Oracle NoSQL Database – A Distributed Key-Value Store

Charles Lamb

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.

Agenda

- Oracle and NoSQL
- Oracle NoSQL Database Architecture
- Oracle NoSQL Database Technical Details

What Does NoSQL Mean To Oracle?

- Distributed
- Large data (Terabyte – Petabyte range)
- Two categories
  - OLTP ← Our focus is here
  - BI (M/R & Hadoop) ← … and we integrate here
- Data Models
  - Key-Value ← Our focus is here
  - Document
  - Columnar
- Berkeley DB by itself is not really “NoSQL” by this definition

Target Use Cases

- Large schema-less data repositories
  - Web applications (click-through capture)
  - Online retail
  - Sensor/statistics/network capture (factory automation for example)
  - Backup services for mobile devices
  - Scalable authentication
  - Personalization
  - Social Networks

ORACLE

Design Requirements

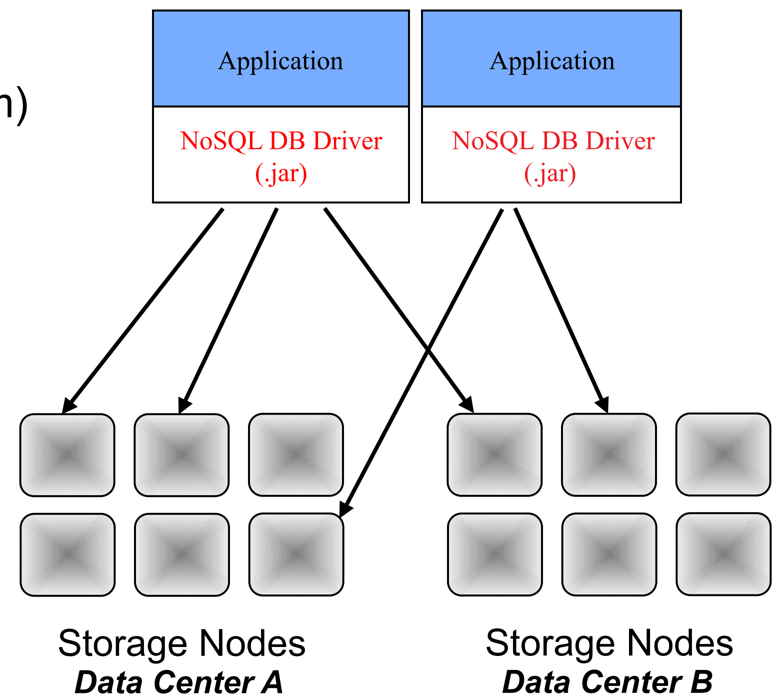
- Terabytes to Petabytes of data
- 10K's to 1M's ops/sec
- No single point of failure
- Elastic scalability on commodity hardware
- Fast, *predictable* response time to simple queries
- Flexible ACID transactions, a la Berkeley DB
- Unstructured or semi-structured data, with clustering capability
- Simple administration, enterprise support
- **Commercial-grade** NoSQL solution

Agenda

- Oracle and NoSQL
- Oracle NoSQL Database Architecture
- Oracle NoSQL Database Technical Details

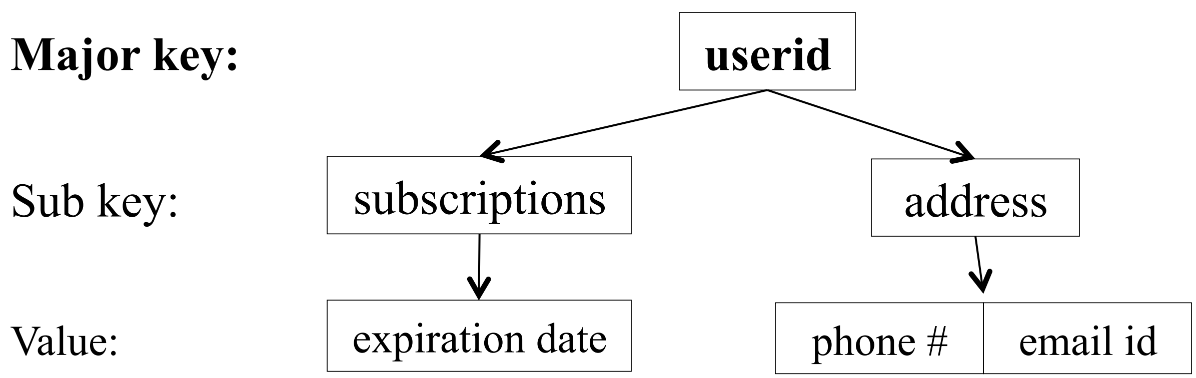
Oracle NoSQL Database Overview
A Distributed, Scalable Key-Value Database

- Simple Data Model
  - Key-value pair (with major/minor key paradigm)
  - CRUD + iteration
- Scalability
  - Data partitioning and distribution
  - Optimized data access via intelligent driver
- High availability
  - One or more replicas
  - Resilient to partition master failures
  - No single point of failure
  - Disaster recovery through location of replicas
- Transparent load balancing
  - Reads from master or replicas
  - Driver is network topology & latency aware
- Elasticity (Release 2)
  - Online addition/removal of storage nodes and automatic data redistribution



Oracle NoSQL Database
What the Programmer Sees

- Simple data model – key-value pair (major/minor key paradigm)
- Simple operations – CRUD, RMW (CAS), iteration
  - Conflict resolution not required
- ACID transactions for records within a major key, single API call
- Unordered scan of all data (non-transactional)
- Ordered iteration across sub keys within a key
- Consistency (read) and durability (write) spec'd per operation



Oracle NoSQL Database
No Single Point of Failure

- App Servers can be replicated by user
- Nodes are kept current (underlying BDB-JE HA technology)
- Driver (Request Dispatcher) is linked into each App Server
- Nodes may live in multiple Data Centers
- Node Replacement
  - Graceful degration until
    - Node is fixed, or …
    - … node is replaced
  - Automatic recovery

Oracle NoSQL Database
Easy Management

- Web-based console and CLI commands
- Manages and Monitors
  - Topology
    - Configuration changes
  - Load: Number of operations, data size
  - Performance: Latency, throughput. Min, max, average, trailing, …
  - Events: Failover, recovery, load distribution
  - Alerts: Failure, poor performance, …
  - We couldn't develop the product without this stuff

Agenda

- Oracle and NoSQL
- Oracle NoSQL Database Architecture
- Oracle NoSQL Database Technical Details

Oracle NoSQL Database Building Blocks

Berkeley DB Java Edition

- Robust storage for a distributed key-value database
  - ACID transactions
  - Persistence
  - High availability
  - High throughput
  - Large capacity
  - Simple administration
- Already used in
  - Amazon Dynamo
  - Voldemort (LinkedIn)
  - GenieDB

Oracle NoSQL Database
Building upon Berkeley DB Java Edition

- Data Distribution
- Dynamic Partitioning (aka “sharding”)
- Load Balancing
- Monitoring and Administration
- Predictable Latency
- Multi-Node Backup
- One-stop Support for entire stack
- Optimized Hardware (Oracle Big Data Appliance)

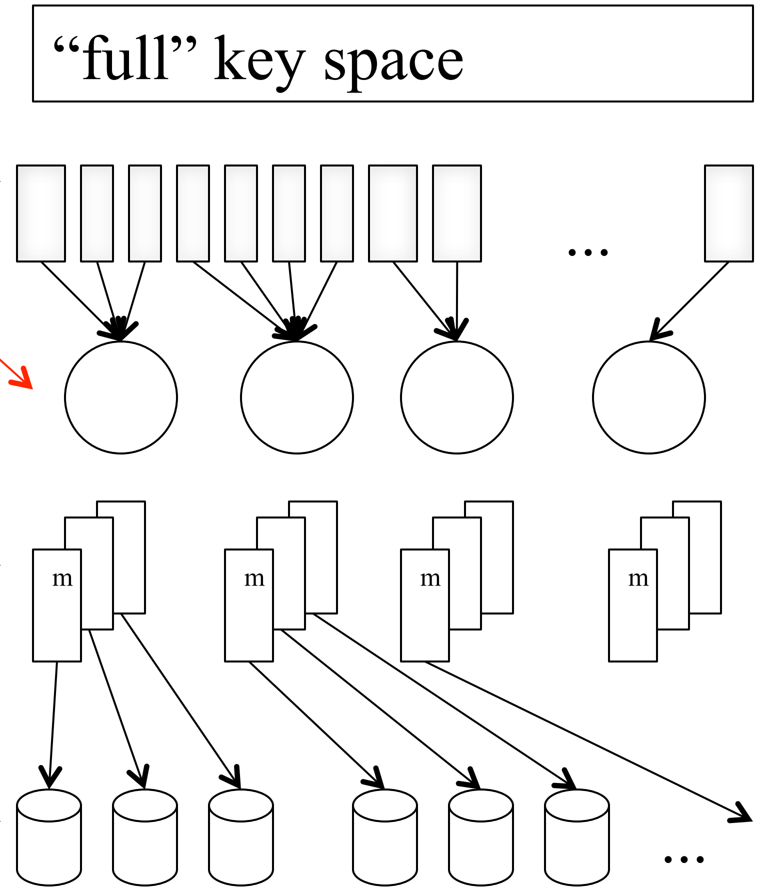
Distributed Data Storage Network Traffic

Minimizing latency is key

- Design Target: High volume -- 100K-1M ops/second
- Many concurrent threads
- Driver is topology aware
  - Directs operations to the proper node
  - Performs load balancing
  - Single network hop except when node fails or topology changes
  - Topology changes returned with results
- Single and multi-record operations
- Common use case will access single key-value pair or related set of sub-keys

Oracle NoSQL Database
Storage Terminology

- Key space hashes into multiple hash buckets (**partitions**)
- Set of partitions maps to a **replication group** (logical container for subset of data)
- Set of **replication nodes** for each replication group provides HA and read scalability for each replication group
- **Storage node** (physical or virtual machine) runs each replication node



Showing only a subset of the storage nodes

Oracle NoSQL Database Driver

Locating Data: Partition Map

- MD5(major key) % nPartitions → Partition ID
- Partition Map maps Partition ID to Rep Group

- Each Driver has a copy of the Partition Map
  - Initialized on the first request to any node
  - Allows direct contact with a node capable of servicing the request

Oracle NoSQL Database Driver

Locating Data: Rep Node State Table

- Rep Node State Table (RNST) locates optimum Rep Node within a Rep Group to handle a request
- Partition ID/Operation Type/Consistency →

  Rep Node in Rep Group
- RNST Contains …
  - Rep Node in Rep Group that is currently the Master
  - VLSN lag at a replica (how far behind a replica is)
  - RNST Entry's last update time
  - Number outstanding requests
  - Trailing average time associated with a request
  - Allows for varying topology and response times as seen from the drivers
- RNST updated by responses

Resolving a Request

- Hash Major Key to determine Partition id
- Use Partition Map to map Partition id to Rep Group
- Use State Table to determine eligible Rep Node(s) within Rep Group
- Use Load Balancer to select best eligible Rep Node
- Contact Rep Node directly
- Rep Node handles (almost always) or forwards request (almost never)
  - Operation result + new Partition Map/RNST information returned to client

ORACLE

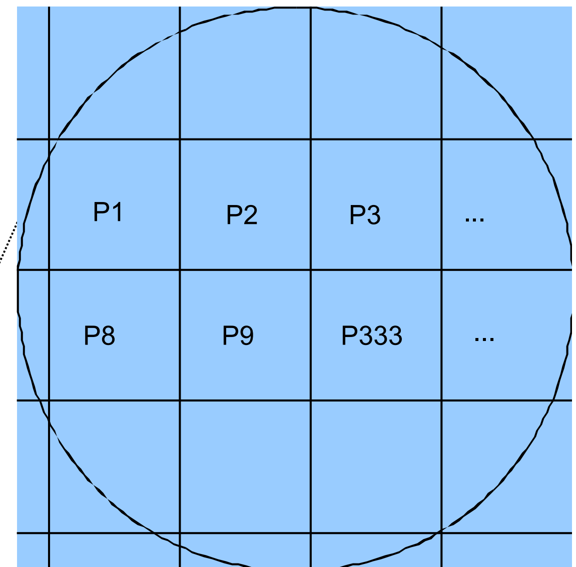
Oracle NoSQL Database
Storage Nodes

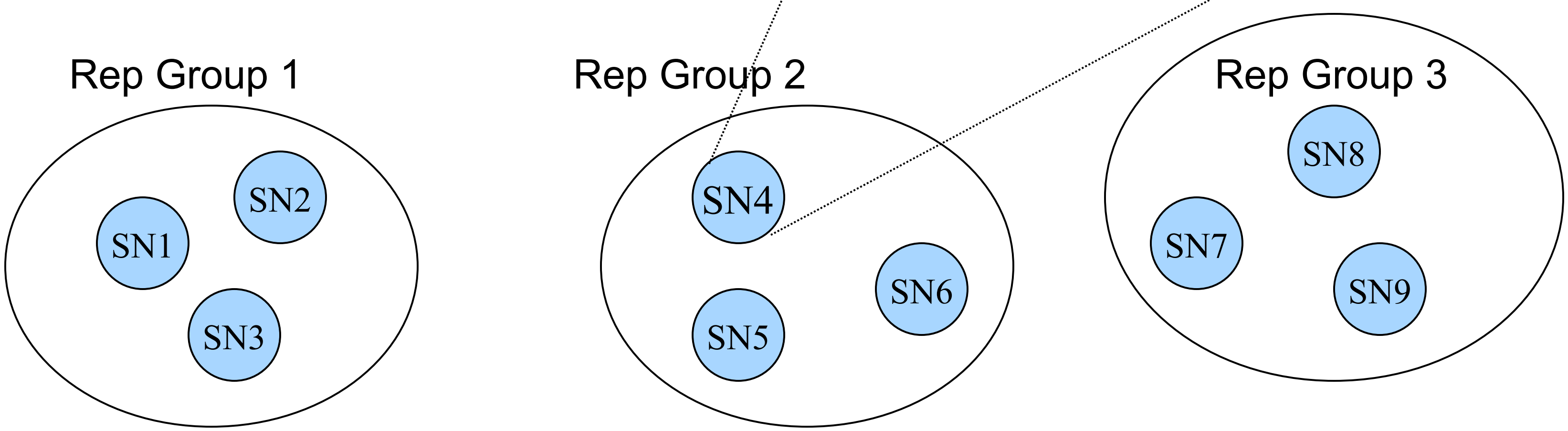
- A “machine” with its own local storage and IP address
  - Physical (preferred) or virtual
- Additional Storage Nodes increase throughput, capacity, or decrease latency
- Not necessarily symmetrical
  - Different processors, memory, capacity
- May host one or more Replication Nodes
  - By default, Replication Nodes are 1:1 with Storage Nodes
  - It would be unwise to locate multiple Replication Nodes in the same Replication Group on the same Storage Node

Oracle NoSQL Database

Topology Example

- 100M keys, 9 Storage Nodes
  might be configured as
  - 1000 Partitions
  - 3 Rep Groups
  - Replication Factor 3
  - 9 Rep/Storage Nodes





High Availability/Replication
Overview

- Replication group consists of single master and multiple replicas
- Logical replication
- Dynamic group membership, failover, elections
- Synchronous or Asynchronous
- Transaction durability and consistency guarantees are specifiable “per operation”
- Uses TCP/IP and standard Java libraries
- Supports nodes with heterogeneous platform hardware/OS
- HA Monitor class tracks Replication Group state

High Availability/Replication
Failure and Recovery

- Nodes join a replication group
  - When quorum is reached, election is held resulting in Master + Replicas
- Node Failure
  - Failed nodes can be replaced from the group via the Administrative API
  - Rejoining nodes automatically synchronize with the master
  - Isolated nodes can still service reads as long as consistency guarantees are met
- Master Failover
  - Nodes detect failure via missing heartbeat or connection failure
  - Automatic election of new master, via a distributed two phase election algorithm (PAXOS)
  - For maximum availability, whenever feasible, elections are overlapped with the normal operation of a node
- System automatically maintains group membership and status

High Availability/Replication
Transaction Behavior

- Controlled by a Durability option
- Transactions on the master can fail if
  - Not enough nodes to support durability
  - Node transitions from Master to Replica

Durability (Writes)

- Specified on per-operation basis, default can be changed
- Durability consists of ...
  - Sync policy (Master and Replica)
    - Sync – force to disk
    - Write No Sync – force to OS
    - No Sync – write when convenient
  - Replica Ack Policy
    - All
    - Simple Majority
    - None

Consistency (Reads)

- Specified on per-operation basis, default can be changed
- Consistency
  - Absolute (read from Master)
  - Time-based
  - Version
  - None (read from any node)

CAP

- We focus on “simple_majority” replica ack policy
- Only one version maintained
- No user reconciliation of records
- We might truncate transactions
- We are probably more CA than AP

What We've Been Testing

- YCSB-based QA/benchmarking
  - Key ~= 10 bytes, Data = 1108 bytes
- Prefer Keys up to 10's of bytes, Data up to 100K
  - > 100KB data ok
- Configurations of 10-200 nodes
  - Typical Replication Factor of 3 (master + 2 replicas)
- Minimal I/O overhead
  - B+Tree fits in memory => one I/O per record read
  - Writes are buffered + log structured storage system == fast write throughput

Performance: “Nashua”

- Single Rep Group (3 nodes), Sun X4170, dual Xeon, X5670, 2.93 GHz (3.3 GHz turbo), 2 x 6 core, 2 threads/core (24 contexts), 72GB memory, 8 x 300GB 10k SAS in RAID-5
- 400M records
  - Inserts 15.3k/sec 9ms (95%’ile), 12ms (99%’ile)
- 500M records
  - 50/50 read/update, overall 6,542 ops/sec
  - Read latency: 7/32/58 ms (avg/95/99)
  - Update latency: 8/33/61 ms (avg/95/99)

Performance: “SLC 72”

- 20 Rep Group on 60 VMs on 12 physical servers
  - 2x6 Xeon, 96GB / physical machine
  - 2 core x 2 threads (4 contexts), 16GB per VM
- 100M records/Rep Group
  - Inserts 109k/sec latency: 5/12/60 ms (avg/95/99)
  - 50/50 read/update, overall 33k ops/sec
    - Read latency: 16/74/155 ms (avg/95/99)
    - Update latency: 19/79/171 ms (avg/95/99)
- Remember: these are VMs so we’re only testing scalability, not real performance

Performance: “Intel Dupont”

- 64 Rep Groups on 192 physical servers
  - 2x6 Xeon x5760 (2.93GHz/3.3GHz turbo), 24GB / machine
  - 300GB local disk/machine
- 2.1B total records (YCSB limitation)
  - 50/50 read/update, overall 101.4k ops/sec
    - Read latency: 10/21/73 ms (avg/95/99)
    - Update latency: 23/25/197(*) ms (avg/95/99)
- Ext3 tuning
- (*) multiple YCSB clients may be considering same keys hot – under investigation

QUESTIONS?