How to use the JDBC 4.3 Sharding API for Massive OLTP Scaling

Live for the Code

Doug Hood
Consulting Member of Technical Staff
Cloud Product Manager
Oracle Database Development
October 22, 2018

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, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Agenda

Database Scalability

- Vertical Scaling
- Horizontal Scaling

JDBC Sharding API

Beyond Sharding ☺

Q & A

What is Scalability?

- **Scalability** is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth
- A system whose performance improves after adding hardware, proportionally to the capacity added, is said to be a ***scalable system***.

What is Scalability?

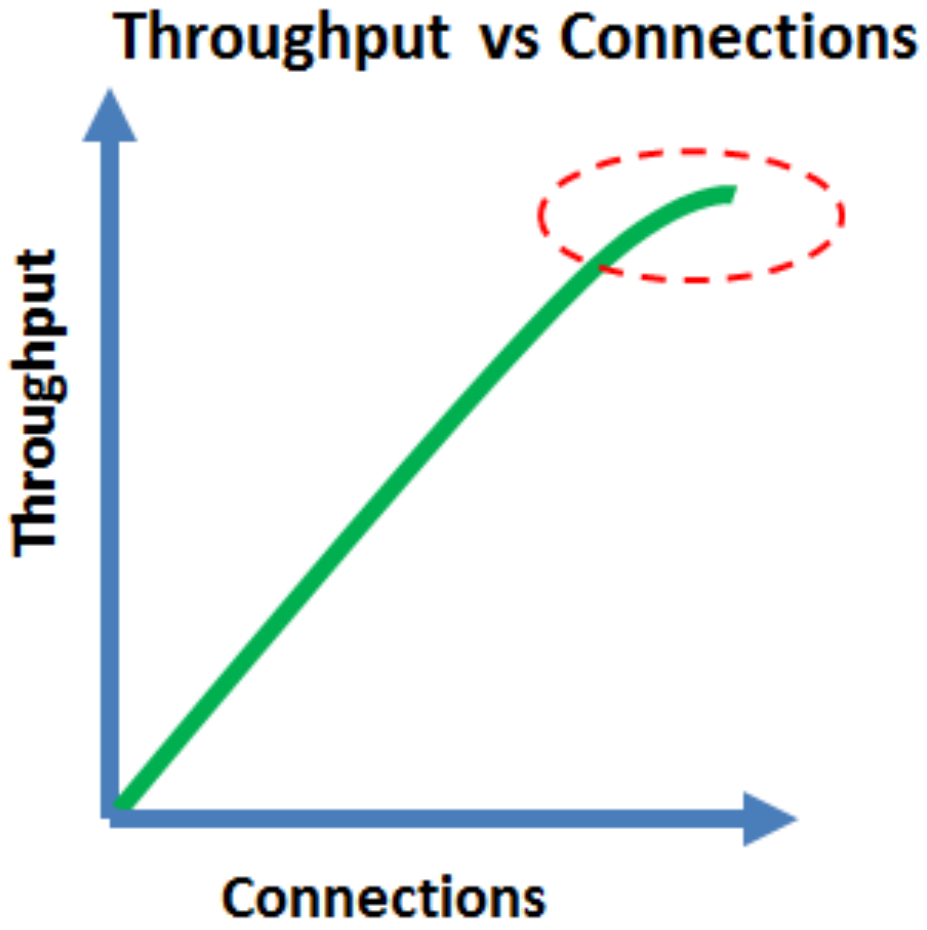
- **Scalability** is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth
- A system whose performance improves after adding hardware, proportionally to the capacity added, is said to be a ***scalable system***.

What is Scalability?

- **Scalability** is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth
- A system whose performance improves after adding hardware, proportionally to the capacity added, is said to be a ***scalable system***.

Add more **connections** to give more **throughput** with **acceptable latency**

The Linear Scalability Lie



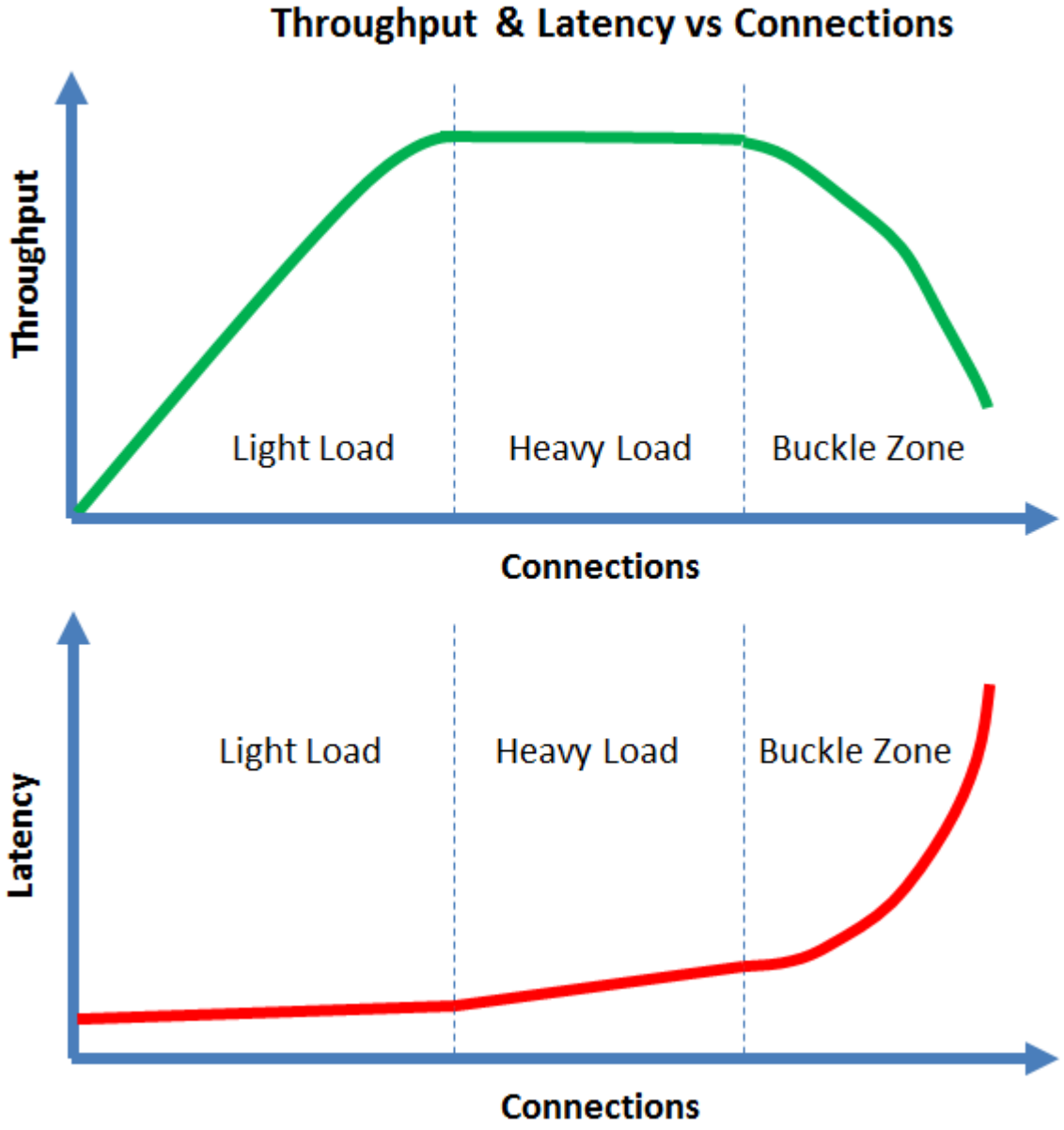
More connections = more throughput ?

Real Scalability Curves

- Throughput does NOT increase forever
- Throughput will tapper off
- Throughput will eventually get worse

- Latency will tend to get worse
- Latency will eventually get really bad

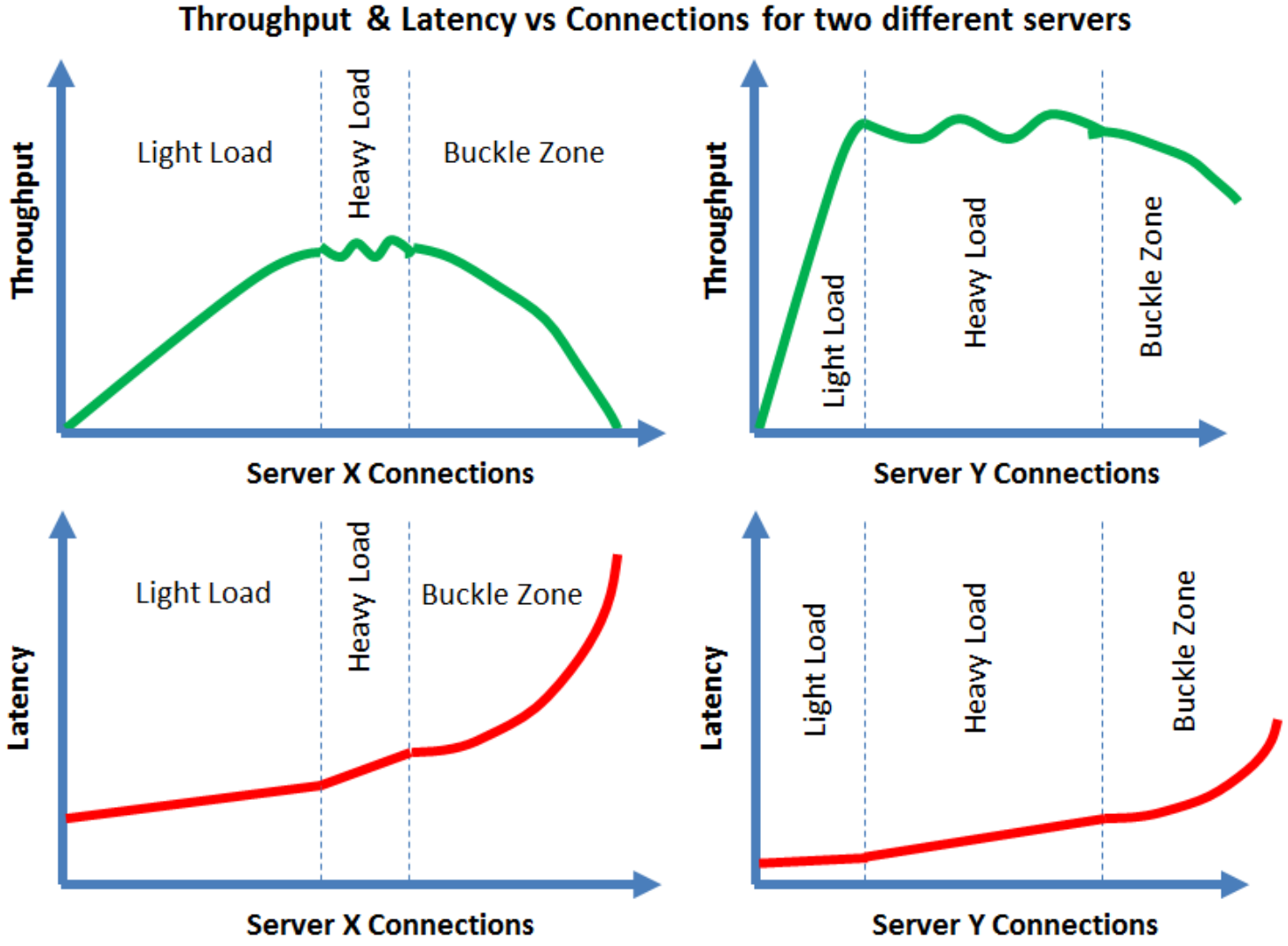
- True for **ANY** Server
  - SQL RDBMS Database
  - NoSQL Database
  - Web Server
  - App Server
  - REST Server



Different Servers = Different Scalability Curves

- Not all Servers scale the same
- You need balanced hardware to scale

Would you prefer server X or Y ?



Vertical Database Scaling

You can only go so big

IBM z14
z14 Microprocessor
32 CPUs @ 5.2 GHz
- 10 cores, 20 threads
8 TB DRAM

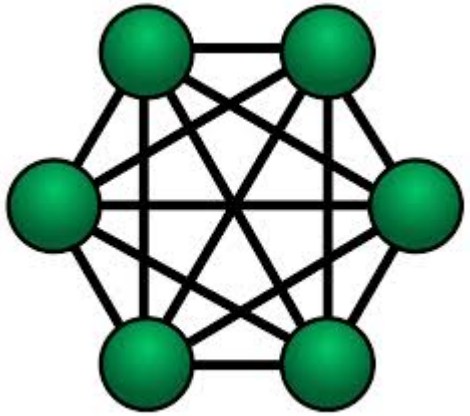
Fujitsu M12-2S
SPARC 64 XII
32 CPUs @ 4.25 GHz
- 12 cores, 96 threads
32 TB DRAM

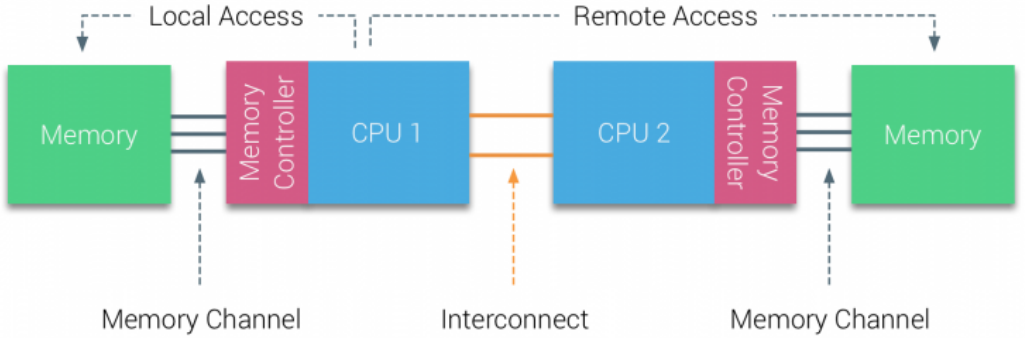
HPE Superdome Flex
Intel Xeon
32 CPUs @ 3.6 GHz
- 28 cores, 56 threads
48 TB DRAM

SGI Altix 4700
Intel Itanium 2
2048 CPUs @ 900 MHz
- 2 cores, 4 threads
128 TB DRAM

Vertical Scaling Limits

- Only so many CPUs interconnected
- NUMA limits
- Complexity & Cost
- Niche Market





8+ Sockets

4-8 Sockets

1-2 Sockets

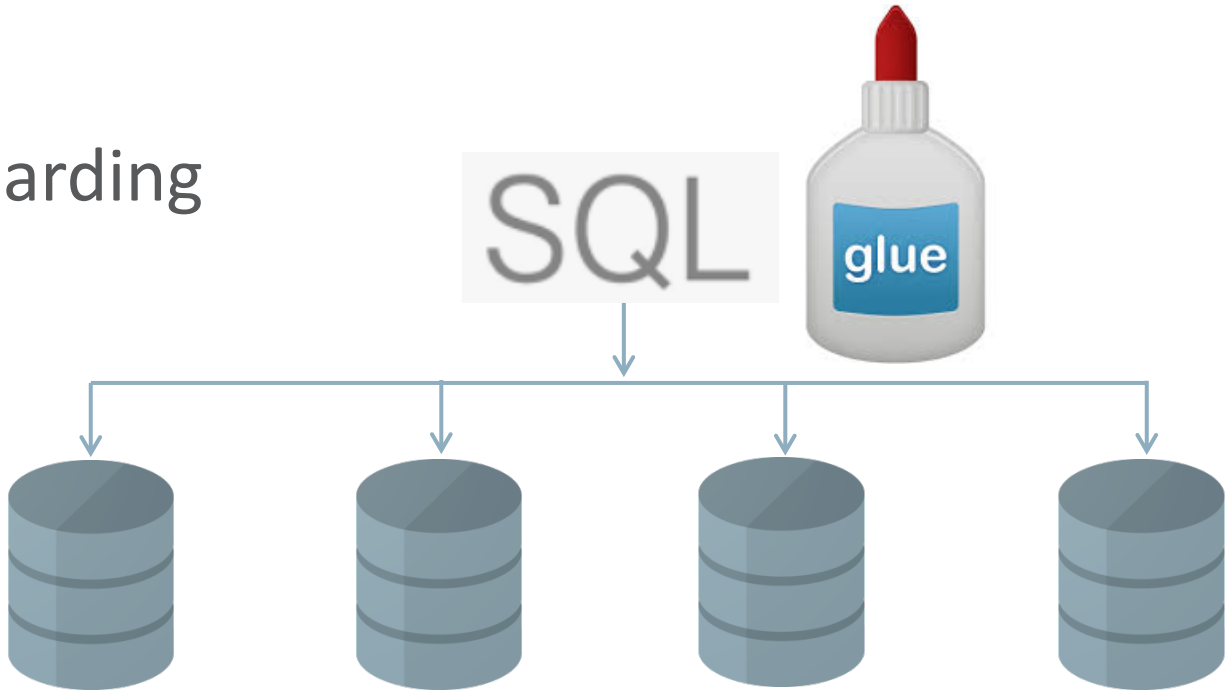
Horizontal Database Scaling

Horizontal Scaling hardware

- Use **cheap/fast Linux x8664 servers**, eg Oracle Sun X7-2
- NUMA affects are minimal
- Commodity servers keep getting **faster**, **cheaper** and **more powerful**
- 1.5 TB DRAM [Persistent Ram coming, **Tuesday Intel/Oracle PMem demo**]
- Two Intel Xeon 8164 @ 2.2 GHz, 26 cores
- Up to eight NVMe SSDs
- **42 1U servers per Rack:**
  - 2*42 = **84 CPUs**
  - 1.5 * 42 = **63 TB RAM**

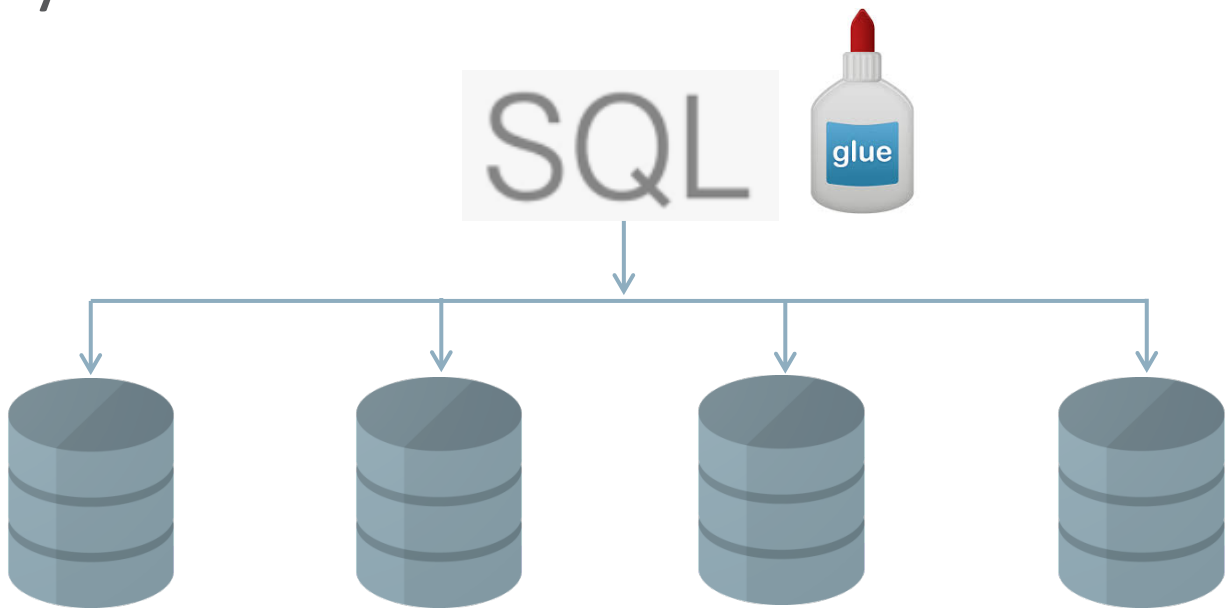
Early SQL Database Sharding

- Data is horizontally partitioned across **independent databases**
- Cross partition SQL operations are **limited or non existent**
- Need a ‘glue layer’ in the **application tier**
- **Need to pass in the ‘shard key’**
- Need to **re-write apps** to use sharding
- Need to **choose** the shard key
- **Can scale very well**



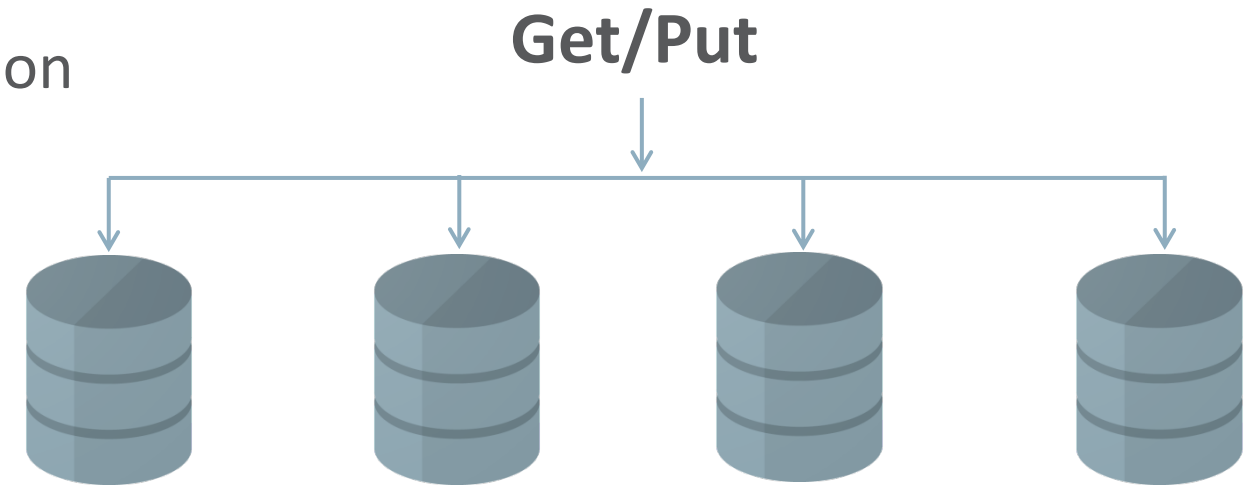
Current SQL Database Sharding

- 'Glue layer' is now **part of the database**
- **Still need to pass in the 'shard key'**
- Still need to **re-write apps** to use sharding
- Still need to **choose** the shard key
- **Can scale very well**



NoSQL Database Sharding

- Simple **Key/Value** data stores
- Data is [**automatically**] sharded across many cost effective servers
- **Get/Put** type APIs
- **Eventually** consistent rather than ACID transactions
- Data is **de-normalized**
  - Can lead to massive data duplication
  - Faster if no 'table' joins
- **Can scale very well**



JDBC 4.3 Sharding API

ORACLE

JDBC 4.3 Sharding API

- **java.sql and javax.sql**
  - Added Sharding support
  - Enhanced DatabaseMetaData to determine if Sharding is supported
- **Interfaces**
  - ShardingKey
  - ShardingKeyBuilder

```
DataSource ds = new MyDataSource();
ShardingKey shardingKey = ds.createShardingKeyBuilder()
                           .subkey("abc", JDBCType.VARCHAR)
                           .subkey(94002, JDBCType.INTEGER)
                           .build();
```

The benefits of JDBC 4.3 Sharding API

- **A Standard [JDBC] interface for any sharded or distributed SQL DB**
- **Can improve performance by eliminating network hops**

Performance of JDBC 4.3 Sharding API vs NoSQL

- Need a benchmark that works for both SQL and NoSQL
- The benchmark needs to be scalable
- Want lots of benchmark results to be useful
- Let each vendor provide their own optimal benchmark results

What is the YCSB Workload?

- YCSB : **Y**ahoo **C**loud **S**erving **B**enchmark
  - Developed at Yahoo for Cloud Scale workloads
  - Widely used to compare scale-out databases, NoSQL databases, and in-memory data grids
- A series of workload types are defined:
  - Workload A: 50% reads, 50% Updates
  - Workload B: 95% reads, 5% Updates
  - Workload C: 100% reads
- The YCSB Client cannot be changed
  - DB Vendors implement the DB Client interface in Java
  - The version and exact configuration matters

What is the YCSB Workload?

- YCSB : **Y**ahoo **C**loud **S**erving **B**enchmark
  - Developed at Yahoo for Cloud Scale workloads
  - Widely used to compare scale-out databases, NoSQL databases, and (non-durable) in-memory data grids
- A series of workload types are defined:
  - Workload A: 50% reads, 50% Updates
  - Workload B: 95% reads, 5% Updates
  - Workload C: 100% reads
- The YCSB Client cannot be changed
  - DB Vendors implement the DB Client interface in Java
  - The version and exact configuration matters

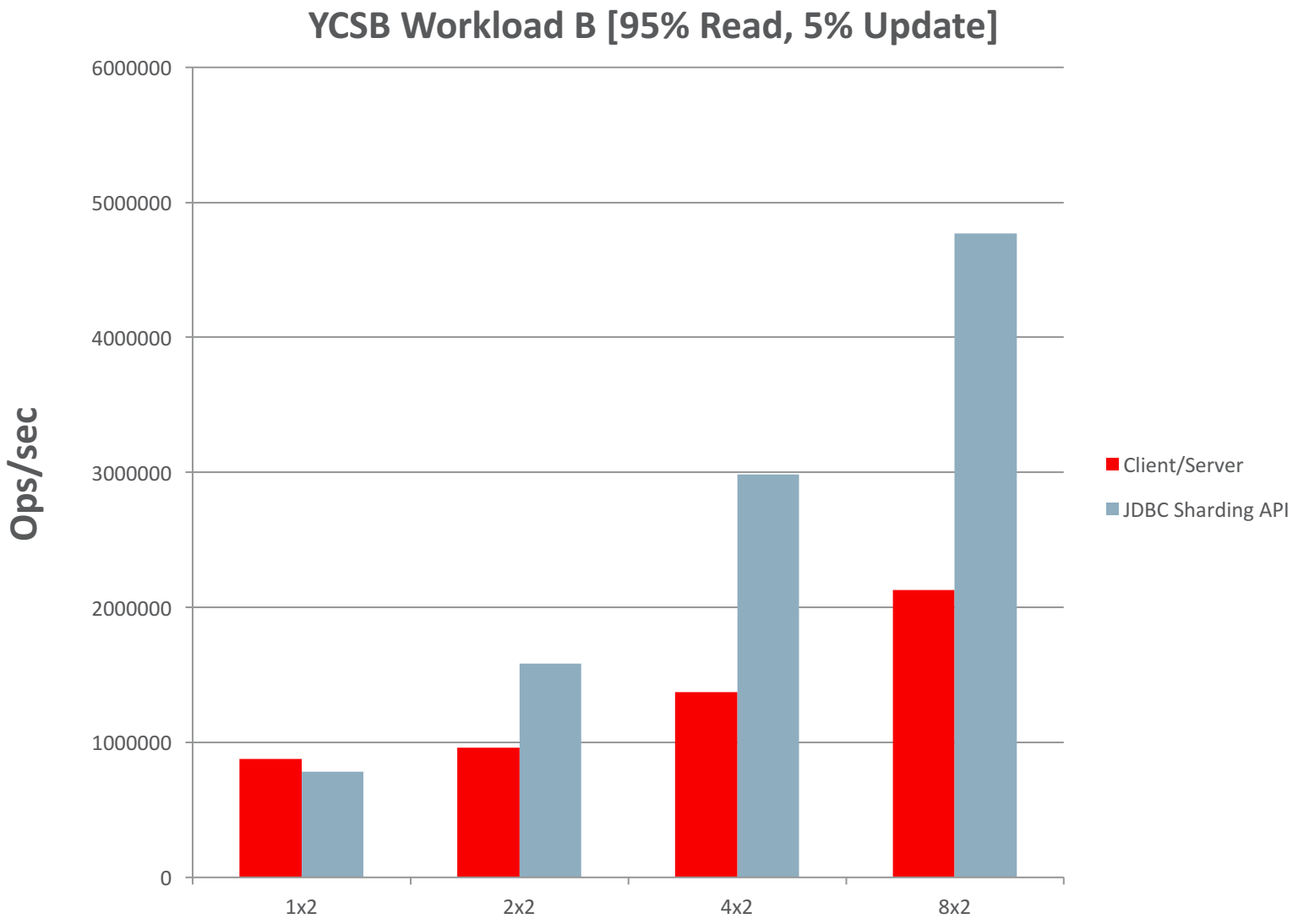
Surveyed YCSB (Workload B) Results*

Product	Type	Nodes	Ops/Sec
cassandra	NoSQL DB	32	227 K
mongoDB	NoSQL DB	2	275 K
SCYLLA.	NoSQL DB	3	715 K
VOLTDB	Scale-Out RDBMS	6	1.6 M
AEROSPIKE	NoSQL DB	8	1.6 M

** There is no official repository of YCSB results*
These were the largest results we found online

The JDBC Sharding API Scales well

- C/S load balancers can scale
  - 2.1M ops/sec YCSB B
- Sharding API scales well
  - 4.7M ops/sec YCSB B
- JDBC Sharding API is faster than all published NoSQL YCSB results!



YCSB Driver using the JDBC Sharding API

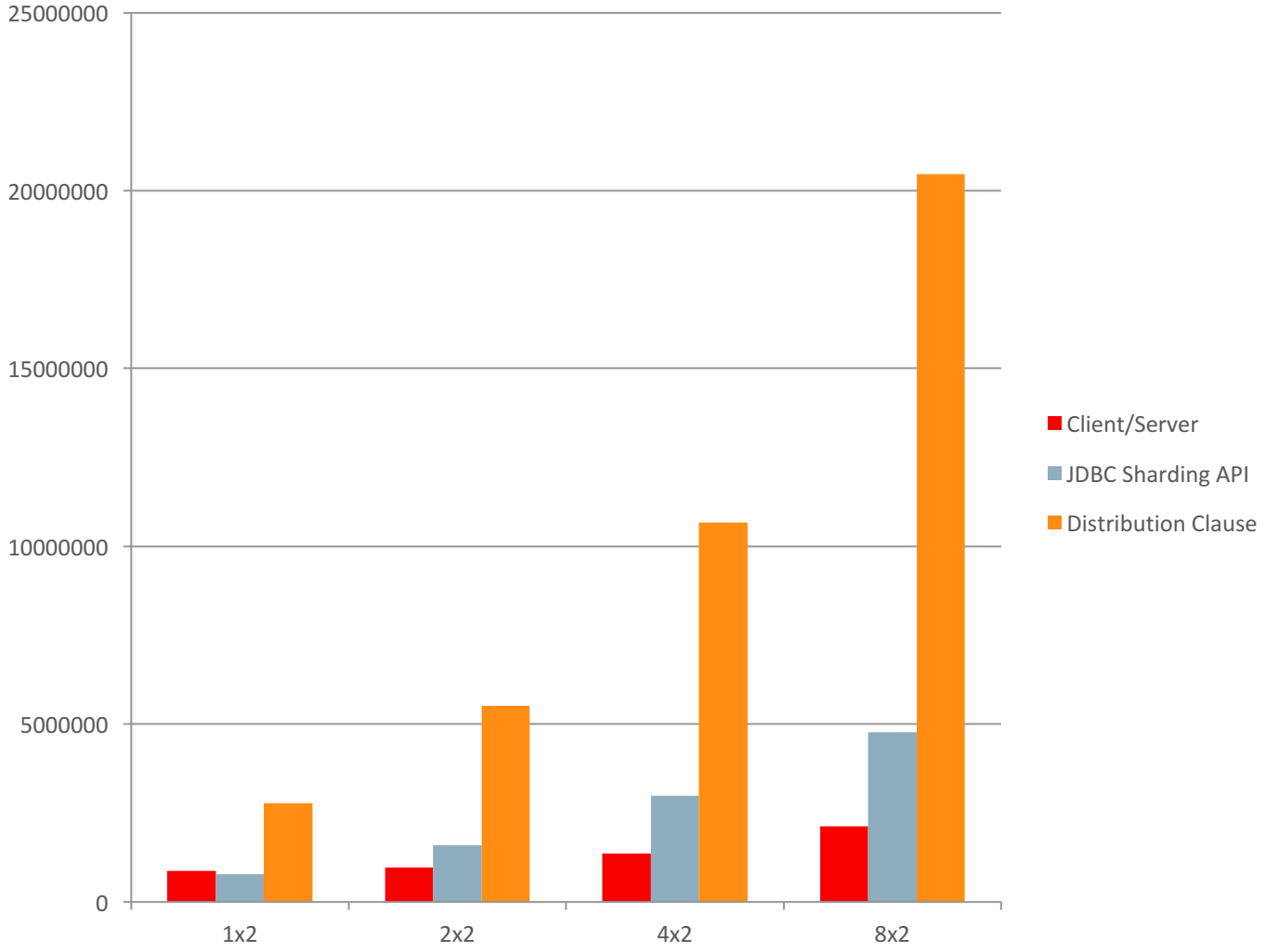
```
@Override
public Status read(String tableName, String key, Set<String> fields, Map<String, ByteIterator> result) {
  try {
    StatementType type = new StatementType(StatementType.Type.READ, tableName, 1, "", getShardIndexByKey(key));
    PreparedStatement readStatement = cachedStatements.get(type);
    if (readStatement == null) {
      readStatement = createAndCacheReadStatement(type, key);
    }
    readStatement.setString(1, key);
    ResultSet resultSet = readStatement.executeQuery();
    if (!resultSet.next()) {
      resultSet.close();
      return Status.NOT_FOUND;
    }
    if (result != null && fields != null) {
      for (String field : fields) {
        String value = resultSet.getString(field);
        result.put(field, new StringByteIterator(value));
      }
    }
    resultSet.close();
    return Status.OK;
  } catch (SQLException e) {
    System.err.println("Error in processing read of table " + tableName + ": " + e);
    return Status.ERROR;
  }
}
```

Beyond Sharding

Dominating the JDBC Sharding API

- CS load balancers can scale
- JDBC Sharding API scales well
- Distribution Clauses scale best

- Distribution Clauses
  - Minimize network hops to maximize throughput



What Hardware was Used?

Oracle Sun X7-2

- Two Intel Xeon 8164 @ 2.2 GHz, 26 cores
- 768 GB RAM
- Four NVMe SSDs
- Two 10G Ethernet

Oracle Cloud Infrastructure

- 32 * BM.DenseIO2.52

What is the YCSB Workload?

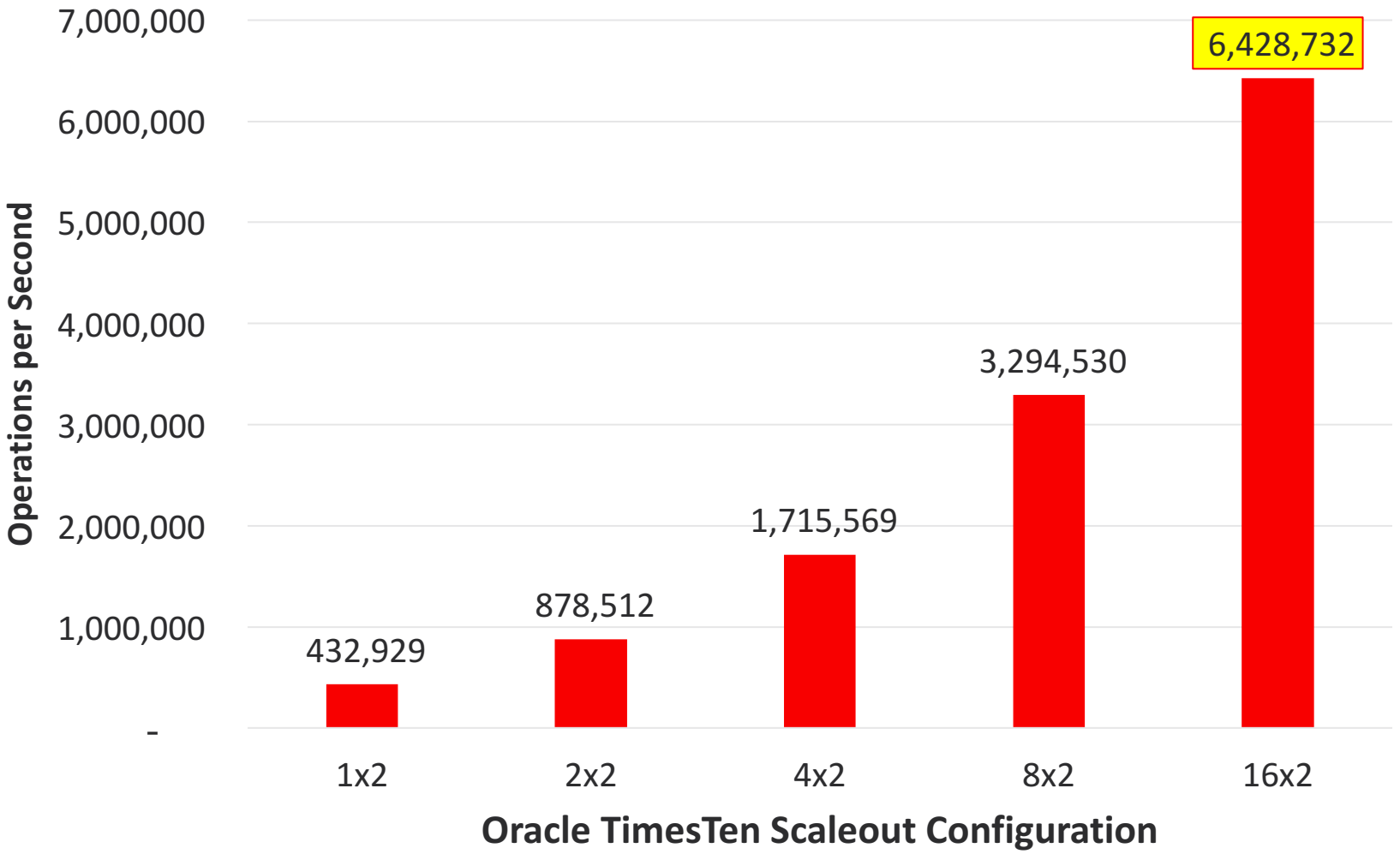
- YCSB : **Y**ahoo **C**loud **S**erving **B**enchmark
  - Developed at Yahoo for Cloud Scale workloads
  - Widely used to compare scale-out databases, NoSQL databases, and (non-durable) in-memory data grids
- A series of workload types are defined:
  - Workload A: 50% reads, 50% Updates
  - Workload B: 95% reads, 5% Updates
  - Workload C: 100% reads
- The YCSB Client cannot be changed
  - DB Vendors implement the DB Client interface in Java
  - The version and exact configuration matters

Surveyed YCSB (Workload B) Results*

Product	Type	Nodes	Ops/Sec
cassandra	NoSQL DB	32	227 K
mongoDB	NoSQL DB	2	275 K
SCYLLA.	NoSQL DB	3	715 K
VOLTDB	Scale-Out RDBMS	6	1.6 M
AEROSPIKE	NoSQL DB	8	1.6 M

** There is no official repository of YCSB results*
These were the largest results we found online

YCSB Workload A (50% Read 50% Update): **6.4 Million** Ops/Sec



YCSB version 0.15.0

- 1KB record
  (100-byte x 10 Fields)
- 100M records / Replica Set
- Uniform Distribution

TimesTen Scaleout

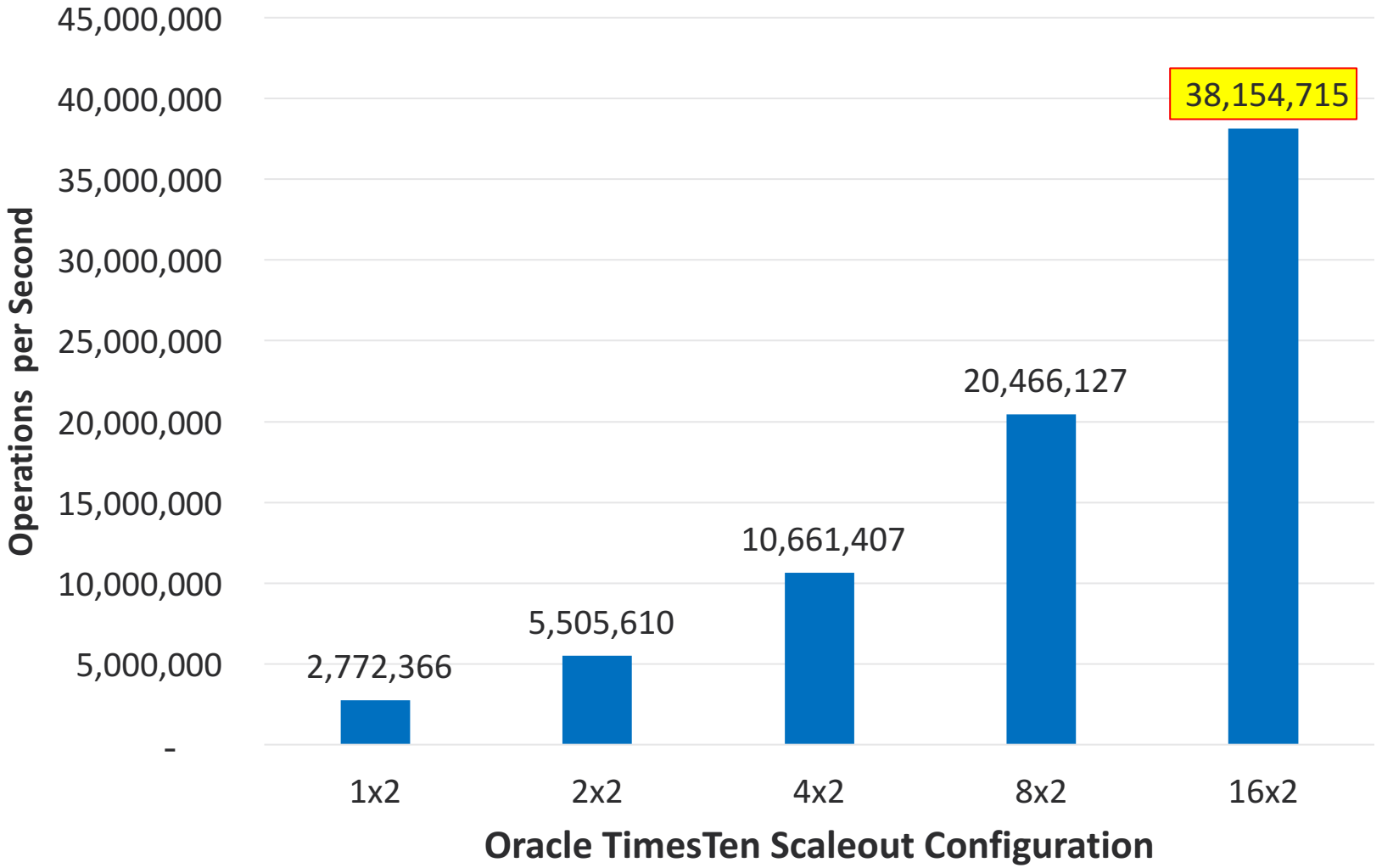
- 1 to 16 replica sets
- 2 synchronous replicas per replica set

Oracle Cloud Infrastructure

- 32 * BM.DenseIO2.52

YCSB Workload B (95% Read 5% Update): **38 Million** Ops/Sec

Reminder: The best YCSB-B result found in our survey was **1.6 Million** Ops/Sec



YCSB version 0.15.0

- 1KB record
  (100-byte x 10 Fields)
- 100M records / Replica Set
- Uniform Distribution

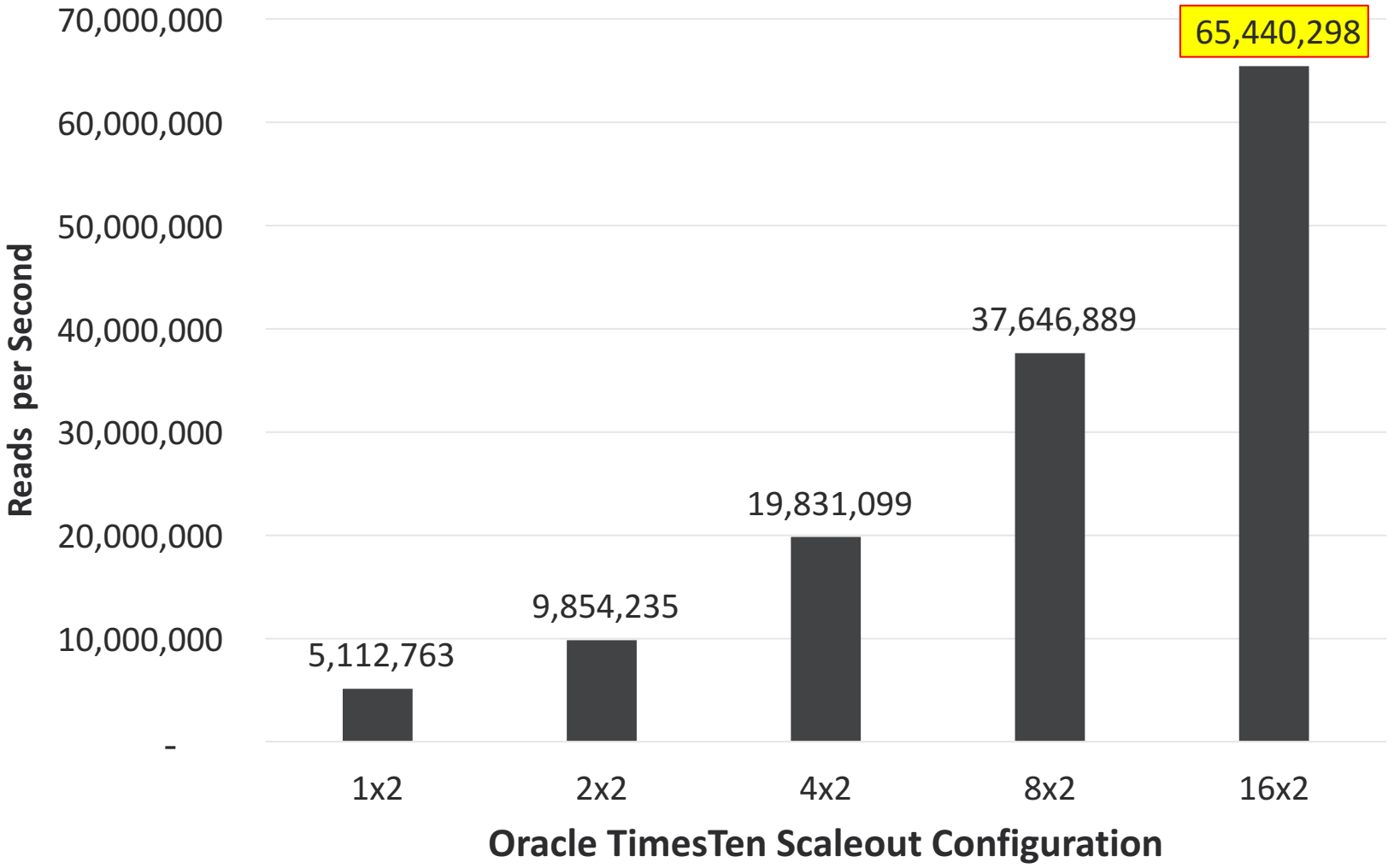
TimesTen Scaleout

- 1 to 16 replica sets
- 2 synchronous replicas per replica set

Oracle Cloud Infrastructure

- 32 * BM.DenseIO2.52

YCSB Workload C (100% Read): **65 Million** Reads/Sec



YCSB version 0.15.0

- 1KB record
  (100-byte x 10 Fields)
- 100M records / Replica Set
- Uniform Distribution

TimesTen Scaleout

- 1 to 16 replica sets
- 2 synchronous replicas per replica set

Oracle Cloud Infrastructure

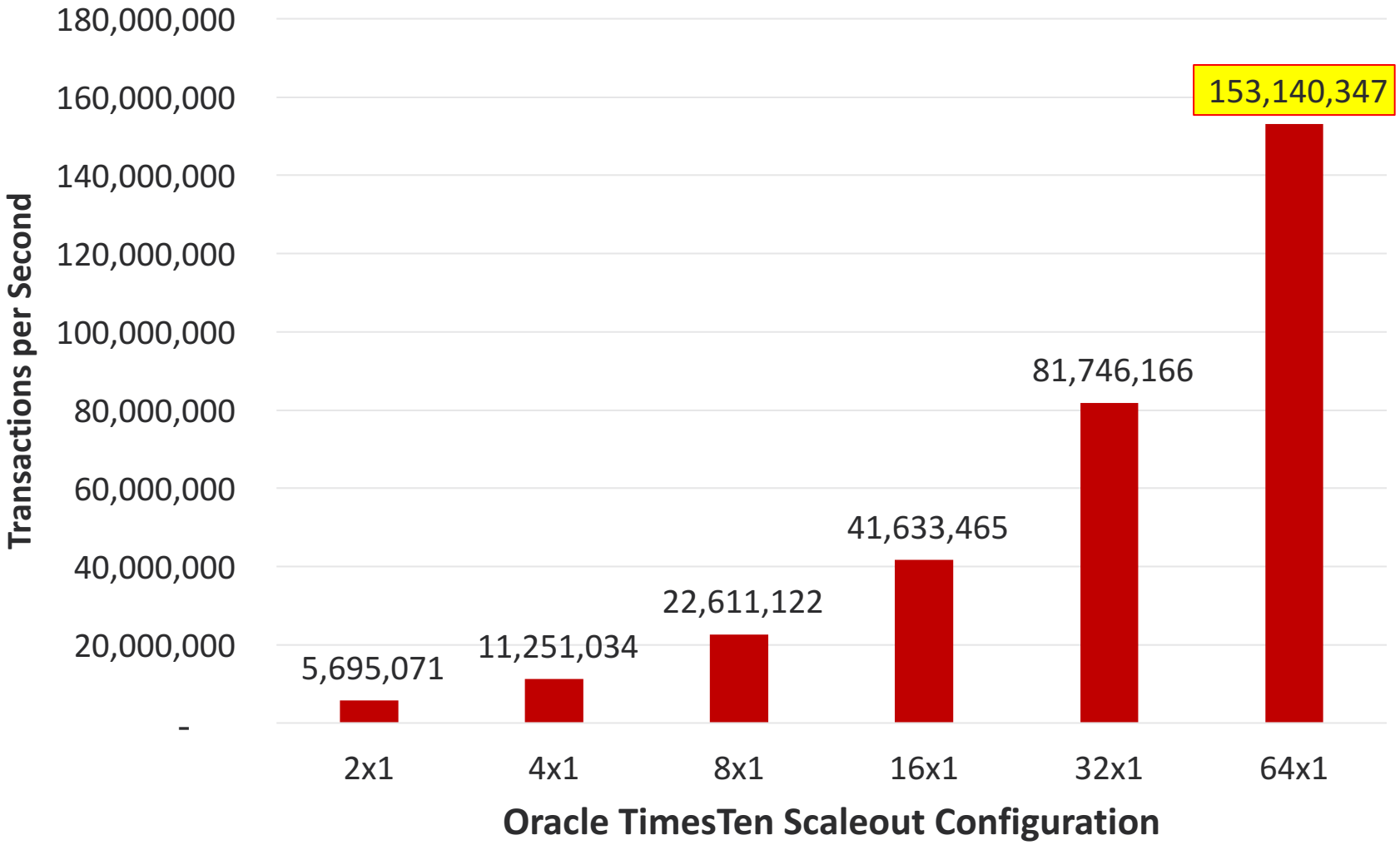
- 32 * BM.DenseIO2.52

What is the TPTBM Workload?

ORACLE TimesTen

- TPTBM : **T**elecom **P**rovider **T**hroughput **B**ench**M**ark
  - A benchmark originally developed by the TimesTen team
  - Represents common operations on a Telecom Subscriber database
  - Uses *standard* SQL and *standard* database APIs
  - Shipped with Oracle TimesTen as C and Java source code for the past 15 years
  - Quickly demonstrates the performance of user's hardware
- Common workload mixes:
  - 80% Reads, 20% Updates
  - 100% Reads
- The version and exact configuration matters

TPTBM 80% Read 20% Update: **153 Million** Transactions/Sec



TPTBM Configuration

- 128-byte record
- 100M records / Replica Set
- Uniform Distribution

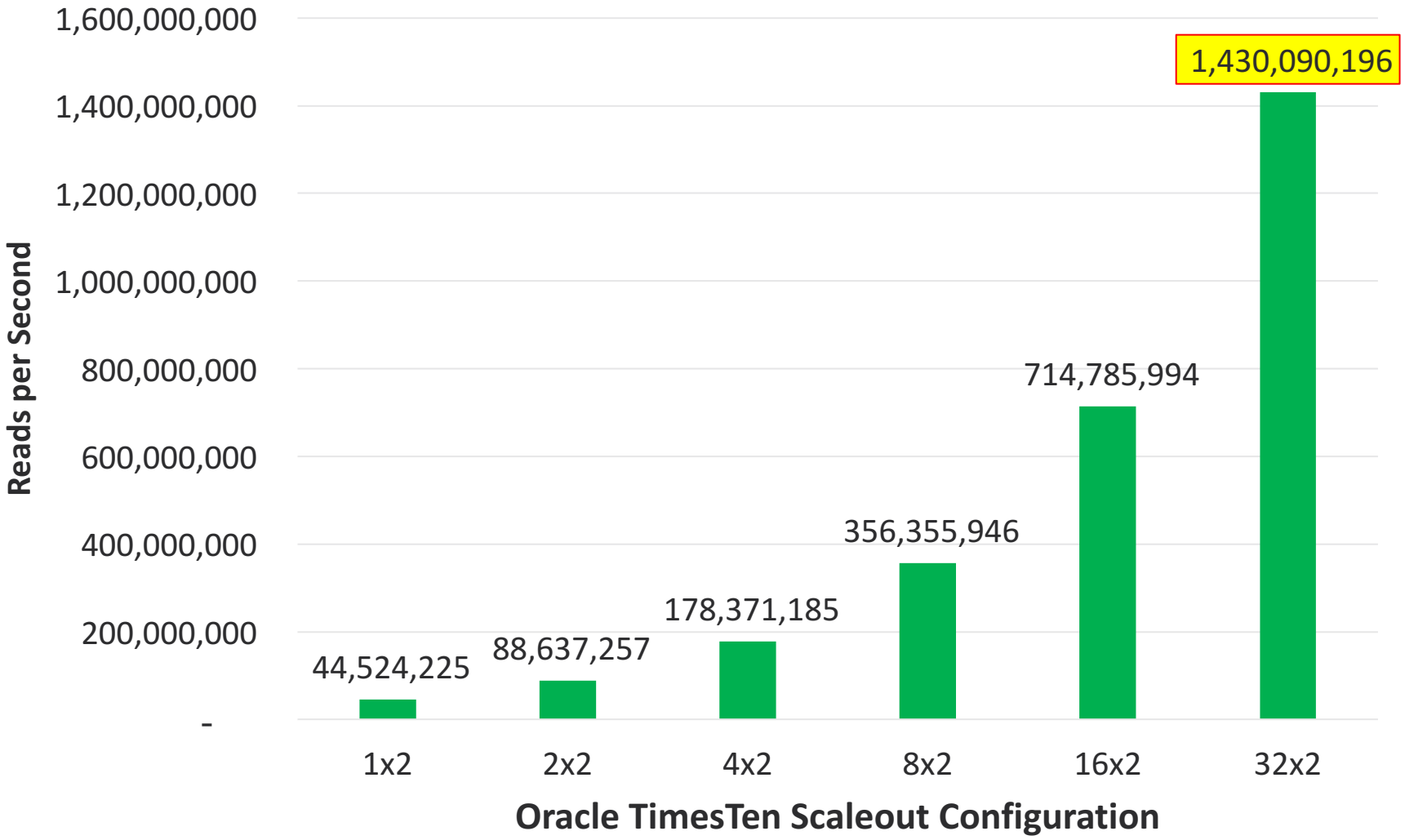
TimesTen Scaleout

- 1 to 64 replica sets
- 1 replica per replica set

Oracle Cloud Infrastructure

- 32 * BM.DenseIO2.52
- Two TimesTen instances per compute node

TPTBM 100% Read: **1.4 Billion Reads** Per Second!!



TPTBM Configuration

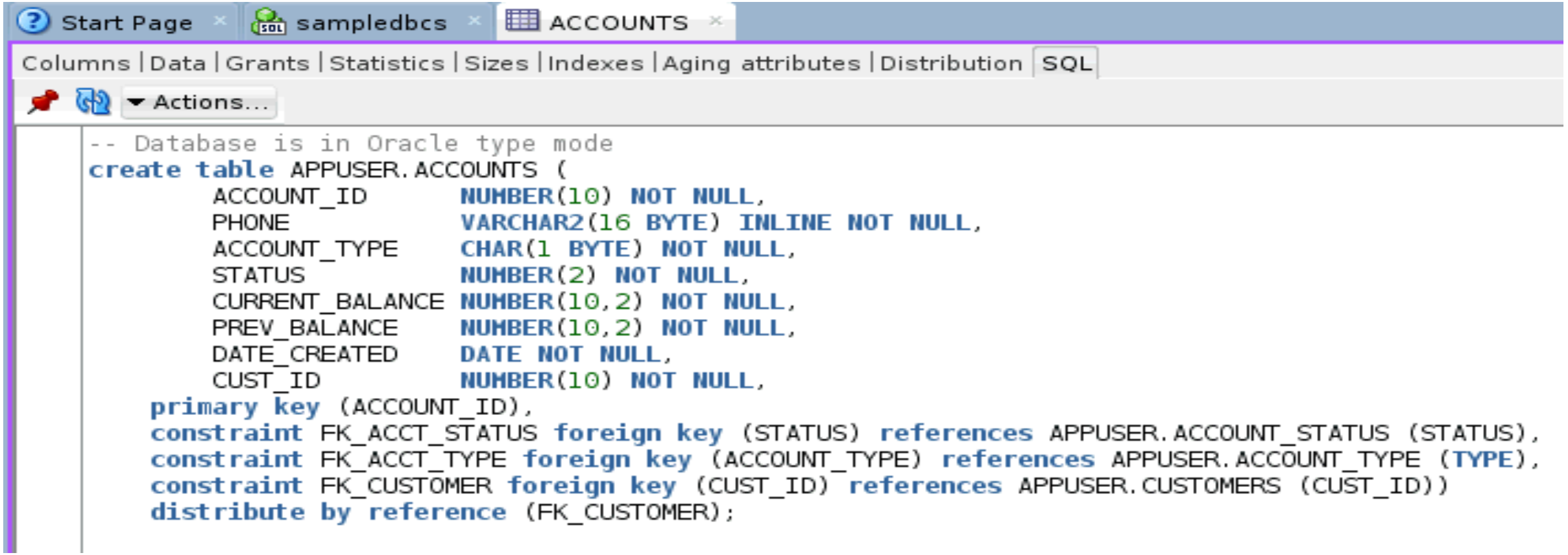
- 128-byte record
- 100M records / Replica Set
- Uniform Distribution

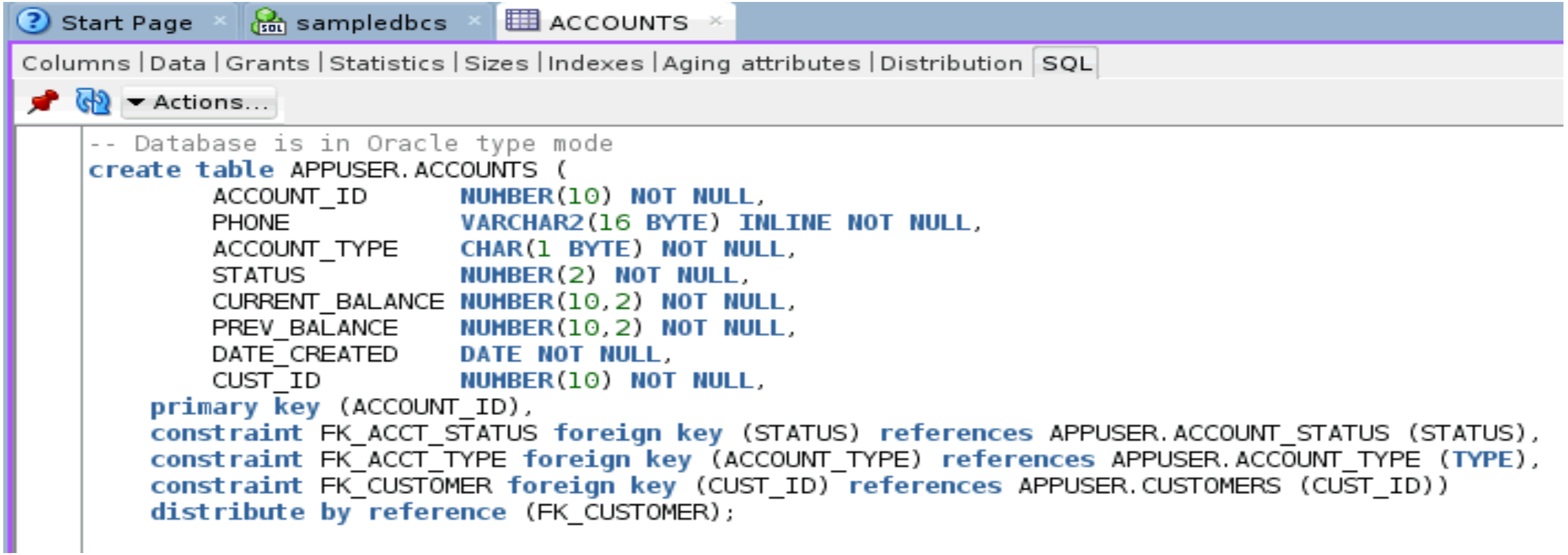
TimesTen Scaleout

- 1 to 32 replica sets
- 2 synchronous replicas per replica set

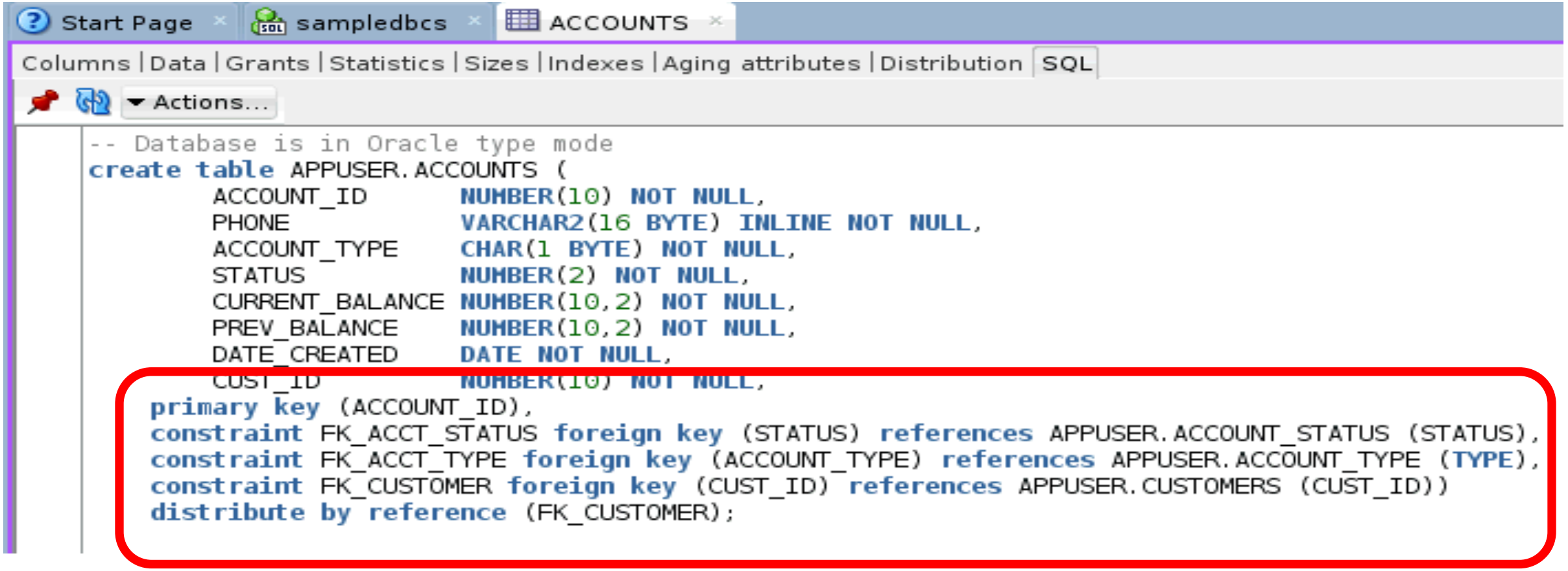
Oracle Cloud Infrastructure

- 32 * BM.DenseIO2.52
- Two TimesTen instances per compute node





Scalability due to SQL optimizations, not API tricks



Start Page | sampledbcs | ACCOUNTS

Columns | Data | Grants | Statistics | Sizes | Indexes | Aging attributes | Distribution | SQL

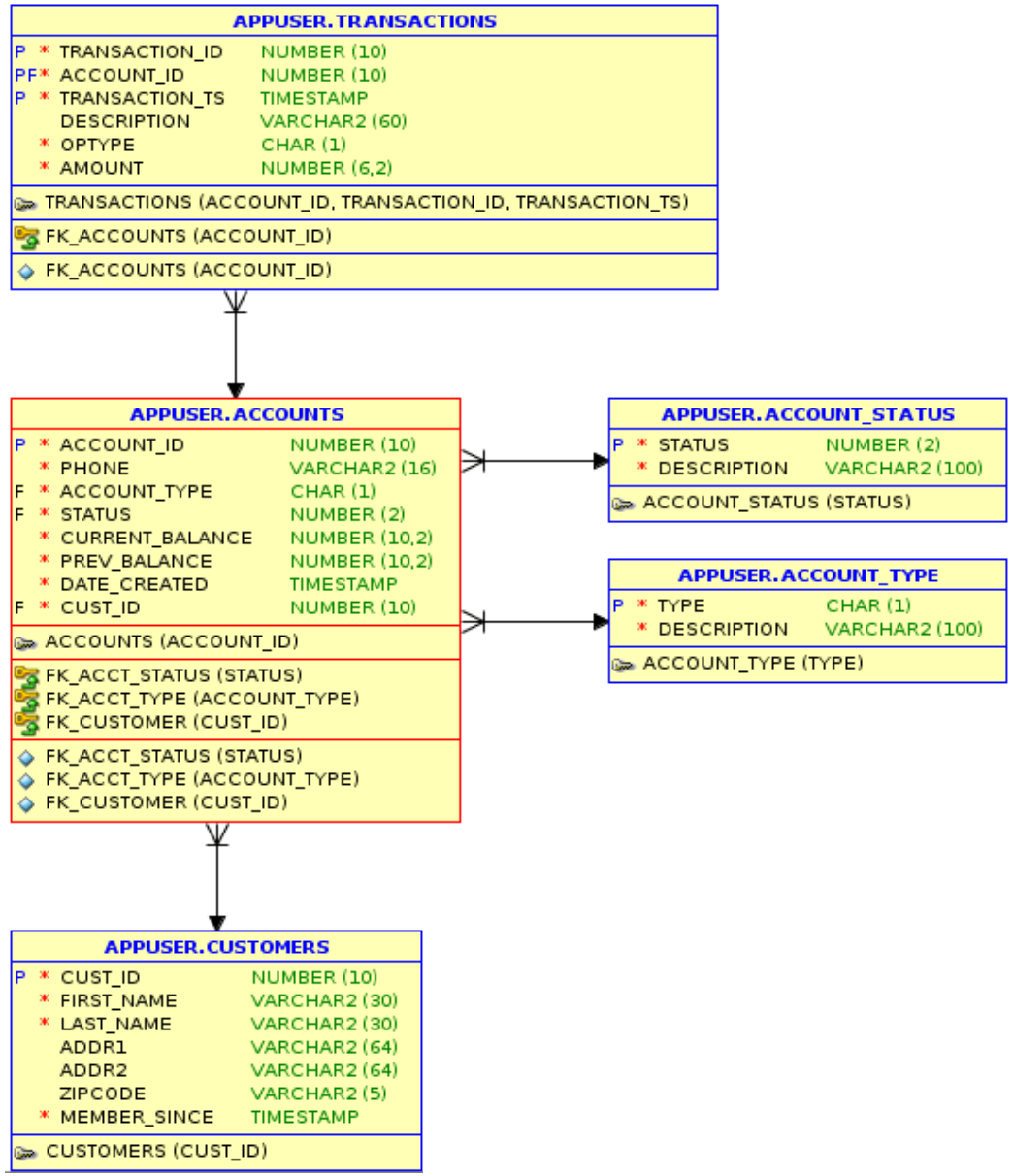
Actions...

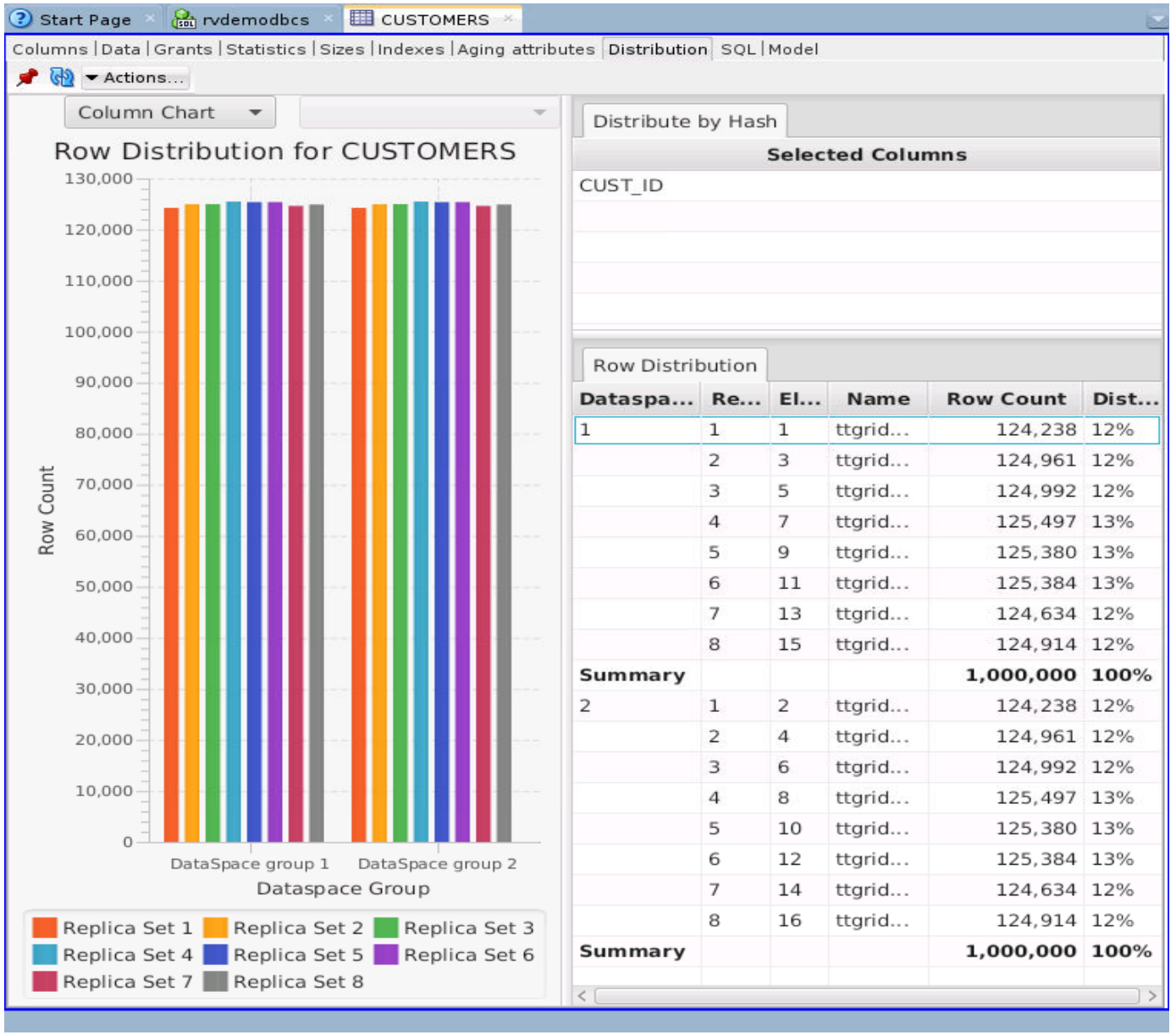
```
-- Database is in Oracle type mode
create table APPUSER.ACCOUNTS (
        ACCOUNT_ID      NUMBER(10) NOT NULL,
        PHONE           VARCHAR2(16 BYTE) INLINE NOT NULL,
        ACCOUNT_TYPE    CHAR(1 BYTE) NOT NULL,
        STATUS          NUMBER(2) NOT NULL,
        CURRENT_BALANCE NUMBER(10,2) NOT NULL,
        PREV_BALANCE    NUMBER(10,2) NOT NULL,
        DATE_CREATED    DATE NOT NULL,
        CUST_ID         NUMBER(10) NOT NULL,
    primary key (ACCOUNT_ID),
    constraint FK_ACCT_STATUS foreign key (STATUS) references APPUSER.ACCOUNT_STATUS (STATUS),
    constraint FK_ACCT_TYPE foreign key (ACCOUNT_TYPE) references APPUSER.ACCOUNT_TYPE (TYPE),
    constraint FK_CUSTOMER foreign key (CUST_ID) references APPUSER.CUSTOMERS (CUST_ID))
    distribute by reference (FK_CUSTOMER);
```

ORACLE®

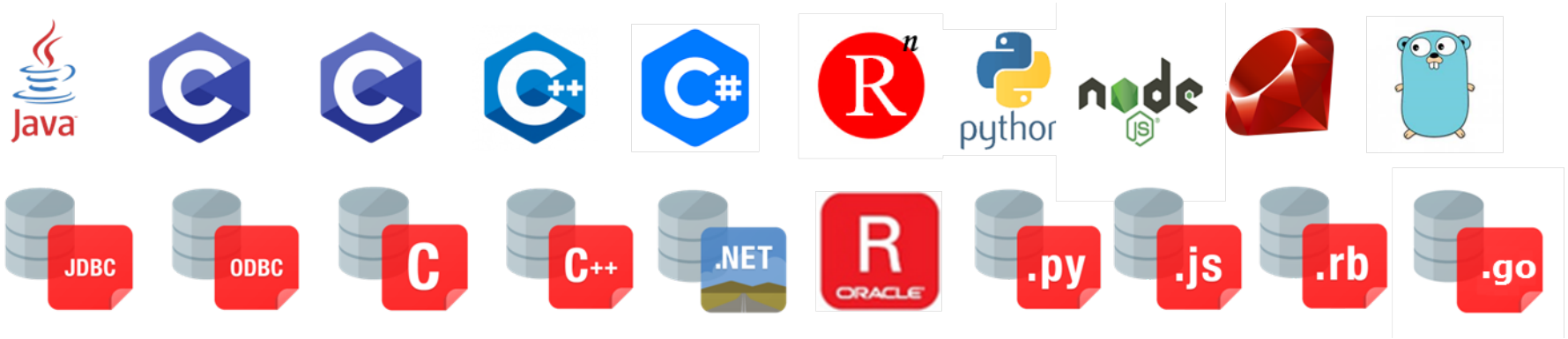
TimesTen Scaleout

ORACLE





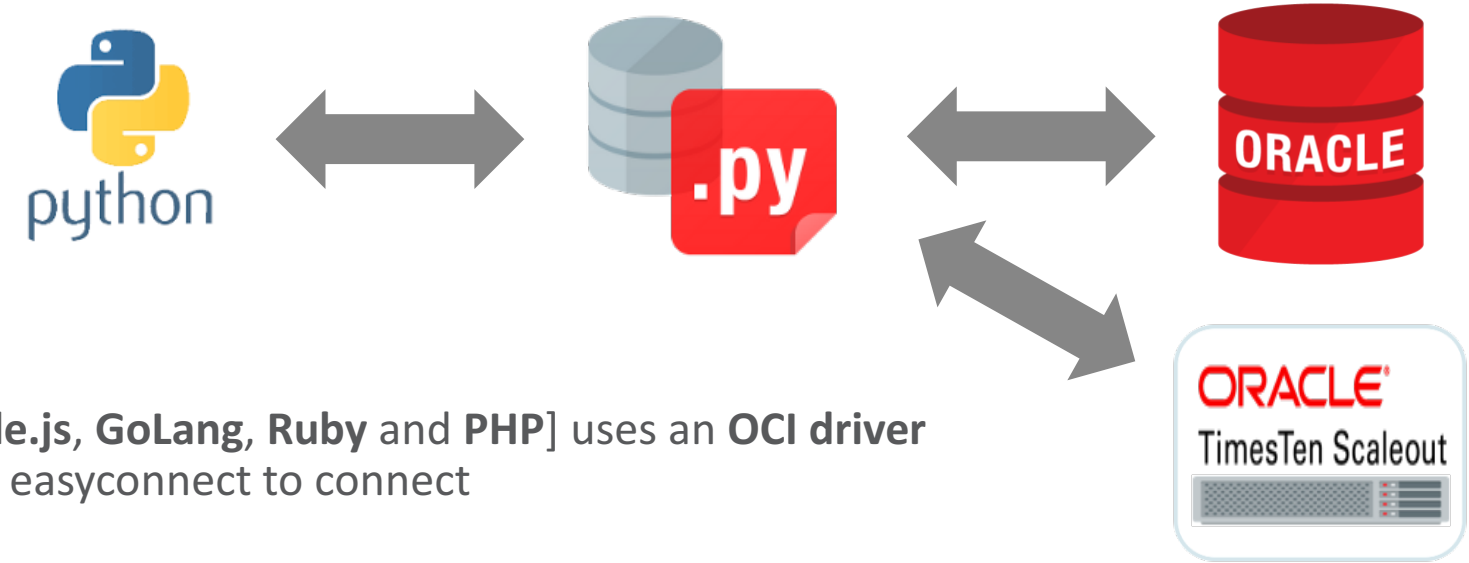
TimesTen Scaleout SQL APIs



API	Comment
JDBC	The same (JDBC 4.3)
ODBC	The same (ODBC 3.5.2)
OCI	The same (OCI 11.2.0.4.+)
R-Oracle	The same (OCI 11.2.0.4.+)
ODP.Net	The same (OCI 11.2.0.4.+)
PL/SQL	The same (11.2.0.4.+)
Python	The same (cx_Oracle, ODPI-C)
Ruby	The same (Ruby-ODPI, ODPI-C)
GoLang	The same (go-goracle, ODPI-C)

ORACLE

Using Oracle cx_Python with TimesTen Scaleout



Python [and **Node.js**, **GoLang**, **Ruby** and **PHP**] uses an **OCI driver**
Use tnsnames or easyconnect to connect

tnsnames.ora :

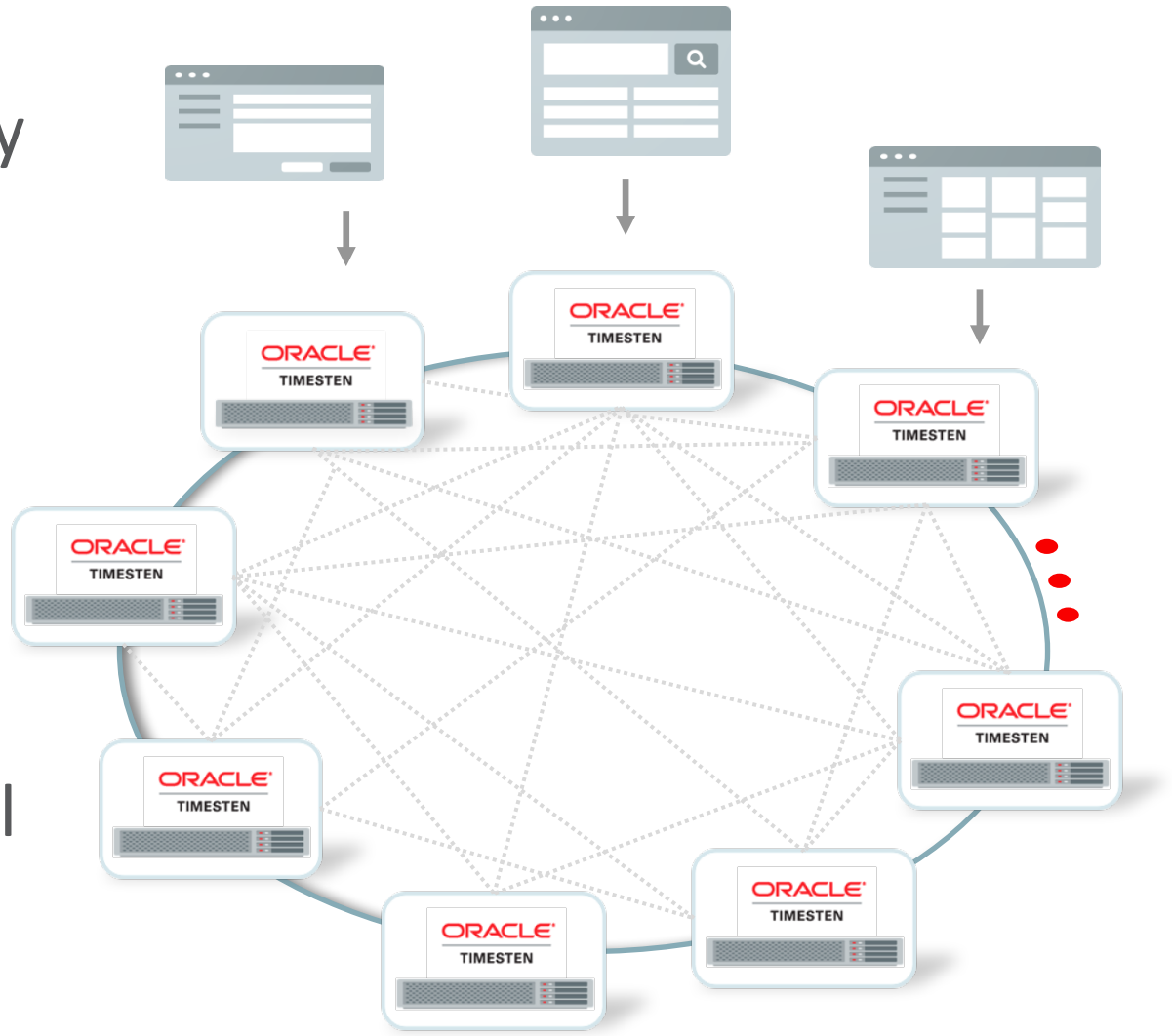
sampledb_1811 =(DESCRIPTION=(CONNECT_DATA = (SERVICE_NAME = **sampledb_1811**)(SERVER = **timesten_direct**)))
sampledbCS_1811 =(DESCRIPTION=(CONNECT_DATA = (SERVICE_NAME = **sampledbCS_1811**)(SERVER = **timesten_client**)))

TimesTen ODBC DSN

Client/Server or Direct Linked

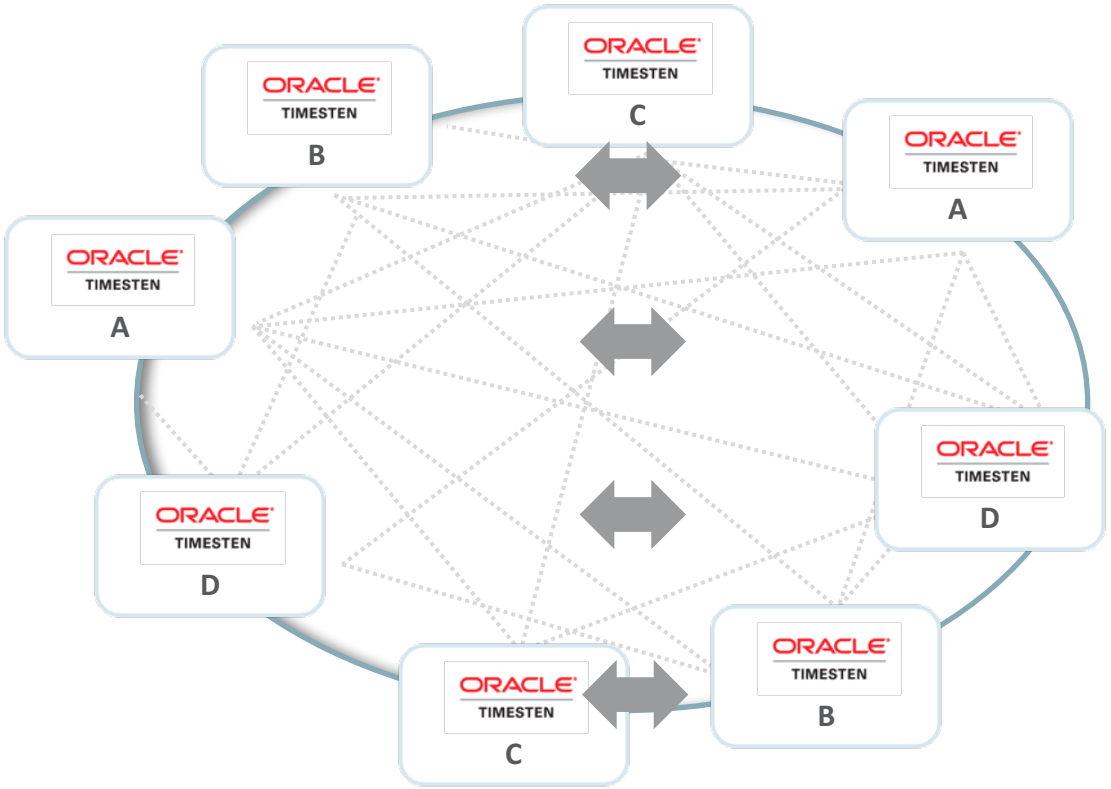
Single Database Image

- Database size not limited by memory
- Table data distributed across all elements
  - All elements are equal
- Connect to **any** element and access **all** data
  - Distributed queries, joins & transactions
- No need to de-normalize data model



High Availability and Maximum Throughput

K-Safety, All Active

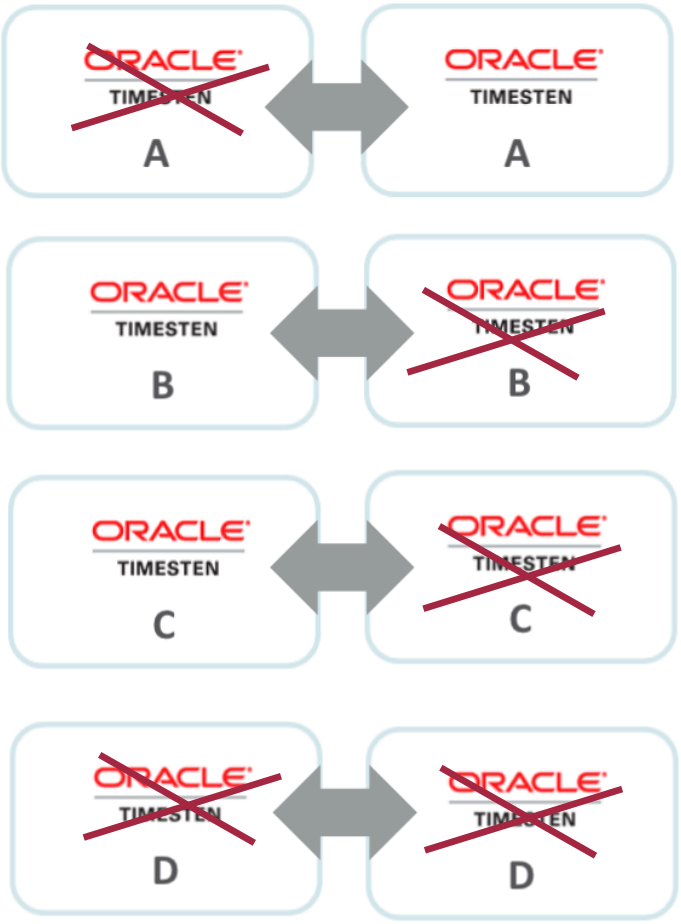


- Built-in HA via multiple copies of the data (K-safety)
  - Automatically kept in sync
- **All** replicas are **active** for **reads** and **writes**
  - Double the compute capacity
- Transactions can be initiated from and executed on any replica

Database Fault Tolerance – No Application Down Time

Provided one entire copy of the database is available

- If multiple elements fail, applications will continue provided there is one complete copy of the database
- Recovery after failure is automatic
- If an entire replica set is down, application can **explicitly** choose to accept partial results



TimesTen Scaleout

World's Fastest OLTP Database

China Mobile Use Case

China Mobile is the largest teleco by both market capitalization and subscribers

China Mobile has 902 Million subscribers
Each province does it own thing

All provinces use TimesTen in some form

- AIX, HP-UX or Linux
- Active Standby Pair
- Application Partitioning / Sharding
- TimesTen Scaleout

China Mobile Use Case

Chongqing Mobile is a small province

- 30 Million Subscribers

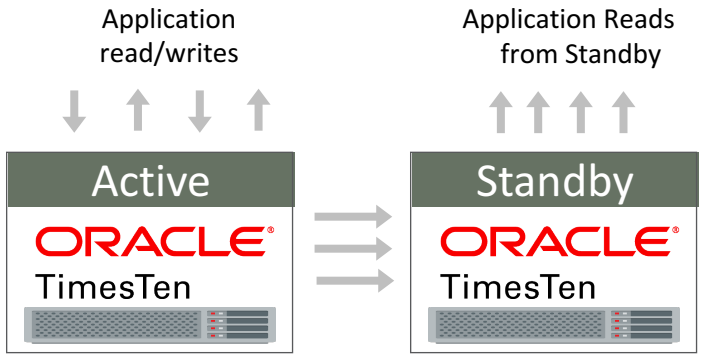
Their application is simple compared to others

- 800 tables
- 80% read, 5% update, 5% insert + DDL
- Location based services
- SMS, WeChat, billing etc
- Java Spring Framework
- SuSE Linux

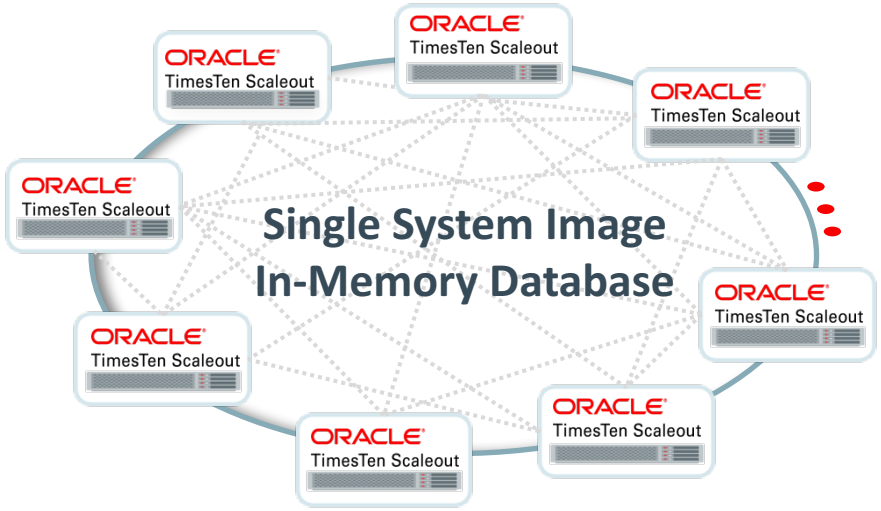
They needed more **read/write scalability**

China Mobile Use Case

TimesTen Classic



TimesTen Scaleout



Chongqing needed to scale their read/write workload

- They were a beta customer
- They migrated their app from TimesTen Classic to Scaleout
- Only two changes were needed
  - Change the app connect strings to point to the new servers
  - Change the table distribution clauses
- **No app changes, no data changes, no data model changes**

Chongqing Mobile Reference Quote

*As the **world's first TimesTen Scaleout PoC and go live customer**, our marketing service system was successfully deployed under the **new TimesTen Scaleout architecture with almost no application code changes**!*

*The entire system has not only **improved performance by more than three times**, but also successfully supported a number of new high concurrent business modules!*

*This fully demonstrates that **Oracle TimesTen Scaleout is an excellent distributed relational in-memory database product for OLTP SQL based applications!***

Head of Construction and Maintenance Department of Chongqing Mobile - Tang Tang

Available on OTN for download **NOW**

Oracle TimesTen Software Downloads

You must accept the **OTN License Agreement** to download this software.

◉ **Accept** License Agreement | ○ **Decline** License Agreement

Oracle TimesTen In-Memory Database 18c Release 1 NEW

Note: The initial release (18.1.1) supports only TimesTen Scaleout. TimesTen Classic support, including Application-Tier Database Cache, will be reintroduced soon in an upcoming patchset.

TimesTen 18.1.1.3.0 for Linux x86 (64-bit) (407,525,246 bytes)

Directions

- See the Release Notes and Prerequisites for system requirements.
- See the complete product documentation for Release 18.1.

TimesTen in On Premise

- TimesTen Velocity Scale requires :
  - Linux x8664 (glibc 2.12+)
    - Oracle Linux / Red Hat / CentOS 6.4+, 7+
    - Ubuntu 14.04+
    - SuSE 12+
  - JDK 8+
  - TCP/IP or IPoIB
  - A file system [eg ext4, not ext2 or ext3]
  - Enough RAM for the DB

TimesTen Scaleout on OCI, AWS, Azure, Google & OpenStack

1. Create your network, VMs, security configuration
2. Download TimesTen from OTN
3. Download Java
4. Download Apache Zookeeper
5. Unzip, configure and deploy

Amazon Linux

redhat

SUSE

ubuntu

CentOS

ORACLE LINUX

ORACLE TimesTen Scaleout

Java

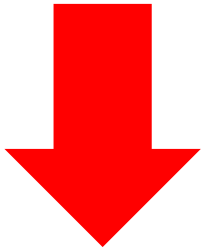
Apache Zookeeper

ZIP

TimesTen Scaleout with TerraForm *[available soon]*

1. Download provisioning script [101KB] & TimesTen 18.1.1.3.0 [389 MB]
2. ./ScaleOutRollOut.py
   ==> Provide Region, Compartment, Shape, OS, Number of Instances

Terraform



ORACLE
LINUX

ORACLE
TimesTen Scaleout

Apache
Zookeeper

YCSB Driver using the JDBC Sharding API

```
@Override
public Status read(String tableName, String key, Set<String> fields, Map<String, ByteIterator> result) {
  try {
    StatementType type = new StatementType(StatementType.Type.READ, tableName, 1, "", getShardIndexByKey(key));
    PreparedStatement readStatement = cachedStatements.get(type);
    if (readStatement == null) {
      readStatement = createAndCacheReadStatement(type, key);
    }
    readStatement.setString(1, key);
    ResultSet resultSet = readStatement.executeQuery();
    if (!resultSet.next()) {
      resultSet.close();
      return Status.NOT_FOUND;
    }
    if (result != null && fields != null) {
      for (String field : fields) {
        String value = resultSet.getString(field);
        result.put(field, new StringByteIterator(value));
      }
    }
    resultSet.close();
    return Status.OK;
  } catch (SQLException e) {
    System.err.println("Error in processing read of table " + tableName + ": " + e);
    return Status.ERROR;
  }
}
```

YCSB Driver using the JDBC Sharding API

```
private PreparedStatement createAndCacheReadStatement(StatementType readType, String key)
    throws SQLException {
  String read = dbFlavor.createReadStatement(readType, key);
  PreparedStatement readStatement = getShardConnectionByKey(key).prepareStatement(read);
  PreparedStatement stmt = cachedStatements.putIfAbsent(readType, readStatement);
  if (stmt == null) {
    return readStatement;
  }
  return stmt;
}
```

YCSB Driver using the JDBC Sharding API

```
/**
 * For the given key, returns Connection object that holds connection to the
 * shard that contains this key.
 *
 * @param key Data key to get information for
 * @return Connection object
 */
private Connection getShardConnectionByKey(String key) {
  return conns.get(getShardIndexByKey(key));
}
```

YCSB Driver using the JDBC Sharding API

```
/**
 * For the given key, returns what shard contains data for this key.
 *
 * @param key Data key to do operation on
 * @return Shard index
 */
private int getShardIndexByKey(String key) {
  int ret = 0;
  if (csRouting) {
    try {
      TimesTenDistributionKey dk = ttds.createTimesTenDistributionKeyBuilder().subkey(key, Types.CHAR).build();
      short[] prospects = dk.getElementIDs();
      ret = (int)prospects[dataSpace] - 1; // zero-based
    } catch (SQLException e) {
      System.err.println("Error in getShardIndexByKey: " + e);
    }
  } else {
    ret = Math.abs(key.hashCode()) % conns.size();
  }
  return ret;
}
```

YCSB Driver using the JDBC Sharding API

```
@Override
public String createReadStatement(StatementType readType, String key) {
  StringBuilder read = new StringBuilder("SELECT * FROM ");
  read.append(readType.getTableName());
  read.append(" WHERE ");
  if (distkey > 0) {
    /* hard-code the key in SQL to avoid extra binds */
    read.append("DIST_KEY = " + distkey + " AND ");
  }
  read.append(JdbcDBClient.PRIMARY_KEY);
  read.append(" = ?");
  return read.toString();
}
```

YCSB Driver using the JDBC Sharding API

```
@Override
public String createUpdateStatement(StatementType updateType, String key) {
  String[] fieldKeys = updateType.getFieldString().split(",");
  StringBuilder update = new StringBuilder("UPDATE ");
  if (cos) {
    update.append("/*+ TT_COMMITDMLONSUCCESS(1) */ ");
  }
  update.append(updateType.getTableName());
  update.append(" SET ");
  for (int i = 0; i < fieldKeys.length; i++) {
    update.append(fieldKeys[i]);
    update.append("=?");
    if (i < fieldKeys.length - 1) {
      update.append(", ");
    }
  }
  update.append(" WHERE ");
  if (distkey > 0) {
    update.append("DIST_KEY = " + distkey + " AND ");
  }
  update.append(JdbcDBClient.PRIMARY_KEY);
  update.append(" = ?");
  return update.toString();
}
```

YCSB Driver using the JDBC Sharding API

```
-- Create the user table with 10 fields.
-- Number of hash pages needs to be adjusted as:
-- (Nubmer of rows per element) / 256
-- e.g. 10M rows/elem => 39062 pages
CREATE TABLE usertable(
    DIST_KEY TT_SMALLINT NOT NULL,
    YCSB_KEY CHAR (24) NOT NULL,
    FIELD0 CHAR(100) NOT NULL,
    FIELD1 CHAR(100) NOT NULL, FIELD2 CHAR(100) NOT NULL,
    FIELD3 CHAR(100) NOT NULL, FIELD4 CHAR(100) NOT NULL,
    FIELD5 CHAR(100) NOT NULL, FIELD6 CHAR(100) NOT NULL,
    FIELD7 CHAR(100) NOT NULL, FIELD8 CHAR(100) NOT NULL,
    FIELD9 CHAR(100) NOT NULL,
    PRIMARY KEY (DIST_KEY, YCSB_KEY)
)
UNIQUE HASH ON (DIST_KEY, YCSB_KEY) PAGES = 39062
DISTRIBUTE BY HASH (DIST_KEY);
```