Python, PySpark and Riak TS

Stephen Etheridge
Lead Solution Architect, EMEA
Agenda

• Introduction to Riak TS
• The Riak Python client
• The Riak Spark connector and PySpark
BASHO SNAPSHOT

Distributed Systems Software for Big Data, IoT and Hybrid Cloud applications

2011 Creators of Riak Distributed Systems
- Riak KV: Resilient NoSQL database
- Riak S2: Large Object Storage

2015 New Products
- Basho Data Platform: Integrated NoSQL databases, caching, in-memory analytics, and search
- Riak TS: Only Enterprise NoSQL database optimized for Time Series data

100+ employees

Global Offices
- Seattle (HQ), Washington DC, London, Tokyo
MEETING THE NEEDS OF THE ENTERPRISE

PRIORITIZED NEEDS

- High Availability - Critical Data
- High Scale - Heavy Reads & Writes
- Geo Locality - Multiple Data Centers
- Operational Simplicity – Resources
  Don’t Scale as Clusters
- Data Accuracy – Write Conflict Options

RIAK KV USE CASES

- User Data
- Session Data
- Profile Data
- Real-time Data
- Log Data

TIME SERIES USE CASES

- IoT/Devices
- Financial/Economic
- Scientific Observations
20 TERABYTES OF DATA PER DAY BILLIONS OF MOBILE DEVICES

- 10 BILLION data transactions a day – 150,000 a second – Apple
- Forecasting 2.8 BILLION locations around the world
- Generates 4GB OF DATA every second

We’re focusing on helping people make better decisions with the weather.
WHAT IS NEEDED FOR TIME SERIES?

- Efficient way to store & retrieve time series data
- Query language that supports range queries
- High data volume
- Enterprise scale solution
- High availability
Riak TS is Riak KV (a complete Riak KV build is included in Riak TS) with the following additional features optimized to handle time series use cases:

- **Tables**: Riak TS introduces tables built on top of the underlying K/V structure.
- **SQL**: Riak TS supports a subset of standard SQL to create and query time series data.
- **Data Locality**: Keys co-located by quanta to enable querying data across time bounded series.
**Riak TS Quanta**

The **Quantam** function in **Riak TS** takes three parameters:

- The name of a field in the table definition of type timestamp;
- A numeric quantity;
- One of the units of time from the list below:
  - Days – ‘d’
  - Hours – ‘h’
  - Minutes – ‘m’
  - Seconds – ‘s’

**Important:** A query covering more than a certain number of quanta (5 by default) will generate too many sub-queries and the query system will refuse to run it. Assuming a default quanta of 15 minutes, the maximum query time range is 75 minutes.
Supported Aggregate Functions

Riak TS supports aggregate functions including:

- **COUNT()** - Returns the number of entries that match a specified criteria.
- **SUM()** - Returns the sum of entries that match a specified criteria.
- **MEAN() & AVG()** - Returns the average of entries that match a specified criteria.
- **MIN()** - Returns the smallest value of entries that match a specified criteria.
- **MAX()** - Returns the largest value of entries that match a specified criteria.
- **STDDEV()** - Returns the statistical standard deviation of all entries that match a specified criteria using Population Standard Deviation.
Supported Data Types

Riak TS tables support the following data types:

- **Varchar** - Any string content is valid, including Unicode. Can only be compared using strict equality, and will not be typecast (e.g., to an integer) for comparison purposes. Use single quotes to delimit varchar strings.

- **Double** - This type does not comply with its IEEE specification: NaN (not a number) and INF (infinity) cannot be used.

- **Sint64** - Signed 64 bit integer

- **Boolean** - true or false (any case)

- **Timestamps** - Timestamps are integer values expressing UNIX epoch time in UTC in milliseconds. Zero is not a valid timestamp.
Developing on Riak TS

Riak TS currently supports the Protocol Buffers API and five client libraries including Java, Ruby, Python, Erlang, and Node.js.

<table>
<thead>
<tr>
<th>APIs</th>
<th>Basho Clients</th>
<th>Community Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Protocol Buffers</td>
<td>• Java</td>
<td>• Not yet!</td>
</tr>
<tr>
<td></td>
<td>• Ruby</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Erlang</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Node.js</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• .NET c#</td>
<td></td>
</tr>
</tbody>
</table>
**Supported Operations**

Riak TS clients currently support following operations:

- **Delete** - Deletes a single row by it's key values.
- **Fetch/Get** - Fetches a single row by it's key values.
- **Query** - Allows you to query a Riak TS table with the given query string.
- **Store/Put** - Stores data in the Riak TS table.
- **(Stream) ListKeys** - Lists the primary keys of all the rows in a Riak TS table.
The Riak Python Client

• Compatible with Python 2.7 and above
• Can be installed easily with pip
• Pre-requisites
  – python-dev
  – libffi-dev
  – libssl-dev

• Riak TS results object can be turned into a Pandas dataframe easily, otherwise it is a list of lists!

• Demo with Aarhus data
Riak Spark Connector

- Enables you to connect Spark applications to Riak TS with the Spark RDD and Spark DataFrames APIs
- Write applications in
  - Scala (if you have to),
  - Python (yay!),
  - and Java (never!).
- Makes it easy to partition Riak data so multiple Spark workers can process the data in parallel.
- Has support for failover if a Riak node goes down while your Spark job is running.
- Comes as one JAR file that needs to be pathed in!
  - Riak TS 1.2+
  - Apache Spark 1.6+
  - Scala 2.10
  - Java 8