CS510 Cloud and Cluster Database Management
Project Assignment #2

Matthew Pate, Timothy Jensen, Jonathan Jensen, Nicholas Hoover

1. *Data Model*
   a. *What data model described in class (Lect 2 - Data Models) is most like your system's data model?*

      The data model used by Riak is the key/value model.

   b. *List the objects in the data model (i.e. database, collection, field, etc.).*

      The data model consists of the following objects:
      - Buckets - define a virtual key-space and provide the ability to define isolated non-default configuration (compared to tables in Relational DBs)
      - Keys - binary values (or strings) used to identify objects
      - Objects - the only unit of storage in
      - Links - metadata that establishes one-way relationships between objects
      - Indices - secondary indices are used to tag objects at write time with one or more queryable values
      - Data Types - Flags, Registers, Counters, Sets, Maps

   c. *Give one examples of the data manipulation API such as a function to create, retrieve, update and delete data items.*

      An example of using the client library (Ruby) to make the connection, create a bucket, store a simple integer:

```
client = Riak::Client.new(:protocol => "pbc", :pb_port => 10017)
my_bucket = client.bucket("my_super_cool_bucket")
obj = my_bucket.new('one')
obj.data = 1
obj.store()
```

2. *Query Support*
   a. *What is the query language for your system?*

The query language for Riak varies but in its most basic form it uses client libraries to query objects by key. There are client libraries in Java, Ruby, Python, C#, NodeJS, Erlang, PHP, Go, and Haskel. Riak also has a rich, full-featured HTTP API.

With Riak 2.0, a Search module was incorporated that utilizes Lucene syntax.

b. *Can your system support range queries?*

Yes, Riak supports range queries. An example of a range query using the HTTP API is as follows:

```
curl -XPOST localhost:8098/mapred\
  -H "Content-Type: application/json" \
  -d @-<<EOF
{
  "inputs": {
    "bucket": "people",
    "index": "field2_bin",
    "start": "1002",
    "end": "1004"
  },
  "query": [
    {
      "reduce": {
        "language": "erlang",
        "module": "riak_kv_mapreduce",
        "function": "reduce_identity",
        "keep": true
      }
    }
  ]
}
EOF
```

An example of a range query using the Ruby client library is as follows:

```
bucket = client.bucket_type('indexes').bucket('people')
bucket.get_index('field2_int', 1002..1004)
```

c. *Can your system support navigation (graph) queries?*

Riak does not support Graph queries directly. It seems that the search module provided in Riak 2.0 when combined with with Solr can provide behavior similar to graph queries. Additionally, Riak has the concept of “links” which consists of meta data on objects allowing one way relationships between objects.

3. *Indexes*
    a. *Does your system support indices?*

    Yes, Riak supports indices. Riak allows tagging objects with one or more values at write-time that can later be queried. Additionally, Riak Search can be used which allows creating named indexes on the objects.

    b. *Are there any limitations on what can be indexed?*

    As Riak allows tagging objects with arbitrary tags at write time, there is technically no limit on what can be indexed. The only requirement on the index is that it is either a binary or a string.

4. *Transactions and Concurrency Control*
    a. *Does the system support consistency?*

    Yes. Riak does support consistency.

    b. *What type of consistency guarantees are supported?*

    Beginning in version 2.0 of Riak, you have the option of choosing between two separate consistency systems. Eventual Consistency and Strong Consistency. It should be noted that Strong Consistency is currently an open-source-only feature and isn't commercially supported.

    When using Strong consistency, you are guaranteed that you will never see inaccurate values, however, some operations may fail if not enough replicas are available.

    When using Eventual Consistency, a read may return an out-of-date value (in the occurrence of network or system failures), but with Eventual Consistency, read and writes will succeed even when clusters are experiencing poor performance.

c. *What technology is used to implement consistency?*

Riak implements Consistency through versioning and vector clocks. Riak tags all objects with versioning information, and vector clocks keep track of which objects have been modified and when. This allows Riak to resolve conflicts on its own by determining what the last modified value is.

5. *Scalability and Replication*

a. *Does the system support replication? If yes, briefly describe how the replication works?*

Riak supports replication. In fact, replication is automatic in Riak. All data stored in Riak is replicated to N number of nodes. By default N is set to 3, meaning that all data stored will be replicated across 3 different nodes. It should be noted that there is no guarantee that the replicas will be stored on three separate physical nodes, they could be partitions or virtual nodes.

Using the Apache 2.0 License, Riak can be used to replicate any nodes in a single cluster. Using the commercial extension enables Riak to be used for Multi-Datacenter deployment.

b. *Does the system support sharding? If yes, briefly describe how the sharding works.*

Riak does not support sharding. Instead, Riak evenly distributes data between all active nodes using consistent hashing. In Riak, consistent hashing is an automatic process. Riak claims that this method makes it significantly easier for companies to scale and supports long-term growth much better than sharding.

6. *Storage and Platform/Deployment*

a. *When data is stored in the system, where is the data usually stored?*

Riak data is usually stored on an SSD. In fact, it is recommended that the fastest SSD's possible are used to store Riak data, as well as possibly configuring RAID0 to further increase read/write speed.

RAM is also used to temporarily store data in order to increase performance even further. Riak will take advantage of all available RAM by holding as much data there as it can.

b. *Where is the system usually deployed?*

It is recommended that Riak is not run in the cloud or on a virtual cluster. This is to maximize performance. However, Riak supports all infrastructures including: Debian/Ubuntu, RHEL/CentOS, Mac OS X, FreeBSD, SmartOS, Solaris 10, SUSE, Windows Azure, and AWS.

If you do choose to run Riak in a cloud environment, it is suggested that you choose the largest VM's possible and deploy VM's within the same region. You should also always use the highest core processors available. Riak is written in Erlang, which is a functional programming language. This means that performance scales with the amount of cores/nodes available.

c. *What programming languages does the system support?*

Riak officially supports the following languages:
Java, Ruby, Python, C#, Node.js, PHP, Erlang, Go.

In Addition to this, there are libraries written for the following languages that are not officially supported but are being used: C/C++, Clojure, ColdFusion, Common Lisp, Dart, Django, Erlang, Grails, Griffon, Groovy, Haskell, OCaml, Perl, Racket, Scala, Smalltalk.

References

http://docs.basho.com/riak/latest/