Oracle NoSQL Database

The roots

Oracle NoSQL database is based on Berkeley DB Java Edition

Berkeley DB is a family of embeddable database products developed by *Sleepy Cat, Inc.* in early 1990s

Berkeley DB includes Berkeley DB based on C, Berkeley DB Java Edition and Berkeley DB XML

Berkeley DB has been acquired by Oracle in 2006

All Berkeley DB products provide all data management capabilities that can be expect from the traditional database system available as an embeddable database library

Embeddable database library means that all database management capabilities are built into an application instead of an application accessing a database server

Overview

Oracle NoSQL database leverage the features and functionality of Oracle Berkeley DB Java Edition

Oracle NoSQL database is *shared-nothing* database system designed to run and to scale on commodity hardware

Key-value pairs are partitioned across the groups of servers called as *shards*

At any point a single key-value pair is always associated with a unique *shard* in the system

Most of Oracle NoSQL database deployments use multiple machines (also called as *nodes*) per each *shard*

Overview

The recommended configuration requires a minimum 3 machines (nodes) per shard; this is also called as *replication factor*

For example a highly available 10 shard system with a replication factor 3 would be deployed on 30 nodes (machines)

At API level a programmer uses *key-value* paradigm

A *major key* as a component of key-value pair is used to determine which shard the key-value belongs to

The rest of a key is called as *minor key*

Overview

A combination of major key and minor key is used to address information stored in key-value pair

A minor key is optional

API can be used for accessing all contents of key-value pair or for accessing parts of a record identified by a major and minor key combination

When the amounts of data grow then it is possible to add new hardware resources

Availability

Availability is achieved by adding redundancy to the system; each shard consist of multiple nodes that contain replicas of identical copies of data

The modifications to a node are propagated to other nodes to keep them current

Monitoring tools are used to detect and to repair failures; if a node fails, the system automatically detects and handles the change in a membership of a shard

Data updates can be performed in a *single-master* or *multi-master* architectures

In a *single-master* architecture there cannot be concurrent changes to the same record on multiple replicas

In a *multi-master* architecture updates are allowed to any node

Eventual consistency

Oracle NoSQL database allows to choose the consistency level required on a per-operation basis

Absolute consistency is required if the most recent version of data item is required; then operation is performed on a master node in a shard

Time based consistency tolerates reading data that is more than one second out-of-date with respect to the most current update

Transaction ID-based consistency ensure that after update of a record subsequent read operations will read a version of the same record or at least as current as the changes made to that record

Eventual consistency

No consistency means that any node in a shard can be used to access a record

Durability

Oracle NoSQL database take advantage of the multiple replicas to ensure durability

An update can be made durable by propagating a change to one or more replicas concurrently

The system declares operation durable after receiving acknowledgment for the update for at least one replica

The system allows to choose durability policy on per-operation basis

Three independent dimensions of durability are supported

For a master node an application programmer can choose whether a change is durable when it is written to log buffer, when it is written to file system buffers, or when it is written to disk

Durability

Application programmer can also choose whether a change should be propagated to replicas asynchronously or synchronously with acknowledgment

When a change has been propagated to replicas then application programmer can choose whether a change is considered durable when it is written to log buffer, system file buffers or when it is writtent o disk on the replicas

Transactions

Oracle NoSQL database supports row-level locking and two-phase locking protocol

DML

API for manipulating key-value pairs includes `put()`, `get()`, and `delete()` operations

The operations `put()`, `get()`, and `delete()` act on a single (multicomponent) key

API can be used to iterate over the key-value pairs

Minor keys can be used to represent a structure of a record

Oracle NoSQL database also supports JSON schemas and Apache Avro schemas for specifying the structure of a value of key-value pair

Examples

Connect/disconnect to a data store

```
String storeName = "kvstore";
String hostName = "data-pc01.cs.uow.edu.au";
String hostPort = "5000";

KVStore store;
store = KVStoreFactory.getStore(
                        new KVStoreConfig(storeName,
                        hostName + ":" + hostPort));
… … …
store.close();
```


Examples

Write/read key-value pair

```
String keyString = "Key1";
String valueString = "Value1";
store.put(Key.createKey(keyString),
          Value.createValue(valueString.getBytes()));
… … …
ValueVersion valueVersion =
             store.get(Key.createKey(keyString));
System.out.println("Record <" + keyString + "," +
   new String(valueVersion.getValue().getValue()) + ">
                                          retrieved.");
```


Examples

Creating major and minor key

```
ArrayList<String> majorList = new ArrayList<String>();
ArrayList<String> minorList = new ArrayList<String>();

String valueString = "Value";

majorList.add("users");
majorList.add("007");
majorList.add("folders");

minorList.add("notepad");

Key myKey = Key.createKey(majorList, minorList);
Value myValue =
            Value.createValue(valueString.getBytes());
store.put(myKey, myValue);
```


Examples

Loading data

```
ArrayList<String> majorList = new ArrayList<String>();
ArrayList<String> minorList = new ArrayList<String>();
Key myKey;
Value myValue;
String valueString;

majorList.add("Bond");
minorList.add("James");
minorList.add("car");
minorList.add("PKR256");
valueString = "Toyota Corolla";
myKey = Key.createKey(majorList, minorList);
myValue = Value.createValue(valueString.getBytes());
store.put(myKey, myValue);
majorList.clear();
minorList.clear();
```


Examples

Searching with major key

```
ArrayList<String> majorList = new ArrayList<String>();
majorList.add("Bond");
Key myKey = Key.createKey(majorList);
Iterator<KeyValueVersion> it =
store.storeIterator(Direction.UNORDERED, 0, myKey,
                    null, null);
while (it.hasNext())
{
  KeyValueVersion kvvi = it.next();
  System.out.println(kvvi.getKey().toString() + "=>" +
            new String(kvvi.getValue().getValue() ) );
}
```


Examples

Searching with major and minor key

```
ArrayList<String> majorList = new ArrayList<String>();
ArrayList<String> minorList = new ArrayList<String>();
majorList.add("Bond");
Key myKey = Key.createKey(majorList,minorList);
Iterator<KeyValueVersion> it =
   store.multiGetIterator(Direction.FORWARD, 0, myKey, null,
null);
while (it.hasNext())
{
   KeyValueVersion kvvi = it.next();
   System.out.println(kvvi.getKey().toString() + "=>" +
                    new String(kvvi.getValue().getValue() )
);
}
```

References

Alam M., *Oracle NoSQL Database,* Oracle Press, 2014

`http://www.uow.edu.au/~jrg/235/HOMEWORK/`

12 How to process distributed database systems ?

`http://docs.oracle.com/cd/E26161_02/html/GettingStartedGuide/`

Getting started with Oracle NoSQL Database