Oracle TimesTen Scaleout

Building a Scale-Out In-Memory OLTP RDBMS

Tirthankar Lahiri  @dbinmemory
Vice President, Data and In-Memory Technologies

Yu-Han Chou
Development Manager, TimesTen

Ananth Raghavan
Vice President, TimesTen
In-Memory Database Technologies at Oracle

**In-Memory Row Store**

- Application
- Application
- Application

**TimesTen In-Memory Database**
- Embeddable In-Memory Database for Application Tier
- Primary use case: Latency-critical custom OLTP
  - Microsecond Response Time
  - Millions of Transactions per second

**In-Memory Column Store**

- Application
- Application
- Application
- Application

**Oracle Database In-Memory**
- Dual-Format In-Memory Database
- Primary use case: Real Time Analytics on any source
  - Billions of Rows/Sec analytic data access
  - Much faster hybrid OLTP (HTAP) workloads
Agenda

1. TimesTen Intro
2. Brief History of TimesTen
3. Why Scale-Out
4. What it takes to build one
5. Design Challenges
6. How fast is it
7. Summary
Oracle TimesTen In-Memory Database

Relational Database
- Pure in-memory
- ACID compliant
- Standard SQL
- Entire database in DRAM

Persistent and Recoverable
- Database and Transaction logs persisted on local disk or flash storage

Extremely Fast
- Microseconds response time
- Very high throughput

Highly Available
- Active-Standby and multi-master replication for HA and DR
- Very high performance parallel replication
TimesTen Application-Tier Database Cache

- Cache subset of Oracle Database tables in TimesTen for better response time
  
  - Read-write caching
    - Transaction execution and persistence in TimesTen
  
  - Read-only caching
    - Transactions executed in Oracle Database
  
- HA and fault tolerance in the application-tier
Very Widely Used Relational In-Memory Database
Deployed by Thousands of Companies
Agenda

1. TimesTen Intro
2. Brief History of TimesTen
3. Why Scale-Out
4. What it takes to build one
5. Design Challenges
6. How fast is it
7. Summary
HP Labs

- Research project started by Marie-Anne Neimat
- 1994
- “Smallbase”
- Named after HP’s Allbase database
- Used in HP Opencall telecom product, 1995

- Team Members:
  - Marie-Anne Neimat
  - Sherry Listgarten
  - Kurt Shoens
  - Kevin Wilkinson
DataXel Corporation

- November 1, 1996
- Spun out as separate company from HP
- With the...
  - HP workstations on their desks
  - One “server”
  - Source code for Smallbase
- Office:
  - 800 Menlo Ave, Menlo Park California

- Founders:
  - Marie-Anne Neimat (HP)
  - Sherry Listgarten (HP)
  - Kurt Shoens (HP)
  - Gautam Bhargava (SGI)

- Early employees:
  - Christina Terzakis
  - **Sam Drake**
  - Danny Lau
  - Rohan Aranha
  - Mark McAuliffe
  - Doug Fang
  - Neda Hajimohamadi (later Nikoo)
  - Interim CEO Mike Manley
TimesTen Performance Software, Inc.

- Permanent CEO Jim Groff
- VP Marketing Tim Shetler
- Tim’s first project: rename the company. 😊
- TimesTen chosen since it was 10 times faster than Oracle
- Marketing team:
  - Bill Hedge
  - Teri Whitaker
- VP Sales: Ed McGinnis
- VP Sales Consulting and Operations: Peter Filice (1998-)
What do you sell before you have a product?

Your people
What do you sell before you have a product?

Your people
What do you sell before you have a product?

Your people
Release History

• TimesTen 1.9 – September 1997
  – Pre-release of first actual product
  – ODBC API
  – AIX, Windows NT, Solaris, HP-UX

• TimesTen 2.0 - January 1998
  – First actual customer product
  – Shipped on new-fangled CD-ROM
Oracle TimesTen Time Line ...
20 Years of In-Memory Development

- **Pre-Oracle acquisition**
  - 1998 First commercial In-Memory RDBMS
  - HA Replication
  - Online Upgrades
  - Application-tier Cache for Oracle Database

- **TimesTen 6**
  - TimesTen 7
  - Oracle RAC integration
  - National Language Support
  - Oracle Data Types support
  - SQL Developer Integration
  - Enterprise Manager integration

- **TimesTen 11g**
  - 11.2.1
  - PL/SQL and OCI Support
  - Oracle Clusterware Integration
  - Cache Grid for Scale Out
  - ODP .NET Support
  - BLOB, CLOB, NCLOB data types

- **TimesTen 11g**
  - 11.2.2
  - Parallel Replication
  - In-Memory Analytics
  - Columnar Compression
  - Index Advisor
  - Oracle R Support
  - In-Memory Star Join
  - Oracle Golden Gate Integration

- **TimesTen 11.2.2.x Enhancements**
  - Parallel data import from Oracle Database
  - Parallel database restart
  - Highly concurrent range indexes
  - Parallel Replication with commit order optimization
Agenda

1. TimesTen Intro
2. Brief History of TimesTen
3. Why Scale-Out
4. Building a Scale-Out Database
5. Design Challenges
6. How fast is it
7. Summary
Scale-Up within a Single Server is the first step to Scalability

- A single scale-up database instance is the most efficient way to exploit multiple CPU cores
  - Memory access in same instance is much faster than messaging across instances to access data

- Some databases require multiple instances on the same server due to scale-up limits
  - This requires messaging for data sharing: not efficient
  - Ideal: One instance per server
Hardware Trends over 20 Years

• 1996 Server:
  – Sun E450
  – 4 cpus
    • 4 hardware threads
  – 480 MHz
  – 4 GB RAM
  – 728 GB disk
  – 100 Mbit/sec Ethernet

• 2016 Server:
  – Oracle SPARC T7-4
  – 4 processors
    • 128 cores (1024 hw threads) 256 X more!
  – 4.13 GHz
  – 4 TB RAM
  – 9.6 TB disk
  – 10 Gbit/sec Ethernet 100 X more!

18 X more!
9 X more!
1024 X more!
13 X more!
Scale-Up was the focus for TimesTen for 20 years

• Since with each generation hosts keep getting more cores, more memory ...

• Focus has been on greater and greater multi-core scaling:
  – Highly concurrent indexes with optimistic concurrency control
  – Highly concurrent log generation (multi-strand redo)
  – Highly concurrent parallel peer to peer replication
  – Fine-grained concurrency control and locking algorithms

• We focused on showing scale-up performance like this ...
TimesTen In-Memory Database

Read Scalability – 32 Million Queries per Second

32 Million Queries per Second

TPTBM 100% Read
E7-8890 v3 @ 2.50GHz
4 sockets, 18 cores/socket
2 threads/core
TimesTen 11.2.2.8.0
(100M rows, 17GB)
TimesTen In-Memory Database
4.6 Million Transactions Per Second Per Processor

Mixed Workload Throughput Per Processor

TPTBM 100% Mixed Workload (80-10-5-5)
E7-8890 v3 @ 2.50GHz
1 socket, 18 cores/socket,
2 threads/core
TimesTen 11.2.2.8.0
(100M rows, 17GB)

80-10-5-5 Workload = 80% select, 10% updates, 5% inserts, 5% deletes
A Pattern Started to Form with our Customers

• TimesTen had great performance but ...
  – Database size was limited to the memory of single host
  – Data was growing faster than memory size
  – Throughput limited by single host

• Although physical hosts were growing more and more powerful
  – Customers started carving them up into 1990’s sized Virtual Machines
  – Hard to scale up on a 2-core VM 😊

• So while we were busy scaling up, customers were scaling out ...
  – Customers started to build their own clusters of independent databases. e.g. ...
### Massive-Scale Real-Time Billing

**Ericsson Sweden**

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Solution</th>
<th>TimesTen Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 5 Billion subscriptions in the world, 20% are charged via Ericsson</td>
<td>• TimesTen In-Memory Database</td>
<td>• Predictable response time</td>
</tr>
<tr>
<td>• Real-time Rating (price calculation, promotion and loyalty)</td>
<td>• TimesTen Replication</td>
<td>• Very fast SQL performance</td>
</tr>
<tr>
<td>• Real-time accounting (spending control, multi-account and units, historical usage)</td>
<td>• Shared nothing clusters</td>
<td>• High availability 99.999% up time (max down time 5 minutes per year)</td>
</tr>
<tr>
<td>• Telecom grade, 99.999% availability, quick and automatic failover</td>
<td>• Standard SQL interface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Low maintenance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Wide platform support</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Low system impact</td>
<td></td>
</tr>
</tbody>
</table>
Ericsson Charging Solutions

Scale-Out in the Application

Data partitioned across shards based on MSISDN
Each shard handles several million subscribers depending on configuration. Each TimesTen database is 10 – 50 GB in size depending on configuration.

Challenges:
• Inter shard queries and transactions – must be handled in the application
• Workload balancing – have to over provision capacity

Request Director
Complex ‘router’
1. Directs requests to applications on specific shard based on MSISDN
2. Combines and recomputes responses for shared data access

Mobile Network

Applications

TimesTen
Replication

Shard # 1

Shard # 2

Shard # n
Ericsson Charging Solutions

Simpler: Scale-Out in the Database

Mobile Network

Load balancer


Benefits:
• Transparent data distribution eliminates request director and over provisioning
Agenda

1. TimesTen Intro
2. Brief History of TimesTen
3. Why Scale-Out
4. Building a Scale-Out Database
5. Design Challenges
6. How fast is it
7. Summary
How do you go from Scale-Up to Scale-Out?

Start with a constitution of requirements:

**Article 1:** The right to have full ACID properties shall not be infringed

**Article 2:** The right to have full SQL shall not be infringed

**Article 3:** The scale-out architecture shall scale capacity AND throughput

**Article 4:** Scale-out shall improve system availability

**Article 5:** The scale-out architecture shall not be insanely hard to manage

**Article 6:** All features of the product must work with scale-out
Prepare for the Road Ahead

• Admit you were wrong when you said “Scale-up should be good enough for anybody”

• Pick the most willing architects in your team to help drive design

• Also need:
  – Networking rock stars: Efficient communication is key to scale-out performance
  – Rock star QA team: Failure testing is key, massively combinatorial failure space

• Write a lot of designs with the willingness to throw many of them away
  – But document every step and every misstep

• Stock up on tea, coffee, late night snacks ...
As time progresses, the road changes ...

• You get to the realization that this is going to be really hard
  – And it is going to take a long time.

• Amend the constitution:
  – First Amendment: Some features of the scale-up product may not work with the first release of the scale-up product (as long as there is a way to get them to work in the second release which will have absolutely everything)
  – Second Amendment: Some things may take beyond the second release

• To the stock of coffee and late night snacks ...
  – Add Alcohol
  – Organize a lot of team building events
But, there will come a time when ...

• The QA team will stop saying ”Nothing works”
• The Product Management team will say “Well, it’s not horrible!”
• The Developers will say “Wait, there’s just a few more features left ...
• The QA team will say “It might work if we stop adding more”
  – The developers will add the features anyway ...
  – The QA team will find more bugs anyway but less than the last time ...
  – This will continue for a while ...
• The QA team will one day say ”It works”
• The Product Management team will say “It looks awesome, can we ship it?”
• And finally …
Tada! TimesTen Scaleout In-Memory Database

Proven TimesTen technology with Scale-Out for High Performance, and Multi-Copy for High Availability

- For High-Velocity **Extreme OLTP** applications
  - IOT, trading, mobile, click stream, billing, orders, fraud, etc.

- Performance-Oriented Design
  - Pure In-Memory, Full SQL, Full ACID Transactions
  - Fault-Tolerant Scale-Out
    - All nodes active for read/writes
    - Multiple data copies for HA
  - Sophisticated and Parallel SQL for reporting and batch

- Extremely easy to install and operate
  - **Less than 15 mins** to install, deploy, and run application
Scale-Out Shared Nothing In-Memory Database
Single-Image DB with High Availability and Elasticity

• Appears as a single DB to applications
  - Not a sharded database

• Adding and removing DB nodes (elements)
  - Data automatically redistributed
  - Workload automatically uses new nodes

• Built-in HA via fully-active node copies
  – Node copies automatically kept in sync

• Highly compatible with Oracle (subset)
  - Data types, SQL & PLSQL, Oracle Call Interface
Agenda

1. TimesTen Intro
2. Brief History of TimesTen
3. Why Scale-Out
4. Building a Scale-Out Database
5. Design Challenges
6. How fast is it
7. Summary
Design Challenges for Scale-Out In-Memory for OLTP

• **Data Distribution**: How should I spread data across the grid

• **High Availability**: How to avoid single points of failure

• **Distributed SQL Execution**: Fully transparent SQL

• **Distributed Transaction Processing**:
  – Modify any rows, anywhere in the grid
  – Atomically across multiple statements

• **Centralized Management**:
  – Managing a 100 node grid should be no harder than managing a 2 node grid
Distribution for Very Large Tables

- Reason for Scale-Out: Large tables too big for one node
- Goals for large tables:
  - Minimize hotspots
  - Minimize redistribution cost
  - Consistent distribution algorithm across all nodes
  - Should allow online reconfiguration (nodes added, deleted)
- Hash distribution can be used for uniform access
  - Ideal if hash key is subset of primary key
  - Range or User-Defined distribution are more difficult to reconfigure online, can suffer from hotspots (hot ranges)
  - Hash values can also be computed from a client
Optimizing Joins between Related Data

• Parent Child joins very common in OLTP
  – Find all orders and shipments for a customer

• NoSQL systems require denormalization
  – This is a really bad thing for RDBMS applications
  – Avoids joins, but causes a lot of other problems
  – Space issues, DML overheads, anomalies ...

• Distribution goals for child tables:
  – Try to co-locate child rows with parent
  – Co-locate across multi-level hierarchies
    • Customer, Order, Line-Item
Optimizing Access to Hot Read-Mostly Data

• Hot read-mostly data also very common in OLTP
  – Small reference tables: Catalogs, price lists, stock symbols
  – Small dimension tables: Colors, Categories, etc.’

• Goal: try to make hot data local
  – Duplication is the best way to distribute these objects
  – All read references then run locally
  – Updates must update multiple copies but but typically not a problem since updates are less frequent
Distribution Methods for All Scenarios

Distribute Table Data by **Hash**, **Reference** or **Duplicate**

- **Distribute by Hash**
  - Primary key or user-specified columns
  - Consistent hash algorithm
  - Examples: Customers, Subscribers, Accounts

- **Distribute by Reference**
  - Co-locate related data to optimize joins
  - Based on FK relationship
  - Supports multi-level hierarchy

- **Full Duplicate**
  - Identical copies on all elements
  - Useful for reference tables
  - Read and join optimization
Other Distribution Concepts

• Scale-Out architectures must provide full location transparency
  – Be able to access and update any data from any location
  – This is how most applications start out

• Location-aware request routing can help key-value applications
  – “Smart clients” can route requests but require a lot of complex logic
  – Complexity in client is a bad thing: need server-side modules, upgrades

• TimesTen Scaleout provides a data-dependent routing API
  – API maps a distribution key to one or more elements storing the record
  – This is a best-guess, if location is wrong, request is forwarded internally
  – Client remains simple
Other Distribution Concepts

• Hard problem for scale-out OLTP: Online redistribution
  – Redistribution must run very quickly or be a background activity
  – Must be idempotent
  – Queries and DMLs should not be blocked, DDLs may be

• High level approach:
  – Query plans should not hard code location info
  – No query recompilation needed on redistribution
  – Separate logical access layer from physical access layer
High Availability

• Stuff happens. Therefore, protect from stuff
• Of course, Database Software Never Fails 😊
• But hosts fail, networks fail, storage fails ...
• Multiple copies of data needed
• Ideally active active copies for maximum performance
• Client failover on timeout or disconnect
• Quorum store also needed to detect network partitions
High Availability in TimesTen Scaleout: Replica Sets

- Elements of a database are logically grouped into *replica sets*
  - Each replica set contains K elements
- Elements in a replica set contain exactly the same data
- All elements are “active”
  - DML can execute on any and all elements
- Distributed commit keeps them in sync
  - NOT replication
  - SQL changes both elements together
- Replica sets are automatically created and managed
Replica Sets and Fault Tolerance

- Data in a replica set is *available* if any element in the replica set is available.

- Recovering element automatically resynchronizes with the other replicas.

- Multiple failures in different replica sets do not result in loss of data.

![Diagram showing replica sets and fault tolerance](image-url)
Replica Sets and Application Failover

- Applications pre-connect to other elements in replica set
- On connection failure or timeout application transparently switches to alternate element
- In flight transactions either committed exactly once or rolled back
SQL Execution

• Challenge: Must support transparent execution for all types of SQL

• Requires:
  – Minimal network round-trips for OLTP operations
  – Parallel Execution for batch and analytic operations with minimal coordination
  – Secondary Indexes
  – Distributed Joins
  – Global Sequences
  – Referential Integrity and Constraints
SQL Execution

- Shared nothing architectures need distributed SQL plans
- Logical plans must be location neutral (due to redistribution)
- Suboptimal placement may have slower performance but not wrong results
- Optimize plan for fewest round trips for simple OLTP access
- There is no (interesting) pure OLTP application any more!!
- Analytics will need to run along with OLTP:
  - Allow longer running analytics to run completely in parallel
  - Analytics should run on current, not stale data
Secondary Indexes

- Primary key based access (e.g. custid) is fastest
  - Secondary key access is also very important in OLTP (e.g. by name)
- Local secondary indexes
  - Each node indexes only its local data (hash or range)
  - Fast to maintain (local maintenance) but lookup needs broadcast
- Global secondary indexes:
  - A separate distributed table mapping index key to primary key
  - Fast lookups: Go to node storing the index key, look up table
  - Slower to maintain, since distributed transaction needed

Need both types of secondary indexes
- Global needed for unique constraints
Distributed Joins

• PK/FK joins can run locally with Distribute by Reference

• OLTP also needs joins on non distribution-key columns
  – E.g. find ads to show customer based on orders placed

• Distributed joins need cost based decision making
  – Choose join order
  – Choose indexes (global or local)
  – Choose join method
Sequences in TimesTen Scaleout

- VERY useful for generating unique values
- Monotonically increasing values are expensive, and usually not required
- Each element “checks out” a range of values to assign (default: 10 million!)
- Gives them out without global coordination
- When all used up the element checks out another range globally

Element 1:
create sequence cheap_seq;
select cheap_seq.nextval from dual;
< 1>

Element 2:
select cheap_seq.nextval from dual;
< 10000001>

Element 1:
select cheap_seq.nextval from dual;
< 2>
Transactions with Scale-Out: Consistency

• Strong consistency is a requirement for enterprise OLTP

• 2 phase commit needed for full consistency
  – Blocking protocol: Transaction remains in-doubt if the Coordinator fails
  – Quorum protocols are non-blocking but expensive especially for reads
    • \((r + w) > N\) for full consistency
    • Participant reconciliation is also very complex

• Need commit protocols that avoid blocking
  – Without making the common case slower
  – Fast eviction of slow or unresponsive participants
Transactions: **Durability**

- Synchronous disk writes are **evil** for low latency systems

- Most applications OK with data in multiple memories
  - Even if not flushed to disk

- TimesTen Scaleout also commits to memory first, disk later
  - Elements within a replica set should not share critical resources!! (same host or same power supply ...)
  - Epoch transactions force full grid-wide durability at configured interval

- On full grid failure (rare), any transaction loss is limited to transactions committed since the last epoch
Centralized Installation and Management

• Administration must be scalable up to a large grid

• All TimesTen Grid management and admin operations can be performed from a single host
  – Installing software
  – Patching software
  – Configuration
  – Database creation and management
  – Backup and restore
  – Collecting diagnostics

• Command line and SQL Developer UI interfaces
Management Instances

• All management of the grid is done from the management instance
  – You never have to log on to or copy files to other hosts manually

• Can just have one management instance
  – Not recommended for production use

• The *standby* management instance can become the active in case of failure
Agenda

1. TimesTen Intro
2. Brief History of TimesTen
3. Why Scale-Out
4. What it takes to build one
5. Design Challenges
6. How fast is it
7. Summary
Linear and Massive Transaction Scalability

**SQL Transactions/sec**

### Workload
- TPTBM telco workload
- Shipped with TimesTen for 10+ years
- 80% read / 20% write workload

### Configuration
- Oracle Bare Metal Cloud
- 64 x BM.HighIO1.36
- Oracle Linux x86 64-bit
- E5-2699 v3 @ 2.30GHz
- 10G Ethernet
- NVMe Disk storage
Linear and Massive Query Scalability

1.2 Billion SQL Selects/sec

Workload
- TPTBM telco workload
- Shipped with TimesTen for 10+ years
- 100% read workload

Configuration
- Oracle Bare Metal Cloud
- 64 x BM.HighIO1.36
- Oracle Linux x86 64-bit
- E5-2699 v3 @ 2.30GHz
- 10G Ethernet
- NVMe Disk storage
YCSB Workload A (50% Read 50% Update)

Workload:
- YCSB version 0.15.0
- 1KB record (100-byte x 10 Fields)
- 100M records / Replica Set
- Uniform Distribution

Configuration
- Oracle Bare Metal Cloud
- 32 x DenseIO.52 hosts
- Oracle Linux x86 64-bit
- Intel Platinum 8167M @2GHz
- 25G Ethernet
- NVMe disk storage
YCSB Workload B (95% Read 5% Update)

**Workload:**
- YCSB version 0.15.0
- 1KB record
  (100-byte x 10 Fields)
- 100M records / Replica Set
- Uniform Distribution

**Configuration**
- Oracle Bare Metal Cloud
- 32 x DenseIO.52 hosts
- Oracle Linux x86 64-bit
- Intel Platinum 8167M @2GHz
- 25G Ethernet
- NVMe disk storage
YCSB Workload C (100% Read)

Workload:
- YCSB version 0.15.0
- 1KB record
  (100-byte x 10 Fields)
- 100M records / Replica Set
- Uniform Distribution

Configuration
- Oracle Bare Metal Cloud
- 32 x DenseIO.52 hosts
- Oracle Linux x86 64-bit
- Intel Platinum 8167M @2GHz
- 25G Ethernet
- NVMe disk storage
Agenda

1. TimesTen Intro
2. Brief History of TimesTen
3. Why Scale-Out
4. What it takes to build one
5. Design Challenges
6. How fast is it
7. Summary
Scale-Out General Purpose OLTP is Fun but Hard

• TimesTen Scaleout is already in production use by customers but it took time to build it to production quality

• Not easy to build a full featured scale-out RDBMS for OLTP
  – Easier to scale-out analytics, key-value

• Wisdom gained:
  – Early architectural decisions really matter
  – Don’t hard code too many assumptions in the code
  – Good to focus on requirements up front than make them up along the way
  – Testing is at least as important as development
What do you sell after you have a product?
Your Product (which owes everything to its people)

Disclaimer:
Only a random subset of team shown here!
What do you sell after you have a product?

Your product (which owes everything to its people)

Disclaimer:
Only a random subset of team shown here!
Integrated Cloud
Applications & Platform Services