Oracle NoSQL Database Compared to HBase

Before comparing NoSQL products

It is important to be aware that there are no standards in the NoSQL database technology space at this time. Each NoSQL product is implemented differently, sometimes very differently, often attempting to address different kinds of data management requirements and priorities. All of the NoSQL databases share certain characteristics, such as automatically sharded/partitioned data, horizontal data distribution, flexible schemas, high availability via replication and integration with Hadoop. However, each NoSQL database provides this functionality in different ways, and includes a variety of features primarily indicative of the technical problem set that they are trying to solve and their level of maturity. It is often best to understand the application’s technical requirements first and match that with a NoSQL database that best addresses those needs. In fact, most customers do not chose a single NoSQL database solution – they chose the solution that best addresses the problem being solved – and commonly end up using more than one NoSQL database.

That said, it is important to understand how to compare NoSQL databases, especially when much of the higher level capabilities and some of the lower level technical features and functionality overlap. I would caution the reader not to overly focus on the feature-by-feature comparison which changes rapidly over time especially in this rapidly evolving technology space, but to focus on the fundamental architecture, efficiency, ease of management and integration since these aspects will have a much longer term impact on the overall data management capabilities that are available to the application.

Overview – Oracle NoSQL Database

- The Oracle NoSQL Database is a horizontally scalable key-value database with multiple higher level data abstractions which support managing data as Binary key-value pairs, JSON objects, SQL-like Tables or as Graphs (using the Oracle Spatial and Graph package). The highly flexible key-value storage model combined with the higher level abstractions that make application development easier, allow the application developer to choose the data model that best fits their needs. ¹

- Oracle NoSQL Database is designed to provide extreme scale OLTP type storage and retrieval for simple key-value and hierarchical data structures. The system allows efficient storage of logically and physically co-located hierarchical data relations ² that can be queried, but like other NoSQL solutions Oracle NoSQL does not support system wide JOINs. Data is stored in the local file system using a set of write-once log files. Data storage provides flexible durability on a per operation basis, ranging from cache-based eventual consistency to proper ACID transactions ³. Data retrieval is by primary key and/or secondary indexes. Queries can support range-based predicates, as well as system-wide ordered results. Query operations are done in parallel and provide flexible data consistency guarantees specified by the application. ⁴
Oracle NoSQL Database secondary indices are implemented using distributed, shard-local B-trees. This implementation provides highly scalable, low-latency transactionally consistent secondary indices as well as parallelized secondary index search. Additionally, Oracle NoSQL Database supports secondary indexing over simple, scalar as well as over non-scalar and nested data values.

Oracle NoSQL Database uses a distributed, shared-nothing architecture which scales data storage and processing horizontally across commodity servers using a hashing algorithm and intelligent client drivers. Oracle NoSQL Database uses a PAXOS leader as the coordinator for data replication, based on the replication configuration for the system. The system supports defining Zones of Availability (akin to “data centers”), including Primary and Secondary zones. Each Availability Zone contains one or more copies of all of the data managed by the system. Replication between Availability Zones is automatically enforced to ensure that the data sets are kept up to date.

The Oracle NoSQL Database unique intelligent client driver design includes cluster topology, cluster status and performance, as well as automated routing information. This provides automated topology management, out-of-the-box data distribution and cluster load balancing of query requests.

Oracle NoSQL Database is not an island of data management technology – it is part of an integrated spectrum of data management technologies. Customers need to integrate NoSQL data management with other innovative technology options AND with their existing Oracle data management and application infrastructure. Oracle NoSQL Database is integrated with crucial Big Data open source technologies like Hadoop, MapReduce, Spark, Hive, etc., as well as with Oracle technologies like Oracle Database, Coherence, Event Processing, Real Time Decisions, Graph and Spatial and Big Data SQL. Additionally, NoSQL DB can be found on Oracle Engineered Systems such as the Oracle Big Data Appliance. This is a critical distinction for Oracle NoSQL Database. HBase is focused on integration with the emerging Big Data open source technologies, but largely leaves integration with exiting data management infrastructure as an exercise for the user. Oracle NoSQL Database integrates with both environments.

Unlike many other NoSQL products which rely on newly-minted storage, transaction and recovery technology, Oracle NoSQL Database is based on proven, reliable, mission-critical data storage technology – namely Berkeley DB. Berkeley DB is used in millions of production installations and provides the data storage, transactions, recovery, indexing and replication technology for Oracle NoSQL.

**Overview - HBase**

HBase is a key-value store that supports a single data abstraction known as table-structure (popularly referred to as column family). It is based on the Google Big Table design. HBase is designed to work on top of the HDFS (Hadoop Distributed File System). HBase accesses HDFS storage blocks directly and storing a natively managed file type. The physical storage is similar
to a column oriented database and as such works particularly well for queries involving aggregations, similar to the shared nothing analytic databases AsterData, GreenPlum, etc.

- HBase uses a partitioned/sharded data and master-slave distribution architecture, where data is hashed and sent to a set of external master processes known as “Region Servers” each of which are responsible for managing a subset of the key space. The Region Servers write the data (thru several layers of indirection) to HDFS which handles data availability thru file system replication. The Region Servers also make the data available to one additional process which can serve read requests. The Region Servers and HDFS must be configured and managed separately from HBase and requires additional open source software components to be installed and configured, including Zookeeper.

- Some of the major challenges with HBase include:
  - Increased hardware requirements (primary processors and memory) due to the fact that it relies on a multiple processes running on special-purpose servers that are configured specifically for that use.
  - More complex configuration and management due to the fact that it relies on separate configuration and management of multiple open source packages in order to provide the basic NoSQL functionality.
  - More complex troubleshooting and system performance management due to the number of sheer number of packages and configurations required.
  - Less performance and throughput for record-based operations.
**Comparison**

The table below gives a high level comparison of Oracle NoSQL Database and HBase.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Oracle NoSQL Database</th>
<th>HBase</th>
<th>Impact</th>
</tr>
</thead>
</table>
| Configuration and Administration| Self-contained, centralized, packaged configuration & administration                  | Requires Hadoop, HDFS, Zookeeper and other open source packages       | NoSQL DB is much simpler to configure, deploy, manage and troubleshoot | ➔ Lower risk  
 ➔ Lower Cost of Ownership (Less HW required + Lower administrative burden)  
 ➔ Better manageability of large clusters |
|                                | Individual administration of required packages                                         | Individual administration of required packages                        | NoSQL DB has better performance, is simpler to manage                  | ➔ Lower Cost of Ownership (Less HW required + Lower administrative burden + Simpler to troubleshoot and tune) |
| Storage Model                  | Self-managed, local files using log-based file system, optimized for high record-based read/write throughput | Uses HDFS (Hadoop Distributed File System)                           | NoSQL DB has better performance, is simpler to manage                  | ➔ Lower Cost of Ownership (Less HW required + Lower administrative burden + Simpler to troubleshoot and tune) |
|                                | HDFS is optimized for large block I/O, not record-based I/O required by OLTP NoSQL applications | Requires separate administration                                     | NoSQL DB has better performance, is simpler to manage                  | ➔ Lower Cost of Ownership (Less HW required + Lower administrative burden + Simpler to troubleshoot and tune) |
|                                | Driver dynamically adjusts to changing topology and throughput.                       | HDFS is optimized for large block I/O, not record-based I/O required by OLTP NoSQL applications | NoSQL DB has better performance, is simpler to manage                  | ➔ Lower Cost of Ownership (Less HW required + Lower administrative burden + Simpler to troubleshoot and tune) |
| Data Access                    | Single-hop data access. Built-in topology and latency-aware client driver maps data to storage location(s). Operations sent directly from the client application to the appropriate storage node. Driver dynamically adjusts to changing topology and throughput. | Multi-hop data access. Separately managed Name and Region servers map data to storage location(s). Operations funneled through one or more servers to appropriate HDFS storage. | NoSQL DB has better performance because it requires only a single hop to access data, requires no additional management | ➔ Lower risk  
 ➔ Lower Cost of Ownership (Less HW required + Lower administrative burden)  |
<table>
<thead>
<tr>
<th>Scale Out</th>
<th>Partitions, grouped into shards Built-in, re-distribution of partitions in background task as hardware is added, optional operator-invoked re-balancing available Simple scale out due to shared-nothing client-server topology – simply add more nodes to scale</th>
<th>Shards, grouped into regions. Regions automatically split and redistribute growing data (significant performance issue) Regions, crash recovery and scaling require operator/DBA intervention Multi-server, multi-region complex topology</th>
<th>NoSQL DB is simpler to configure, expand and manage, designed for consistent, predictable throughput. Performance and management differences more significant as the cluster grows. ➔ More predictable performance ➔ Lower Cost of Ownership (Less HW required + Lower administrative burden) ➔ Better manageability of large clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datacenters (Availability Zones)</td>
<td>Primary and Secondary Data Centers. Built-in synchronous and asynchronous replication between DCs. Reads can be DC-specific.</td>
<td>HBase Regions can be replicated across DCs. Requires separate configuration &amp; management.</td>
<td>NoSQL DB is simpler to configure and has more application options ➔ Shorter time to market ➔ Lower Cost of Ownership (Lower administrative burden)</td>
</tr>
<tr>
<td>Replication</td>
<td>Built-in, configurable, integrated with transaction sub-system Basis for HA and scalability Uses Berkeley DB – 20+ years of field validation</td>
<td>Combines HDFS in-cluster replication and internal intra-cluster replication which is configured and managed separately HBase has been an Apache project for 6 years</td>
<td>NoSQL DB replication is easier to manage, more automatic and integrated. NoSQL is more mature based on 20+ years of field validation ➔ Lower Risk ➔ Lower Cost of Ownership (Lower administrative burden)</td>
</tr>
<tr>
<td>Integration</td>
<td>Integrated with Big Data technology stack. Integrated with Oracle technology.</td>
<td>Integrated with Big Data technology stack only.</td>
<td>NoSQL DB integrates with new IT projects as well as existing IT infrastructure ➔ Shorter time to market ➔ Lower Cost of Ownership (Lower implementation cost)</td>
</tr>
<tr>
<td>Transactions and Concurrency</td>
<td>Configurable ACID and BASE transactions</td>
<td>Strongly consistent reads &amp; writes</td>
<td>NoSQL DB has more configuration options ➔ Better fit for some applications ➔ Shorter time to market</td>
</tr>
<tr>
<td>Data Model</td>
<td>Support for Key-Value Pairs, JSON objects, Tables, Graph Data</td>
<td>Column-family Tables.</td>
<td>NoSQL DB has more developer options ➔ Shorter time to market¹</td>
</tr>
<tr>
<td>APIs</td>
<td>APIs for Java, C, REST, Thrift APIs for JavaScript, Python, C# planned for early 2015</td>
<td>APIs for Java, Jython, Groovy, REST, Thrift Many open source APIs</td>
<td>Similar API choices, better API support for NoSQL¹0 ➔ Lower risk ➔ Enterprise-class Support</td>
</tr>
<tr>
<td>Monitoring and Administration</td>
<td>Web-based monitoring, Command Line Interface. Integrated with Oracle Enterprise Manager. JMX and SNMP interface supports 3rd party plug-ins.</td>
<td>Command Line Interface. Open Source community-supported graphical tools.</td>
<td>NoSQL DB leverages OEM integration ➔ Enterprise-class Support ➔ Lower Cost of Ownership (Lower administrative burden because of OEM skills re-use)</td>
</tr>
</tbody>
</table>
**Oracle NoSQL Database Proof Points**

**NTT Docomo**

NTT Docomo uses Oracle NoSQL DB to provide a Digital Marketplace for millions of their smart phone users. The DM provides web-based DRM information, configuration management and product recommendations on a per-user/per-device basis. NTT benchmarked their application against Oracle NoSQL DB and HBase to assess the long term cost of ownership, based on the hardware required to sustain the target throughput (transactions per second). They determined that Oracle NoSQL DB would be much more cost effective because it required less hardware. See the use case [here](#).

![Performance Comparison](image)

1. Oracle NoSQL Database with SSDs achieves **26000 tps** with **3 nodes**.
2. Compared to HBase, Oracle NoSQL Database achieves **high performance goals with much smaller number of nodes**.
3. Considering high performance goals combined with additional future requirements, the total cost of Oracle NoSQL Database is estimated to be much lower than HBase.

Over 20,000 tps can be handled with just 3 nodes.
Internal YCSB Benchmarks

Oracle Engineering ran an internal benchmark, comparing Oracle NoSQL DB against HBase using YCSB. Tests were run on the Oracle Big Data Appliance using a 6 and 12 node cluster. YCSB is an OLTP-style, record-based application benchmark. Oracle NoSQL Database demonstrated better query performance and lower latency as well as more predictable scalability as the size of the cluster grew.

HBase vs Oracle NoSQL Database
YCSB Benchmark Results

NoSQL DB Highlights

- Much better READ Performance
- Lower READ Latency
- Stable Performance
- Linear Scalability

Setup
- Replication Factor (RF) = 3
- Each BDA node has 12 Disk

95/5 Read/Update Throughput

Throughput (ops/sec)

6 (672M) 12 (1344M)

BDA Nodes (record count)

Average Latency (ms)

NoSQL Throughput (ops/sec) HBase Throughput (ops/sec)
NoSQL Read Latency (ms) Hbase Read Latency (ms)
Passoker was able to reduce their application Time-to-Market by 75% because they were able to save time on application development and QA due their use of the flexible data modeling options in Oracle NoSQL Database. More information available here: http://medianetwork.oracle.com/video/player/1972123662001

All of the records which share the same Shard key are co-located within the same Storage Node on disk. This supports ACID transactions because NoSQL DB can ensure that reads and writes within the same storage node are transactionally consistent. This also supports low-latency data access of related data records because all of the records can be managed by accessing a single storage node.

Oracle NoSQL Database supports both eventually consistent or BASE transactions as well as traditional ACID transactions. BASE transactions are often used by customers who want to increase data access throughput, but are willing to tolerate potentially inconsistent results or loss of data in the event of system failure. ACID transactions are often required by applications that can NOT tolerate inconsistent results or loss of data in the event of system failure. For many applications it is important to provide BOTH types of transactions, however most NoSQL systems support one other, but not both. HBase has a more limited range of transaction options than Oracle NoSQL DB.

Queries in Oracle NoSQL Database are automatically run in parallel across shards, when appropriate (table scans and secondary index searches for example). The application controls how many parallel threads are executed and the batch size of each thread. This is an important application performance tuning feature, as it allows the application to control the degree of parallelization executed for a given query, thereby controlling the impact of the execution on the overall system. For example, low latency interactive secondary index lookups (that will likely return few results) can be configured for maximum parallelization and batch size so that results are returned faster, where as higher latency report-style scans can be configured to reduce the impact on the system.

The intelligent NoSQL DB client driver is Availability Zone-aware. Queries can be specifically directed by the application to one or more Availability Zones. This can be used by the application to restrict reporting, statistical and batch processing-like queries to specific Availability Zone(s), thereby reducing their impact to throughput and performance on the production portion of the NoSQL cluster.

The intelligent NoSQL DB client driver has the advantage that it can automatically respond and adjust to changes in the NoSQL cluster topology. For example, if a given Storage Node fails (or is taken offline), the client driver is notified of the topology change and can automatically direct queries to the appropriate surviving Storage Nodes. Conversely, if additional Storage Nodes are added or the NoSQL cluster is rebalanced, the NoSQL DB driver can immediately start utilizing the newly available hardware (and throughput).

The NoSQL DB driver includes performance statistics for each Storage Node, allowing it to dynamically adjust query load balancing if a Storage Node(s) throughput starts to change due to other queries that may be accessing the affected Storage Node(s), increasing queries if the SN is more responsive or directing queries to alternate SNs if the SN has become less responsive.

HDFS as a file system, is tuned for large block I/O and as such is not very efficient at individual record read/write operations. Oracle NoSQL Database’s log-structured files stored in the local file system is tuned for high volume, low latency read and write operations on specific records. Customers have found that HBase works very well for batch oriented bulk read/write data management, but that Oracle NoSQL DB provides better per-record operational throughput. This can be verified using the Yahoo Cloud Services Benchmark (YCSB) and is validated by NTT Docomo in their Digital Marketplace use case mentioned earlier in the document.

This is distinctly different from Oracle NoSQL Database which allows the customer to determine how many read replicas are needed in order to meet their concurrency requirements. Additionally, Oracle NoSQL Database uses an integrated, optimized in-memory and process-based replication architecture for high availability rather than relying on the external file system (HDFS). Oracle NoSQL DB offers a more simplified, easily managed, more tightly integrated solution.

Although there are many HBase open source APIs available, support and committers can be spotty. Open source APIs run the risk that committers are not available or have other priorities. Oracle provides enterprise-class support for all of the supported APIs available on Oracle NoSQL DB.