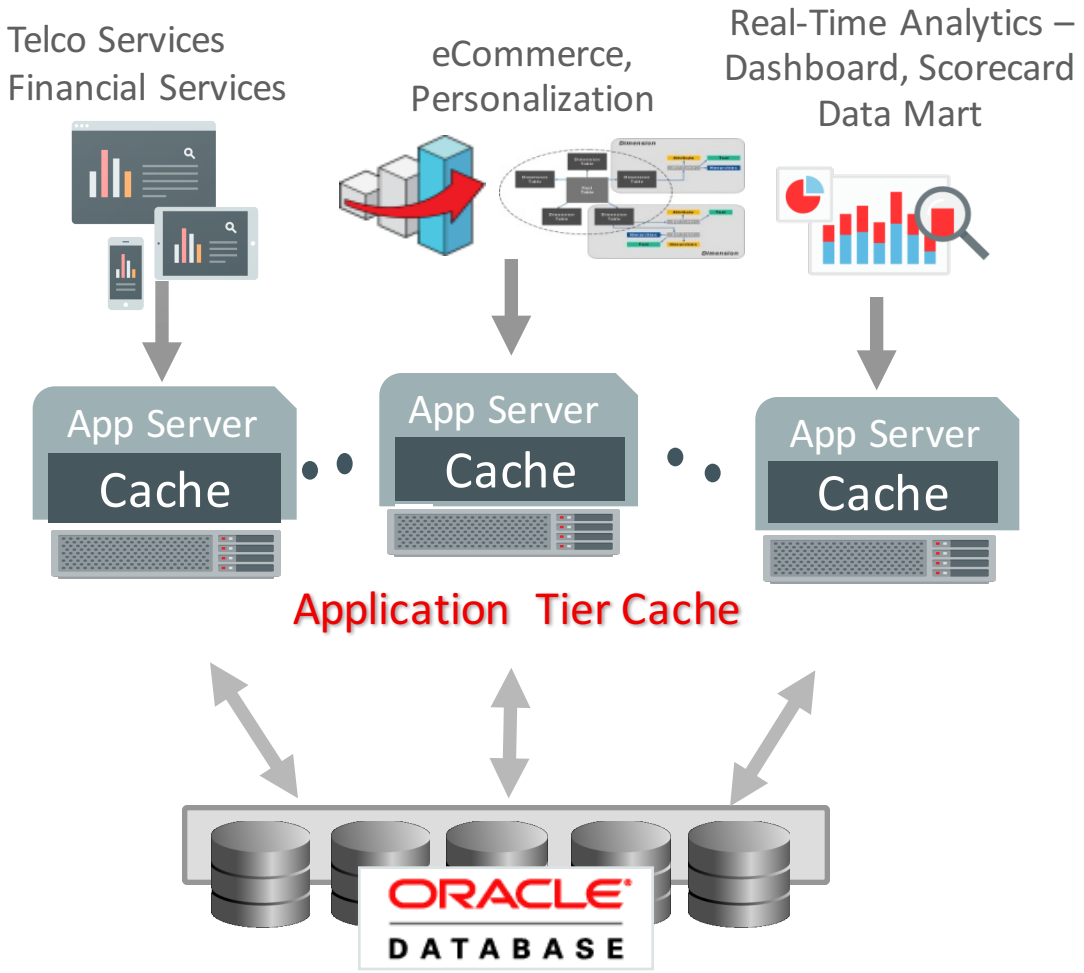
- **Most important: Cache => Cash**

Application-Tier Caches for Enterprise Databases



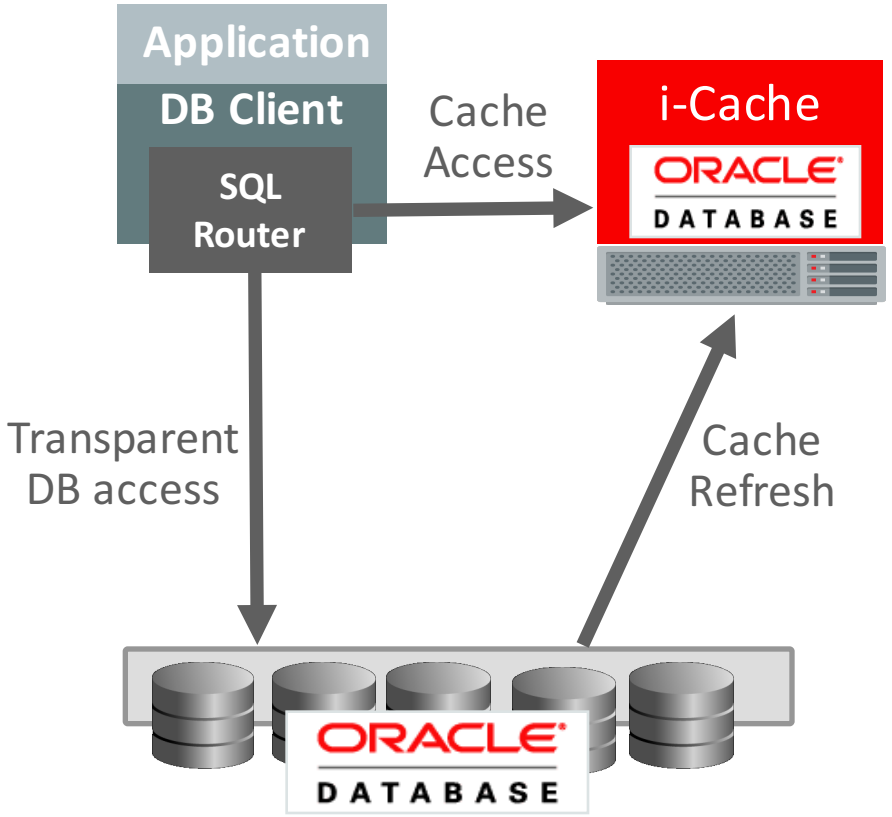
- Reduce response time
- Increase throughput
- Increase application availability
- Reduce load on database
- Allow ad-hoc browsing on cached data
  - E.g. if Tweet table is cached, find all the tweets by @realDonaldTrump tweeted during normal hours

Application-Tier Caches for Enterprise Databases



- Reduce response time
- Increase throughput
- Increase application availability
- Reduce load on database
- Allow ad-hoc browsing on cached data
  - E.g. if Tweet table is cached, find all the tweets by @realDonaldTrump tweeted during normal work hours

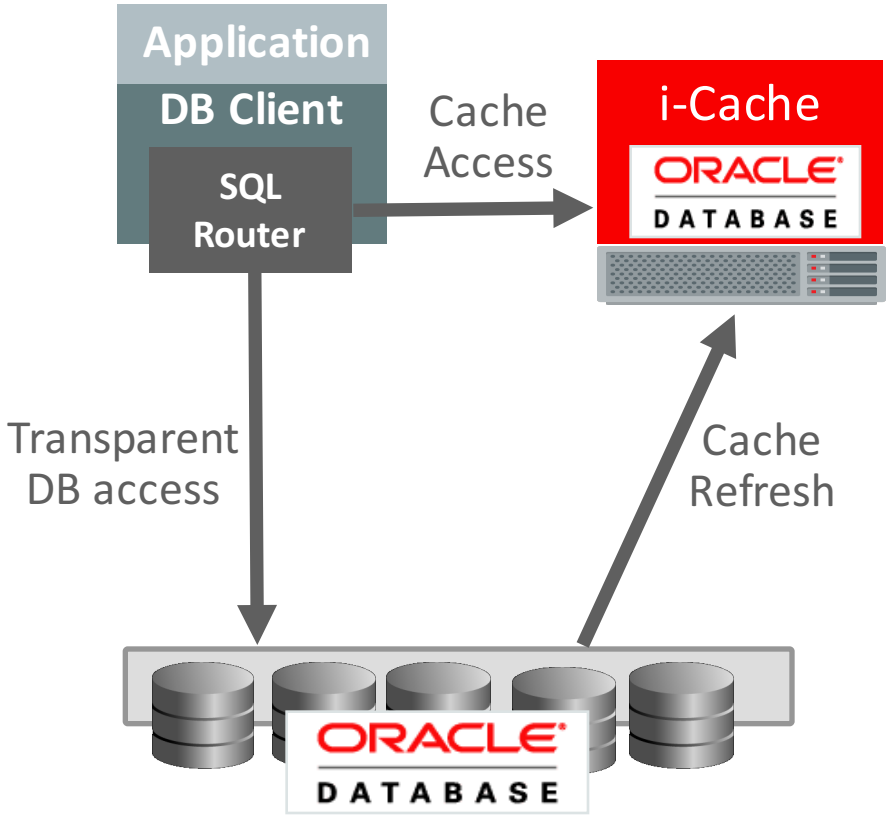
FAD #1 Using Oracle Database as a Cache: i-Cache (Oracle 8i)



- **Basic Idea:** Cache selected tables from Oracle Database in a smaller application-tier Oracle database

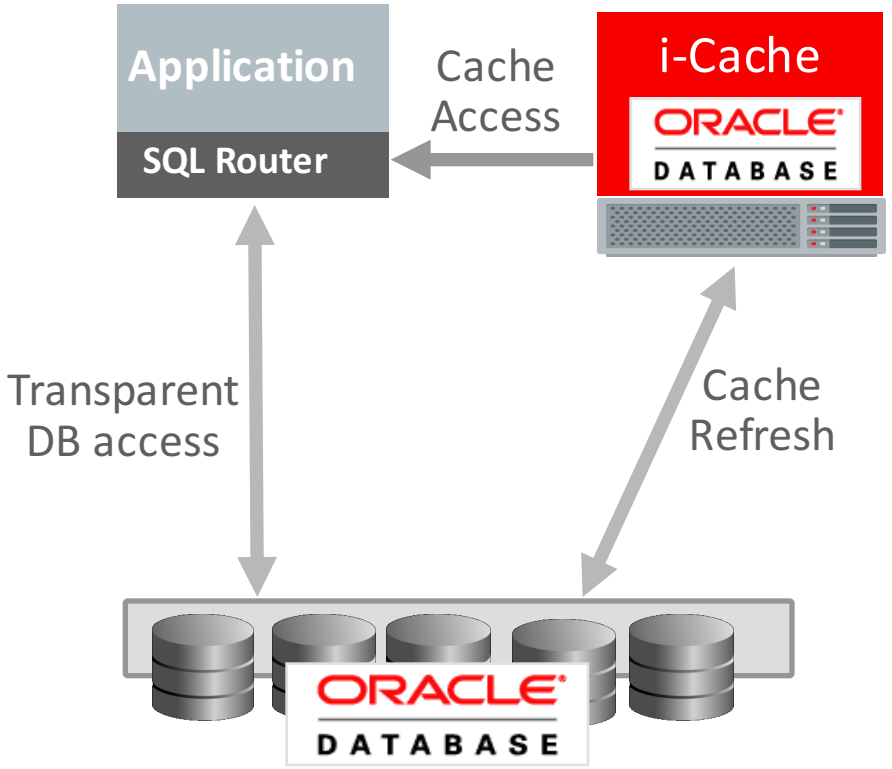
- Cache contents are Read-Only
  - Queries on cache tables transparently redirected to cache
  - DMLs directed automatically to Database
  - Cache periodically refreshed
  - Application must be willing to tolerate stale data

i-Cache Project: Architecture Benefits



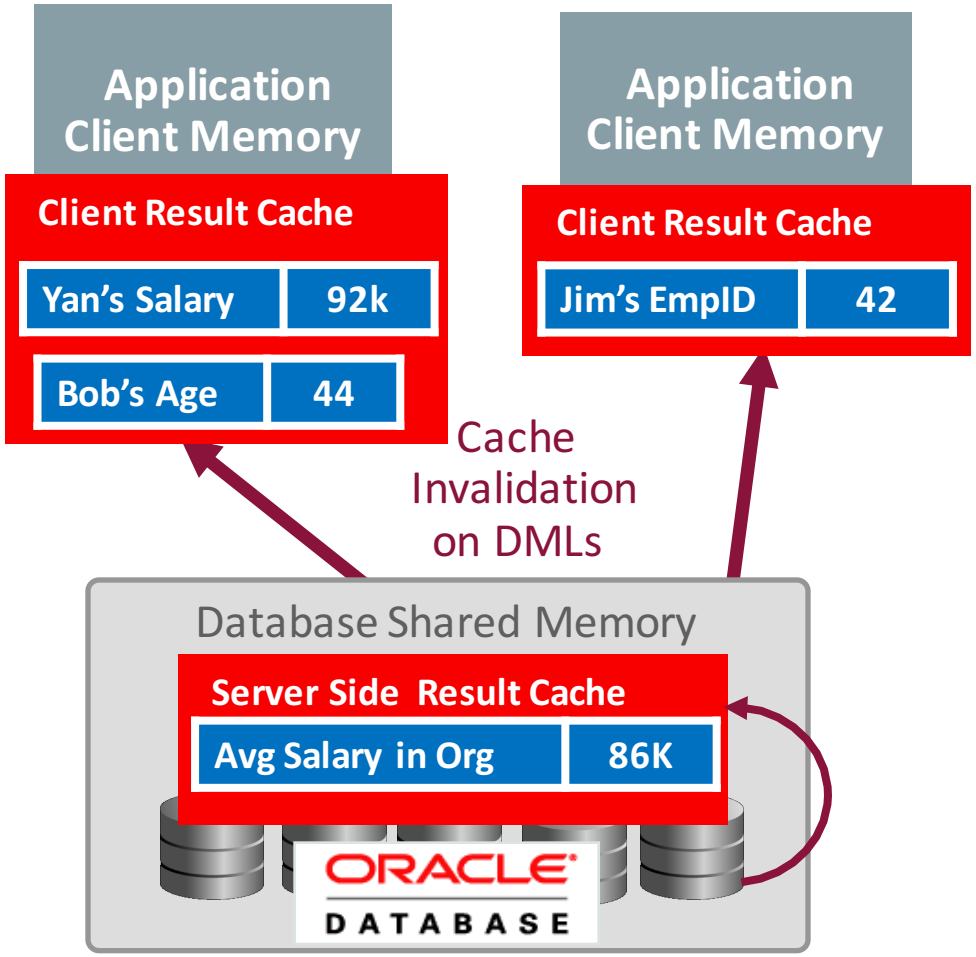
- Cache is 100% compatible with Database
- Full power of Oracle Database for sophisticated cache browsing / subset operations
  - E.g if Tweet table is cached, find most retweeted tweets of @realDonaldTrump without involving Database
- Performance benefits: Can colocate cache DB with application running on same host

i-Cache Project: Lessons



- Drawbacks of the approach:
  - Footprint: IO requirements, memory footprint, installation size, Management cost, etc.
  - Some application changes required (tolerate staleness), cannot use cache for queries within a transaction
  - Overall cost esp with standby database cache
- Verdict:

Spiritual Successors to i-Cache: Result Caches



Simple and transparent query result caches

- Fully consistent and completely transparent
- Invalidated on table DML via Change Notification

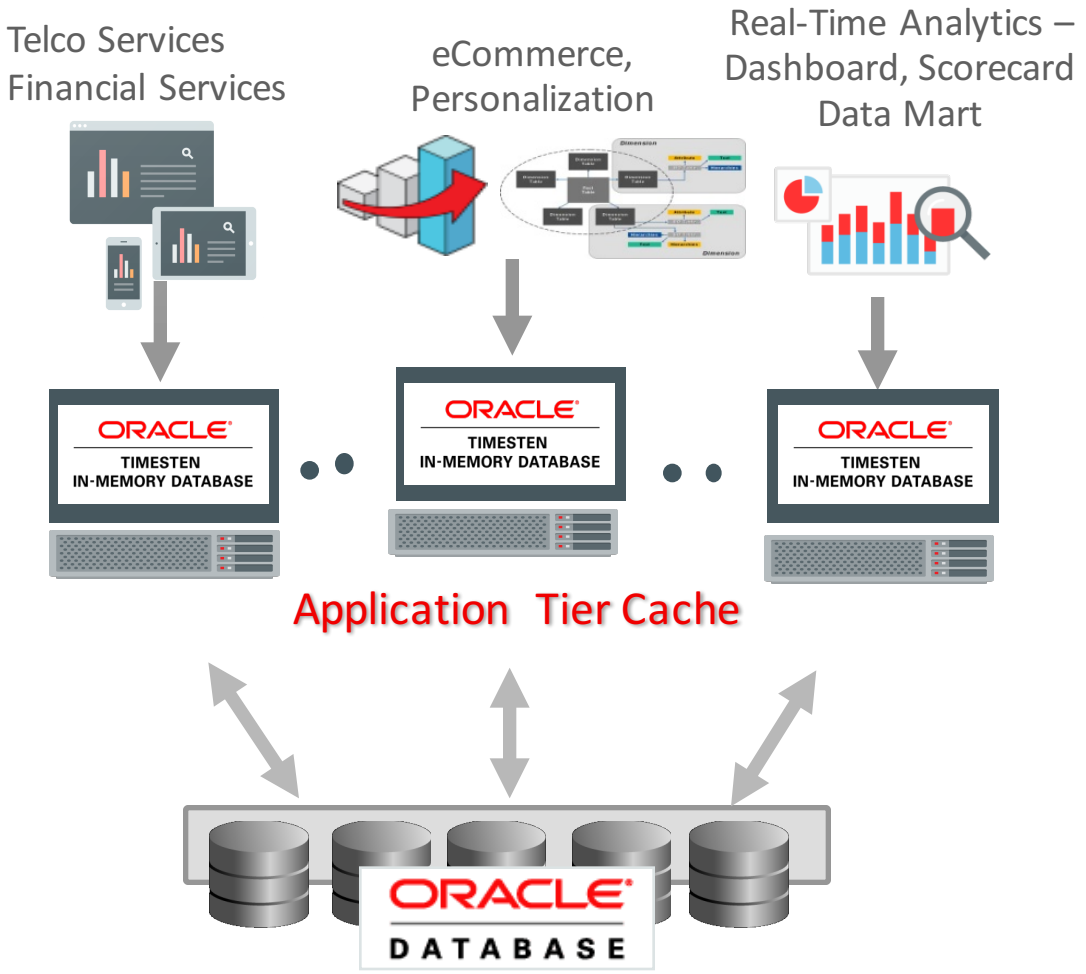
1. Client-Side Result Cache
   - Client cache of query results for repeated queries
   - Avoids DB access for repeated access to read-mostly tables
2. Server-Side Result Cache
   - Cache of results of queries and sub-queries
   - Reduces processing when many rows accessed to generate small result (e.g. max, average, group by)
   - Different queries benefit if they share same sub-query (e.g. result of an aggregation or group by)

ORACLE

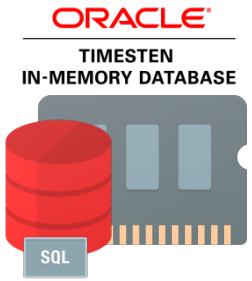
Back to Using a Database to Cache from a Database

- Result caches are widely used …
  - Result caches are simple and transparent
  - But lack query / browse capability
  - No Application HA benefits either

- Again: Back to using a database …
  - What about a lightweight, easily embeddable, and superfast database?
  - TimesTen had built a big business around caching
  - Acquired by Oracle in 2005

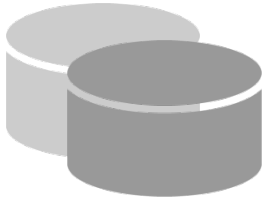


Oracle TimesTen In-Memory Database



Relational Database

- Pure in-memory
- ACID compliant
- Standard SQL
- Oracle Compatible SQL, types

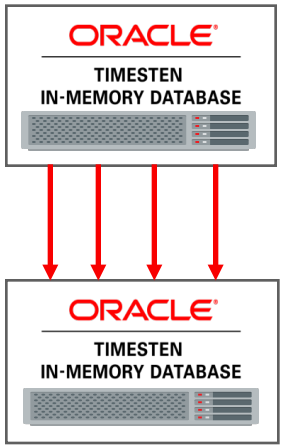


Persistent and Recoverable

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

Extremely Fast

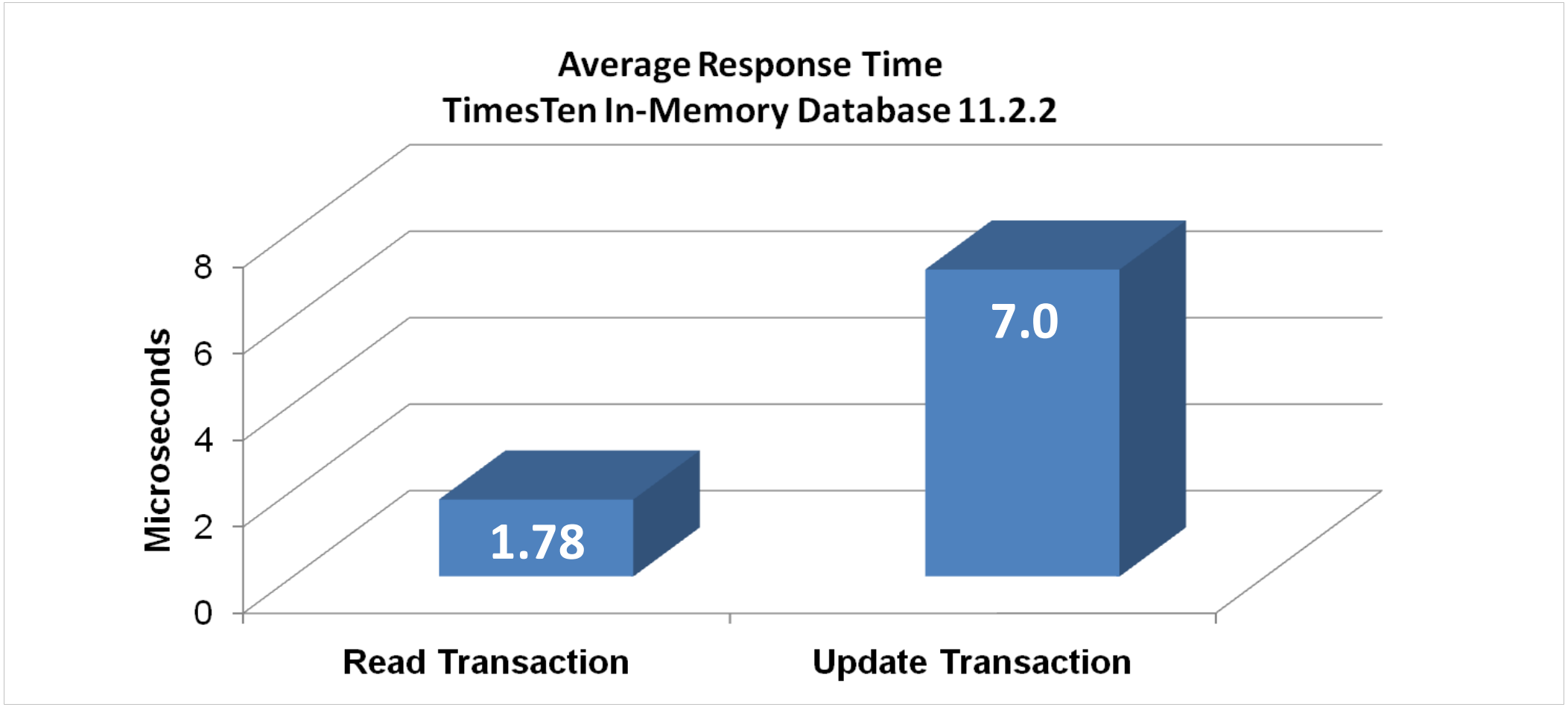
- Microseconds response time
- Very high throughput



Highly Available

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

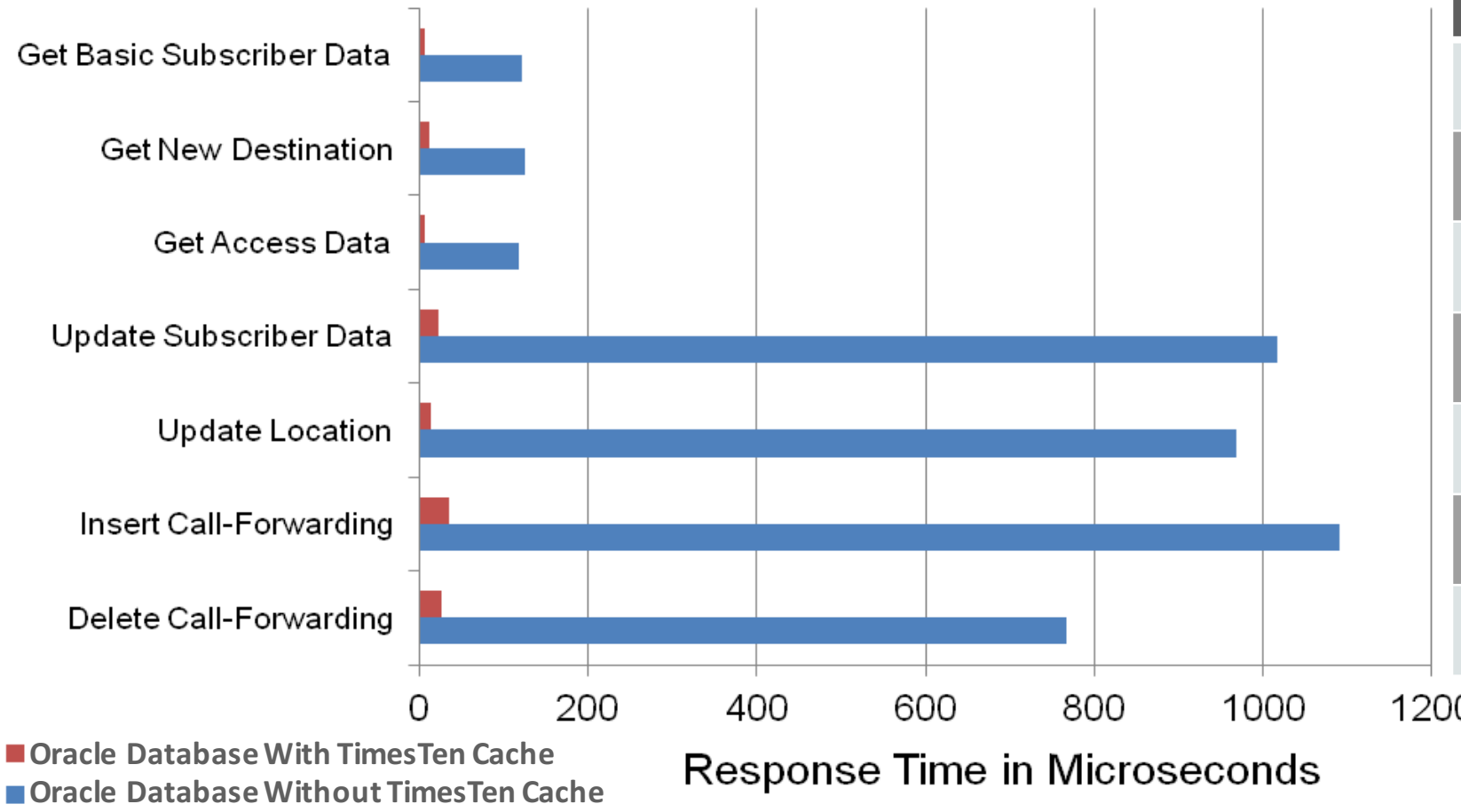
Microsecond Response Time



Oracle TimesTen In-Memory Database 11.2.2.0 - Intel Xeon 5670 2.93Ghz, 2 CPUs, 6 cores/CPU - Oracle Linux 5.6

HLR Mobile Transactions Response Time

Response Time Improvement With TimesTen Application-Tier Database Cache

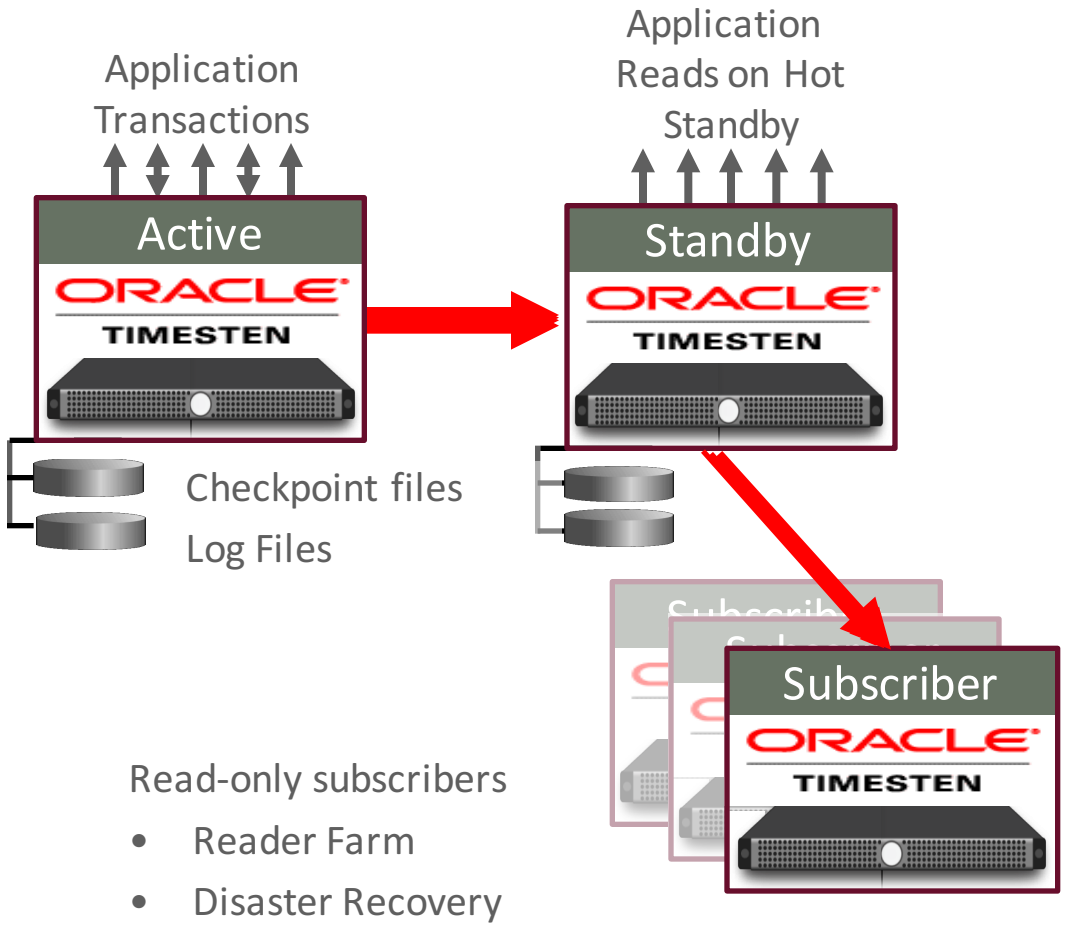


Speedup
17x
10x
17x
44x
69x
31x
29x

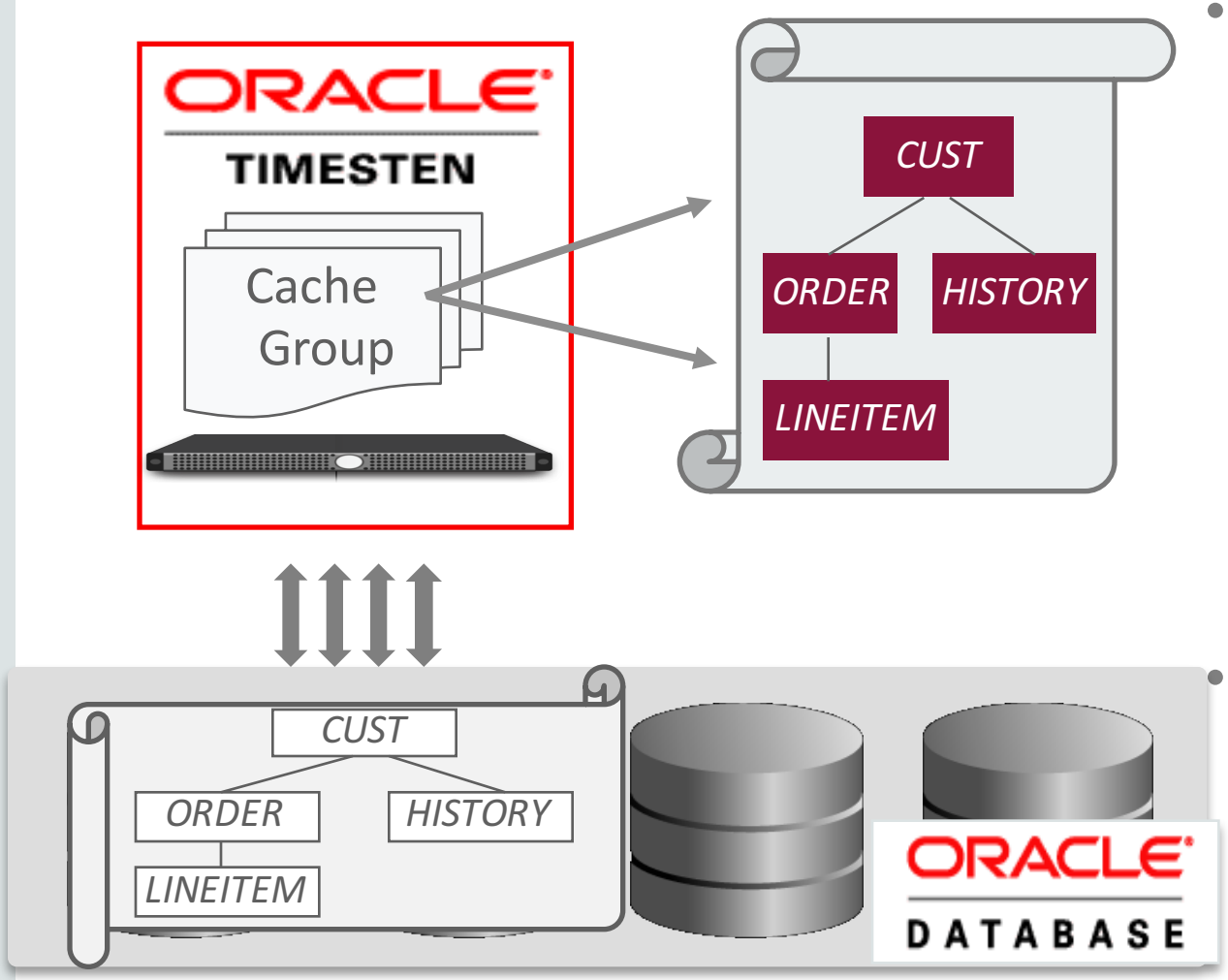
Intel® Xeon CPU E5-2680 @2.7GHZ 2 sockets 8 cores/socket 2 hyper-threads/core 32 vCPU

TimesTen is Designed for High Availability

- Fully persistent
  - Checkpoints, logging, txns
- Real-time transactional replication
  - Multiple topologies
    - Active, Standby, Reader Farm scale-out
  - Online upgrades
- Oracle Clusterware integration
  - Automatic database switchover
  - Automatic application failover



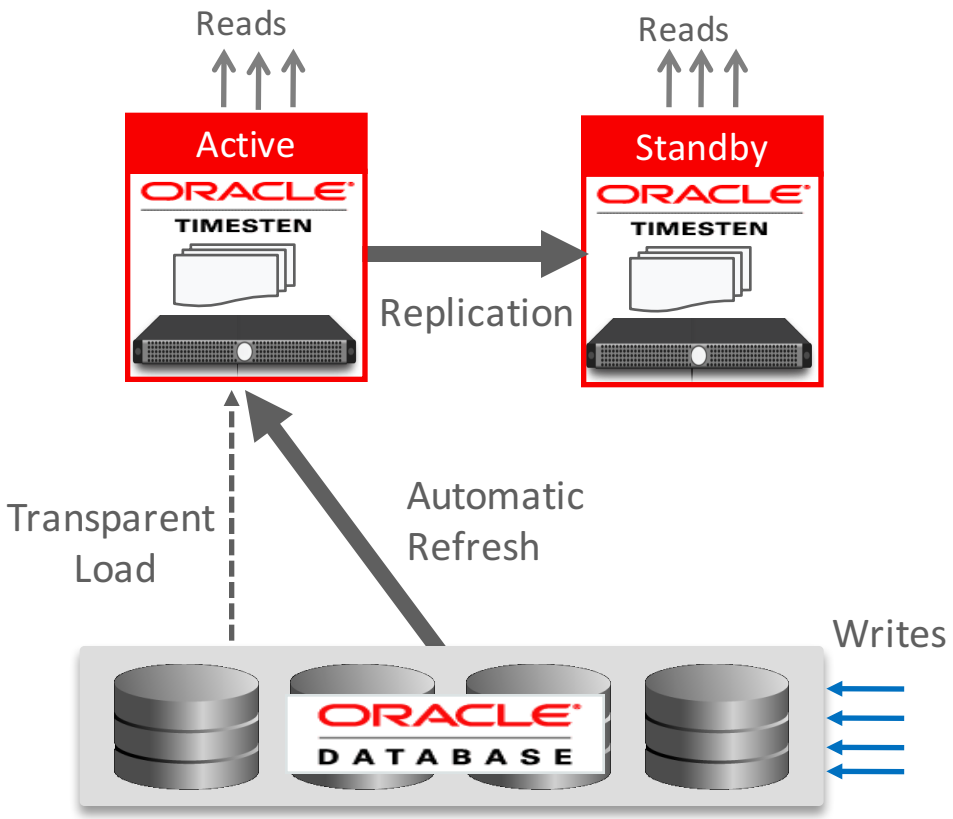
Caching Data using TimesTen



- Cache Group: Declaration of tables to cache
  - Hierarchy of Oracle database tables
  - Always has a Root table
  - Optional child tables (FK/PK)
  - Optionally subset by rows, columns
  - Related rows are called *Cache Instance*
    - *Customer row plus related orders, orderlines, history rows*
- Cached tables are native TimesTen tables
  - Standard database APIs
  - Transactional, persistent
  - Automatic synchronization with Oracle

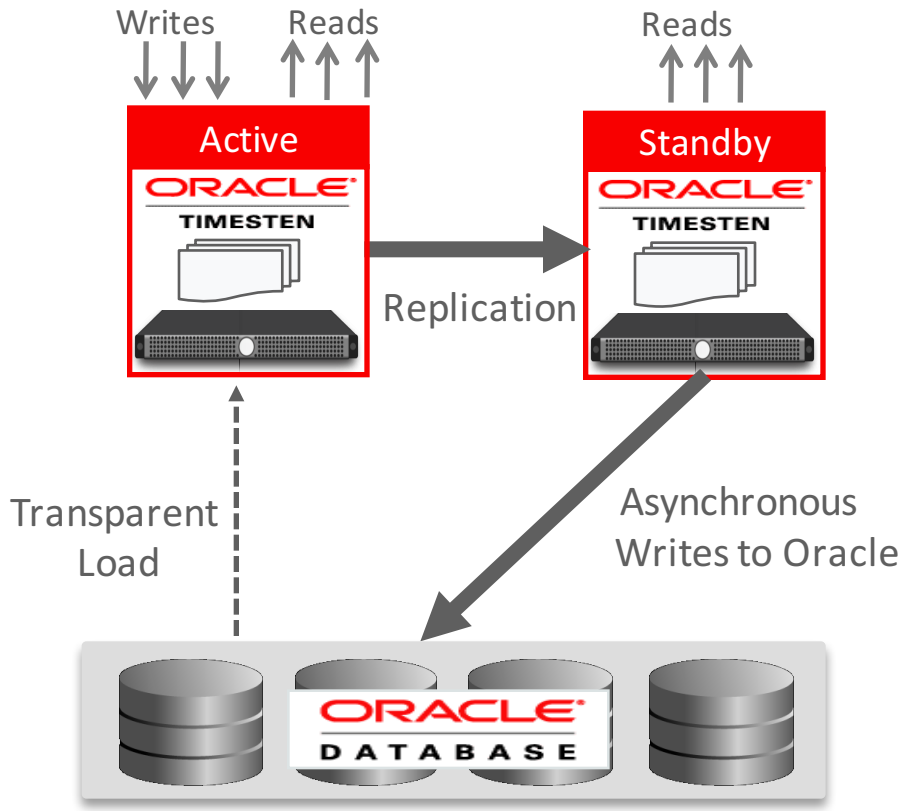
Read-Only Cache Groups

- For hot reference data: catalogs, price lists, etc.
- Updates occur on Oracle and tracked in change log tables (via triggers)
- Periodic refresh of changes, configurable interval
- Tables are pre-loaded by default
- Optional dynamic loading for queries by primary key for very large tables
  - Transparent Load: Insert cache instance on cache miss
  - LRU aging of older cache instances to limit space



Writeable TimesTen Cache Groups

- Useful for update intensive data like sensor data, cellular subscriber account information
- Changes propagated asynchronously to Oracle
- Static Loading: Tables are pre-loaded
- Dynamic Loading: For very large tables and queries by primary key
  - Transparent Load: Insert cache instance on cache miss
  - LRU aging of older cache instances to limit space



TimesTen Cache Adoption

Deployed by Thousands of Companies

ORACLE

Example of Cache Deployment

中国平安 PINGAN

Application Overview

- Industry : Finance
- Business : Insurance, Banking, Investment
- Application : Agent Task Management
  - Automatic task assignment based on predefined rules
  - Manually reassign task from one agent to another

Challenges

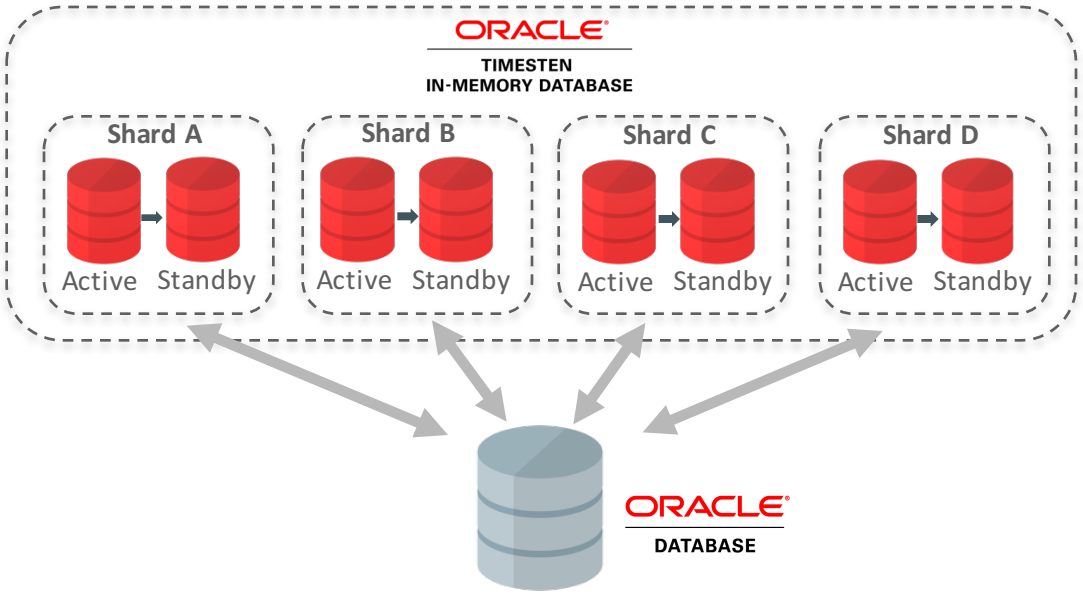
- Database scalability with extreme high concurrency affecting end to end response time
- Maintain user satisfaction
- Minimal changes to existing architecture and application
- Must be highly available

Solution

- Oracle TimesTen Application-Tier Database Cache
- TimesTen Replication for High Availability
- Oracle Database

Why TimesTen ?

- Delivered lower and consistent response time; **achieved 40x improvement in both response time and throughput**
- Automatic data synchronization between TimesTen and Oracle Database
- With built-in HA, supports automatic failover and switchover



ORACLE

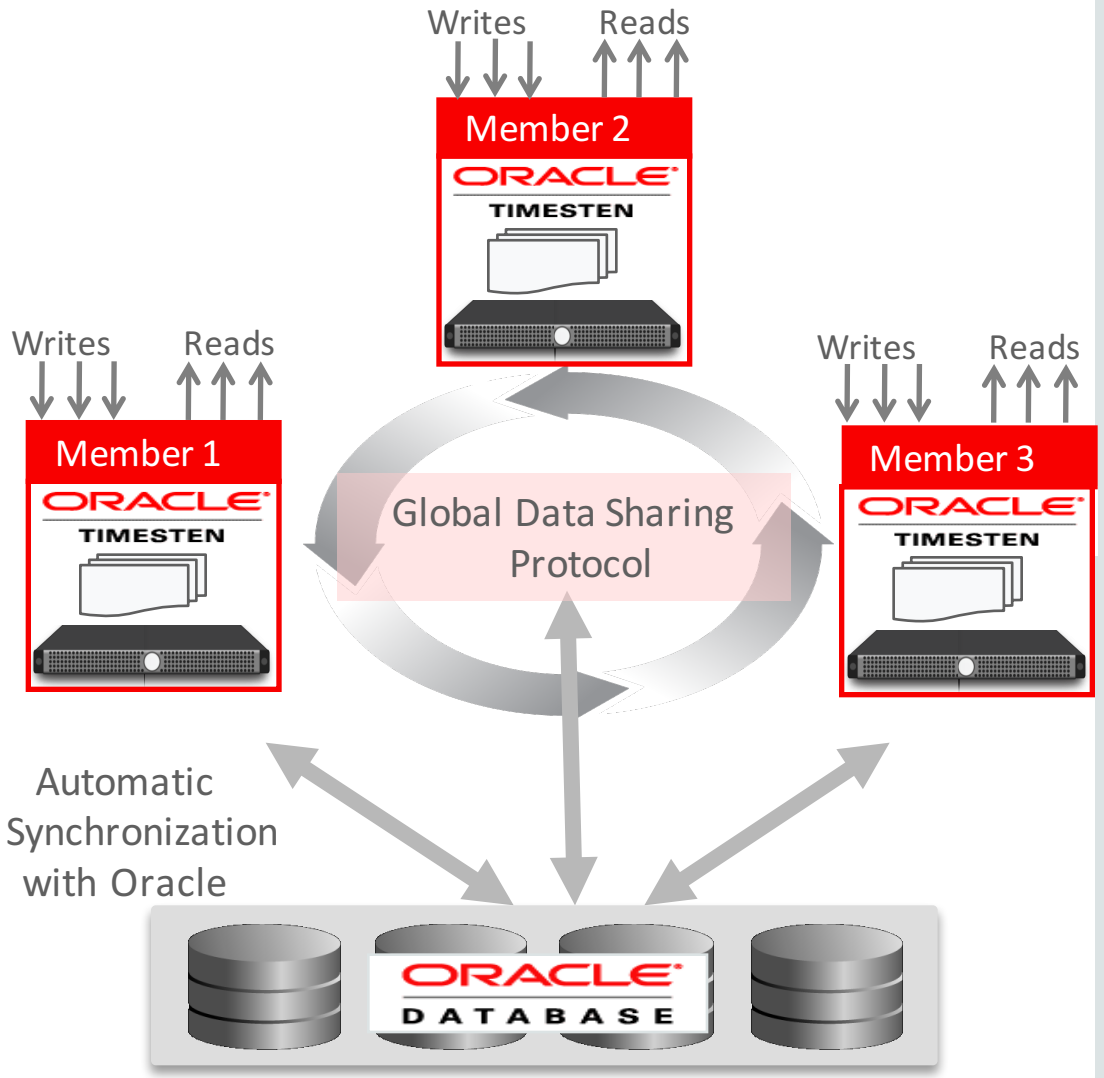
TimesTen Cache Post-Mortem

Not a FAD, an M-SAD (Mostly-Successful Aspiration in Databases)

- Reasons for adoption
  - TimesTen provides very high performance
  - Sophisticated SQL on cached data
  - Cached data survives Oracle DB restart, failure of Active or Standby node
- Some factors that limit adoption:
  - Oracle Database, especially on Exadata, is often fast enough
  - Not transparent: Application explicitly connects to the TimesTen cache
  - Subset of Oracle functionality - application re-coding maybe needed
    - E.g. CONNECT BY queries or Full outer join, SQL calling PLSQL functions
  - Does not scale-out, explicit sharding required (like PingAn)

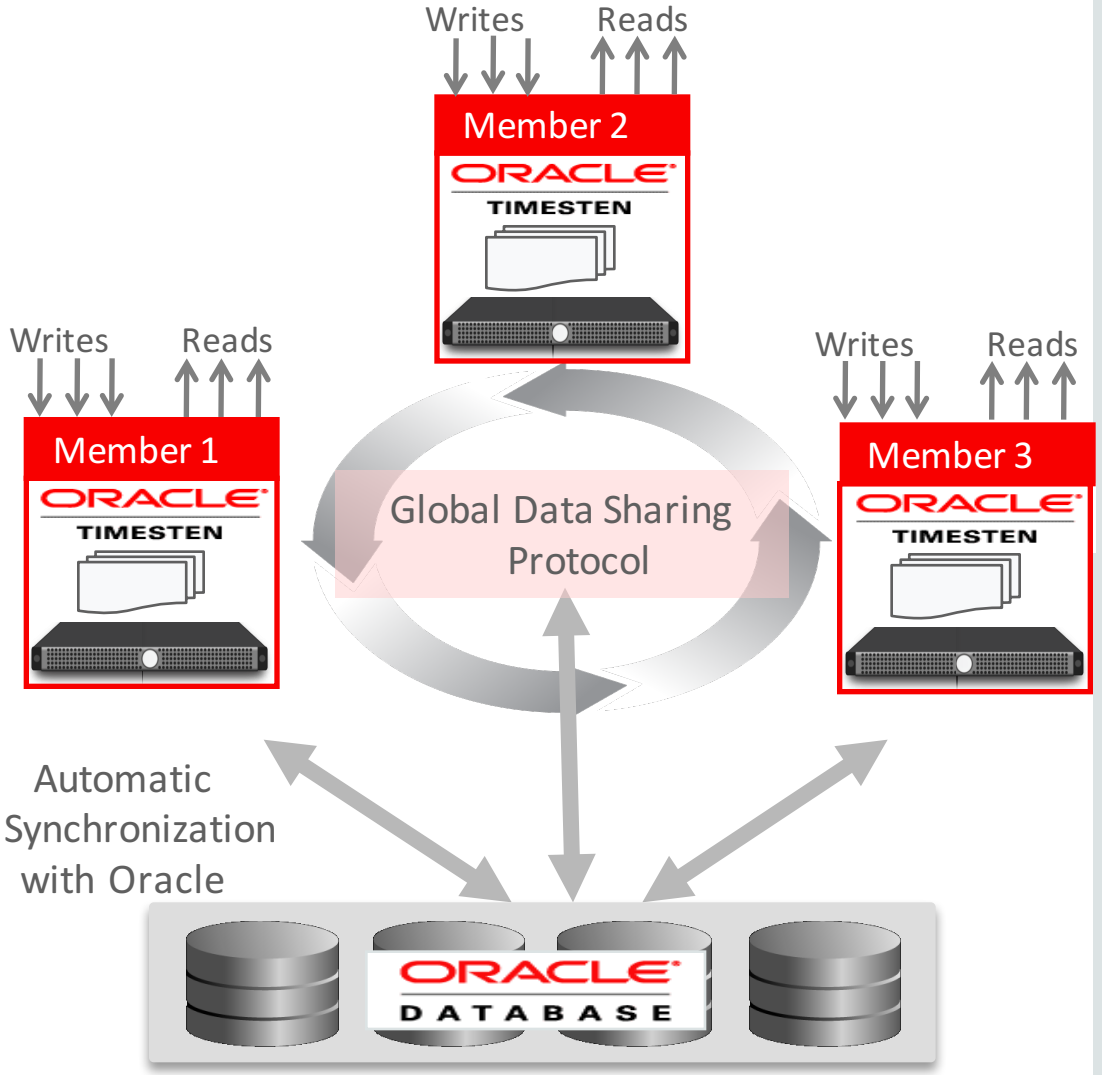
FAD #2 Scale-Out TimesTen Cache: Global Cache Groups

- Sharding requires significant application changes
- Not possible when data is not partitionable
  - Friends and family calling plans
  - Networks of friends in social networking sites
  - Related items in an online catalog
- **Cache Grid** with ***Global Cache Groups***:
  - Scale-Out cache group across a *Cache Grid* of member TimesTen databases
  - Cache instance as unit of cache coherence
  - Data shipping between grid members
  - Uses Oracle as a shared database

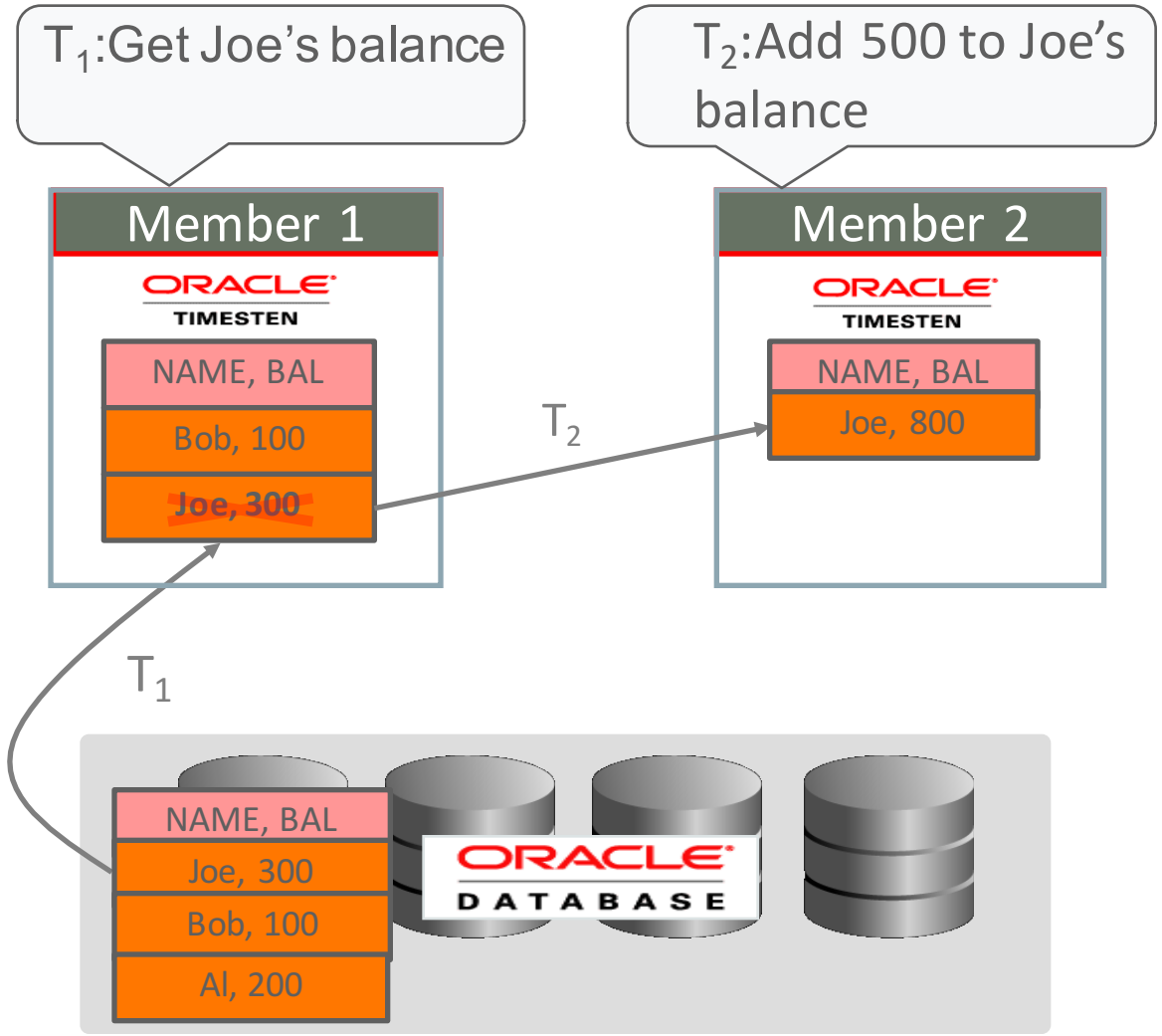


Scale-Out TimesTen Cache: Cache Grid (TimesTen 11g)

- Global Cache Groups:
  - All cache instances accessible everywhere
  - On demand loading from other grid members
  - On demand loading from Oracle when not in grid
- Transactional consistency – only committed versions shipped between grid members
- Automatic data synchronization with Oracle Database

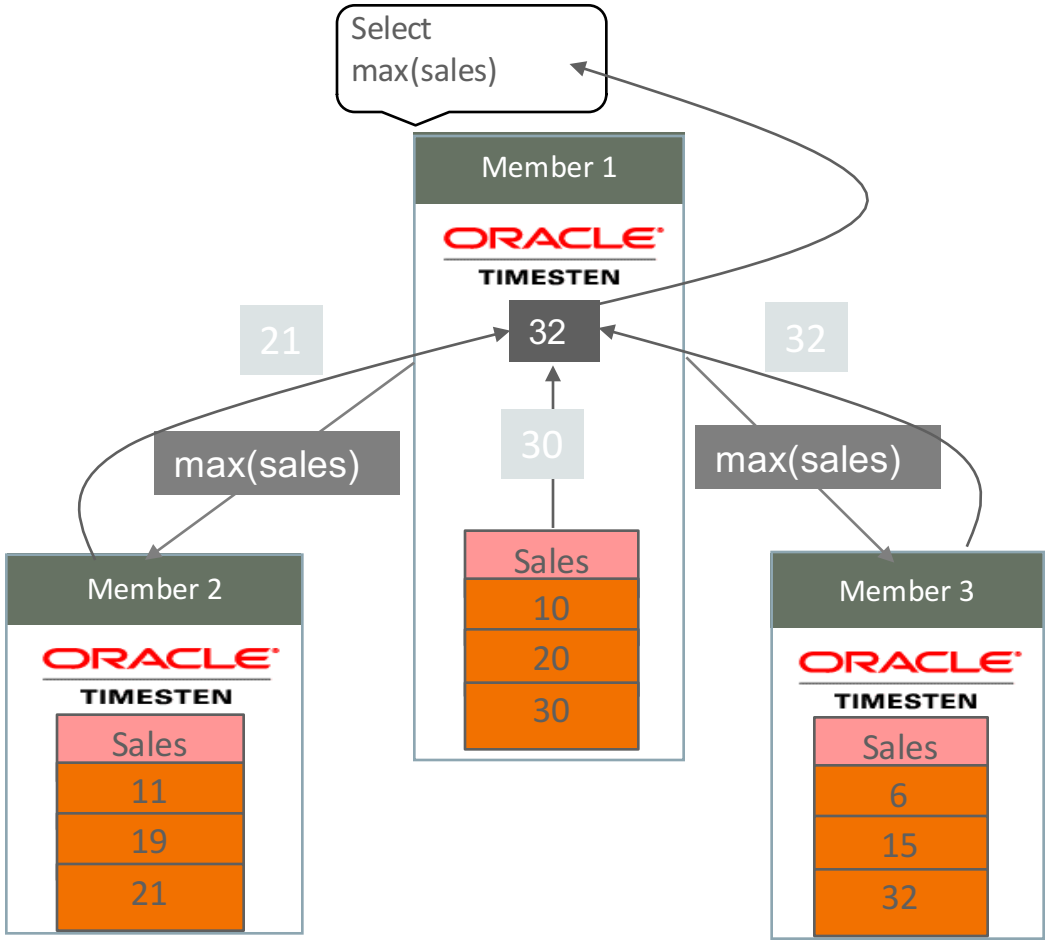


Global Data Sharing Example



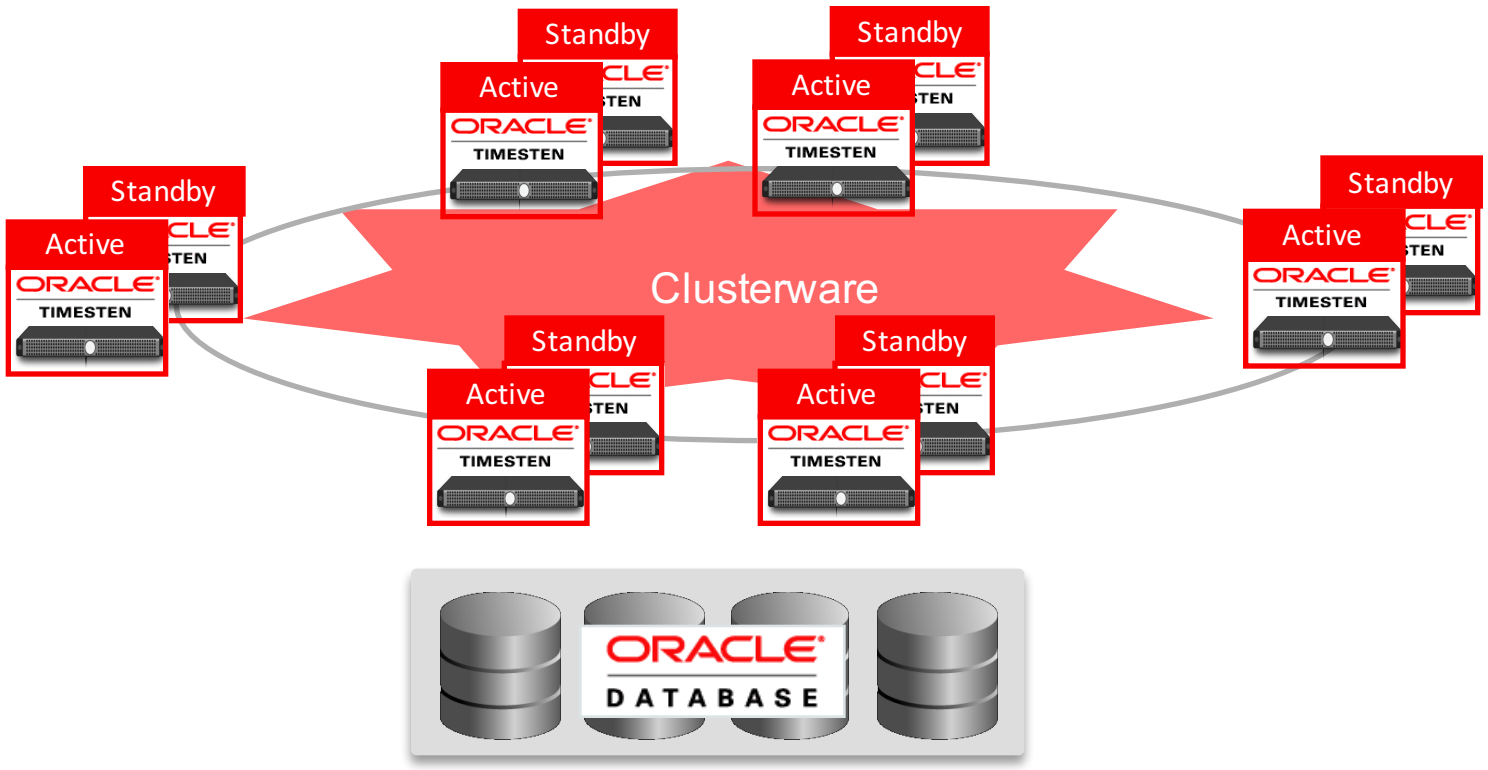
- Data loaded from Oracle when not present in grid
- Data-transfer between members on reference
- Location of cache instances in grid tracked using ownership tables in Oracle Database

Parallel Query on Global Cache Groups



- Parallel query across grid for reports
- Query Coordinator:
  - Determines sub-query to send to all members
  - Broadcasts query to members
- Members execute in parallel
- Coordinator performs final processing (sort, grouping, filter or aggregation)

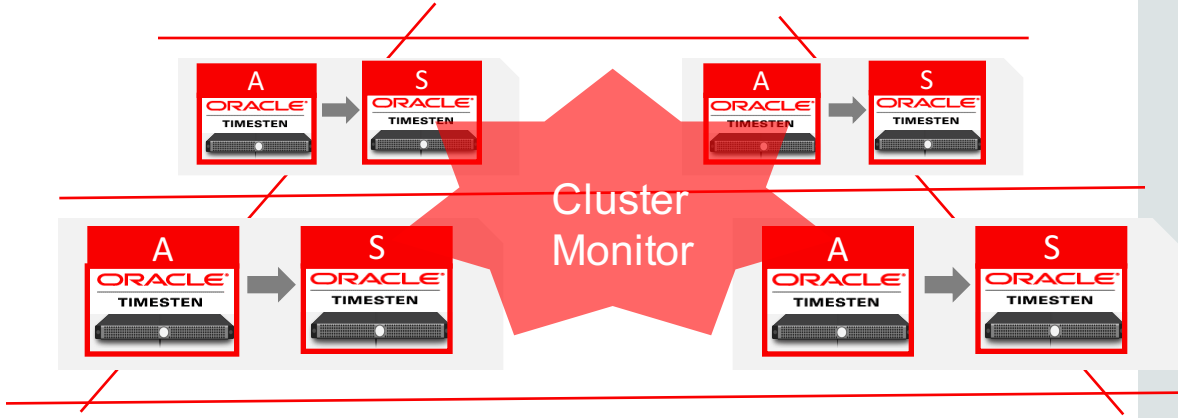
High Availability of Cache Grid



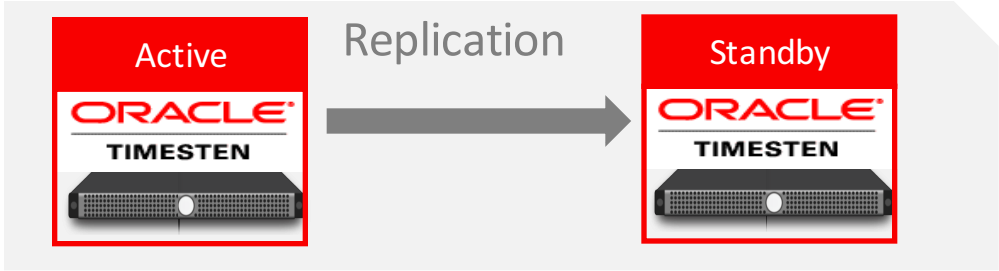
- Each Grid Member can be an Active-Standby pair for fault tolerance
- Automatic failover when Active or Standby die in any member
- If both Active and Standby die, data automatically redistributes by consistent hash
- Oracle Clusterware monitors and reconfigures grid members

Multiple Levels of High Availability

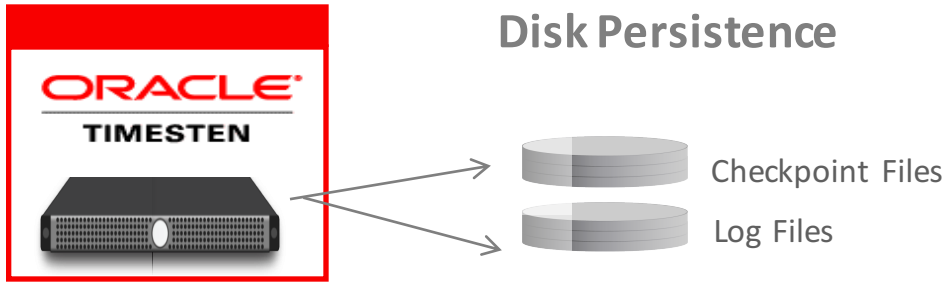
Grid HA
Cluster Monitor
Automatic Grid Reconfiguration
Automatic App Failover



Grid Member HA
Replication
Automatic Host Failover
Automatic App Failover



Database Persistence
Transactions
Checkpointing
Logging
Recovery



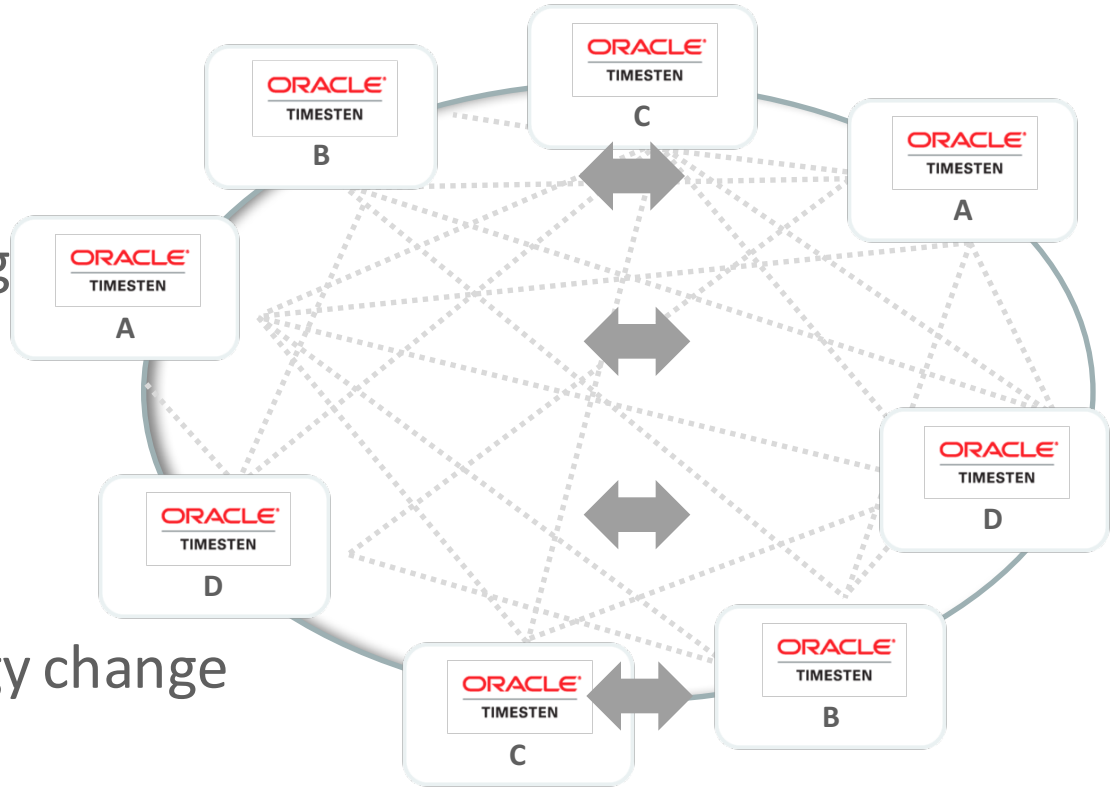
Global Cache Groups Post Mortem

- Pluses: Scale-Out, Elastic, HA, SQL, Transactions, etc, but …
- Minuses:
  - Coherency is not transparent (only coherent for primary key / cache instance access)
    - Arbitrary updates can mess things up since different nodes get out of sync
  - Data sharing is expensive:
    - Data movement requires writes to mapping tables on Oracle Database
    - Modified data must first be written to Oracle Database
  - Total cost (managing a large grid of separate databases)
- Verdict:

Spiritual Successor to Cache Grid: Velocity Scale (Beta)

Do the Hard Work: Transparent, shared-nothing, scale-out In-Memory Database

- Single DB image for all applications
  - No restrictions on SQL (unlike sharding)
  - Data distribution by consistent hash
  - Parallel Query / DML with function shipping
- Built-in HA via fully-active node copies
  - Node copies automatically kept in sync
- Fully elastic
  - Data automatically redistributes on topology change
  - Workload automatically uses new nodes



Conclusions

- Application-tier database caches can be awesome
- Most important: Transparency, Transparency, Transparency
  - Compatibility with source Database
  - Ease of application migration to cache-based architecture
- Also important: Total cost of solution
  - Software cost, manageability cost, install, upgrade patching etc.
- Early-stage big-picture architecture decisions are super-critical
- Your scientists were so preoccupied with whether they *could*, they didn't stop to think whether they *should*

Q&A

Integrated Cloud

Applications & Platform Services