BEFORE WE BEGIN...

• NOSQL: It is mechanism for storage & retrieval of data using means other than tabular relations.

• KEY – VALUE: A data storage paradigm that uses (key, value) pairs to store, retrieve and manage data.
BEFORE WE BEGIN...

- DISTRIBUTED DATABASES: A setup in which databases are hosted at different locations and are interconnected through a computer network.

- EVENTUAL CONSISTENCY: A consistency model which informally guarantees that, if no new updates are made to a given data item, eventually all accesses to that item will return the last updated value.
Riak KV is a **distributed, NoSQL, key-value database** with advanced local and multi-cluster replication.
SHORT BIO OF RIAK

• Developed by Basho Technologies

• Open source

• Written in Erlang – a functional programming language

• Latest version – Riak v2.1
CAP THEOREM

- AVAILABILITY
- RIAK
- PARTITION TOLERANCE

RDBMS

CONSISTENCY
BASHO’S GOALS FOR RIAK

• Availability

• Operational Simplicity

• Scalability

• Masterless
FEATURES OF RIAK KV

- Resiliency
- Massive Scalability
- Operational Simplicity
FEATURES OF RIAK KV

- Intelligent Replication
- Complex Query Support
SUPPORTED LANGUAGES AND PLATFORMS

Languages

Ruby
Microsoft Visual C#
Erlang
node
Erlang
Node.js
Python
Java
PHP

Operating Systems

CentOS
Fedora
Ubuntu
Debian
Solaris
OpenSolaris
GOOD FITS FOR RIAK

• Immutable data

• Small objects

• Independent Objects

• Objects with Natural Keys

• Data Compatible with Riak Datatypes
NOT SO GOOD FITS FOR RIAK

• Objects that exceed the size 1-2 MB

• Objects with complex interdependencies
WHY MOVE FROM RELATIONAL TO RIAK

• High Availability
• Minimizing the cost of scale
• Simple Data Models
• Multi Data center Operations
RIAK RING ARCHITECTURE

- Keys and Objects - basic elements of Riak.

- Interaction with Objects - using the Bucket and the Key.

- Internally the structure is a ring as shown.
KEY TERMS IN THE RIAK RING

- Node
- Vnode
- Partition
FORMATION OF THE RING

• The method used is **Consistent Hashing** technique used to limit the reshuffling of keys when a hash-table data structure is rebalanced.
<table>
<thead>
<tr>
<th>FEATURE</th>
<th>USE CASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counters - keep track of increments/decrements</td>
<td>Track number of page &quot;likes&quot; or number of followers</td>
</tr>
<tr>
<td>Flags - enabled/disabled</td>
<td>• Has a tweet been re-tweeted</td>
</tr>
<tr>
<td></td>
<td>• Is a user eligible for preferred pricing</td>
</tr>
<tr>
<td>Sets - collection of binary values</td>
<td>• List items in an online shopping cart</td>
</tr>
<tr>
<td></td>
<td>• UUIDs of a user's friends in a social networking app</td>
</tr>
<tr>
<td>Registers - named binary with values also binary</td>
<td>• Store user profile names</td>
</tr>
<tr>
<td></td>
<td>• Store primary search location for a search engine user</td>
</tr>
<tr>
<td>Maps - supports nesting of multiple data types</td>
<td>Store user profile data composed of</td>
</tr>
<tr>
<td></td>
<td>• register user_name</td>
</tr>
<tr>
<td></td>
<td>• flag email_notifications</td>
</tr>
<tr>
<td></td>
<td>• counter site_visits</td>
</tr>
<tr>
<td>Hyperloglog</td>
<td>Count unique elements within a data set or stream, for example counting unique IP addresses or User IDs</td>
</tr>
</tbody>
</table>

SPECIAL DATA TYPES IN RIAK
SPECIAL DATA TYPES IN RIAK

• These data types are eventually convergent replicated data types (CRDTs)

• The advantage is Automatic Conflict Resolution
Intelligent Replication is used:

Another protocol that is used is the **Gossiping Protocol** – used to pass the ring state around the cluster.

An example of using the Gossiping Protocol is shown in the diagram:

```
put("artist", "REM")
```

(N=3)
ACHIEVING EVENTUAL CONSISTENCY

- What is quorum?
  - minimum number of parties that need to be successfully involved to consider an operation as whole.
- Default value of the quorum is 3
- Default quorum can be changed to user specified value
ACHIEVING EVENTUAL CONSISTENCY

- Writing with non-default value of quorum
  - change the parameter \(w\)

- Reading with non-default value of quorum
  - change the parameter \(r\)

- \(N > (r + w) = \) always consistent data
TALKING WITH RIAK

• HTTP API
  - Uses curl to interact
  - Operations specific to different elements

• PROTOCOL BUFFERS API
  - Riak listens to TCP port 8087 by default
  - On establishing connection – communication starts
  - Operations specific to different elements
## Comparing HTTP and Protocol Buffers

<table>
<thead>
<tr>
<th>HTTP</th>
<th>Protocol Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Higher data load</td>
<td>• Lesser data load</td>
</tr>
<tr>
<td>• Slower</td>
<td>• Faster</td>
</tr>
<tr>
<td>• Feature rich</td>
<td>• Feature deficient</td>
</tr>
</tbody>
</table>
OPERATIONS IN RIAK KV

**OBJECT**
- CREATE
- READ
- UPDATE
- DELETE

**CLUSTER**
- ADD NODE
- REMOVE NODE
- REPLACE NODE
OPERATIONS IN RIAK

• CREATE OBJECT

```python
bucket = client.bucket_type('animals').bucket('dogs')
obj = RiakObject(client, bucket, 'rufus')
obj.content_type = 'text/plain'
obj.data = 'WOOF!'  
obj.store()
```
OPERATIONS IN RIAK

• READ OBJECT

```
GET /types/<type>/buckets/<bucket>/keys/<key>
```

```
bucket = client.bucket_type('animals').bucket('dogs')
obj = bucket.get('rufus', r=3)
print obj.data
```
OPERATIONS IN RIAK

• UPDATE OBJECT

```python
bucket = client.bucket_type('siblings').bucket('coaches')
obj = RiakObject(client, bucket, 'seahawks')
obj.content_type = 'text/plain'
obj.data = 'Pete Carroll'
obj.store()
```

```python
def update_coach(team, new_coach):
    bucket = client.bucket_type('siblings').bucket('coaches')
    # The read phase
    obj = bucket.get(team)
    # The modify phase
    obj.data = new_coach
    # The write phase
    obj.store()

# Example usage
update_coach('packers', 'Vince Lombardi')
```
OPERATIONS IN RIAK

• DELETE OBJECT

```
DELETE /types/TYPE/buckets/BUCKET/keys/KEY
```

```
bucket = client.bucket_type('quotes').bucket('oscar_wilde').bucket('genius')
bucket.delete('genius')
```
OPERATIONS IN RIAK

• ADDING A NODE

```
bin/riak start

bin/riak-admin cluster join A
bin/riak-admin cluster join B
bin/riak-admin cluster join C

Success: staged join request for 'riak@192.168.2.5' to 'riak@192.168.2.2'
```
OPERATIONS IN RIAK

• REMOVING A NODE

```
riak-admin cluster leave riak@192.168.2.1
riak-admin transfer-limit <node> 0
```
OPERATIONS IN RIAK

• REPLACING A NODE

```
sudo tar -czf riak_backup.tar.gz /var/lib/riak /etc/riak
```

```
riak start
```

```
riak-admin cluster join riak0
```

```
riak-admin cluster commit
```

```
riak-admin cluster plan
```

```
riak-admin cluster replace riak4 riak7
```
RECOVERY MECHANISMS

• Node Failure
  - Hinted Handoff

• Data Integrity Failure
  - Read repair
  - Active Anti-Entropy
CONFLICT RESOLUTION STRATEGIES

• The type of strategy depends on the context of the conflict

• 2 contexts — casual and concurrent

• Casual Context - the difference of two versions of the same variable can be identified easily.

• Concurrent Context – the difference of the two versions of the same variable cannot be identified easily.
CONFLICT RESOLUTION STRATEGIES

• In case of casual context, the strategy is the **Dotted Version Vectors (DVV)** or the **Vector Clocks**

• DVV is also used for ordering the events in the distributed systems

• A vector clock is an algorithm for generating a partial ordering of events in a distributed system

• Main difference is that DVV can identify the value for each update, VC cannot do the same.
CONFLICT RESOLUTION STRATEGIES

• In case of concurrent context, the strategy is to create **Siblings** or resolve using the **Timestamps**

• In timestamp strategy, the value with the latest timestamp is used to resolve the conflict.

• In sibling strategy, siblings are created for each concurrent update, which need to be handled by the client logic.
**USE CASES & REAL TIME APPLICATIONS**

**Stores session data for gamers and players**

**Manage customer data records, and store subscriber session information for mobile and web applications.**

**store passenger information and session data**

**SESSION DATA**
USE CASES & REAL TIME APPLICATIONS

STORES BETTING INFORMATION FOR ALL THE CUSTOMERS.

USED TO HELP IN DELIVERING DEMOGRAPHIC AND CLINICAL INFORMATION.

USED AS THE CORE CONTENT STORE FOR THE PRODUCT CATALOG.
USE CASES AND REAL TIME APPLICATIONS

MESSAGING & CHAT

League of Legends
Used to provide worldwide in-game chat for millions of people

GE
Internal messaging for employees and customers.

Yammer
The Enterprise Social Network
Notification and alert systems.
PRESENT SCOPE OF RIAK KV
TO RIAK, OR NOT TO RIAK... THIS IS THE QUESTION

• What type of the data you have?
  - Requires distributed database?
  - Can it be managed as key and values?

• Is downtime unacceptable?

• Can the data be modelled as one of the data types or can be stored using these datatypes?
SOURCES

- https://images.google.com/
- http://wikipedia.org
- https://github.com
- Riak Handbook: A Hands-on guide by Mathias Meyer
Thank you