A TECHNICAL OVERVIEW OF RIAK® TS ENTERPRISE

INTRODUCTION

The Internet of Things (IoT) or the Internet of Everything is changing the way companies interact with their customers and manage their data. These connected devices generate high volume time series data that can be created in milliseconds. This fast growth of IoT data and other time series data is producing challenges for enterprise applications where data must be collected, saved, and analyzed in the blink of an eye. Your application needs a database built to uniquely handle time series data to ensure your data is continuously available and accurate.

IoT AND TIME SERIES DATA

- Internet of Things (IoT) / Device data
- Stock market indices
- Commodity prices
- Unemployment numbers
- Tide measurements
- Solar flare tracking
- Health information
- ARIMA models

Let's look at some of the challenges created by time series data. In the past, data was collected from a small number of sources and had a limited scope. However, as you collect more and more data with increasing frequency from greater numbers of devices over longer periods of time, the resulting data becomes increasingly unmanageable. Beyond just collecting this time series data, organizations need to process, analyze, and often take action on their findings. Frequently, the period of time that data is useful depends on the type of data and its frequency. Generally, as your time series data grows older, it becomes less relevant. The latest data is the most compelling and much older data is occasionally interesting. For example, detailed weather data may be important for a day or week but aggregate data becomes important over time. Lastly, the infrastructure to collect, process, and analyze this data often cannot handle the load leading to missing data and unavailable applications. To address these problems, Basho built Riak TS.

Riak TS is the only enterprise-ready, NoSQL database specifically optimized to store, query, and analyze time series data. Like Riak KV, Riak TS solves difficult distributed systems challenges to ensure your time series data is accurate, highly available, scalable, and is easy to manage at scale.

HOW IS TIME SERIES DATA DIFFERENT?

Time series data is sequenced information recorded at different points in time going forward. Time series data is different from other unstructured data, because "close-in-time" data points are likely to share common traits and be queried together based on time ranges. Examples of time series data include IoT device data, stocks, commodity prices, tide measurements, solar flare tracking, and health information.

Time series data has unique requirements, including high performance reads and writes, and the ability to retrieve data using range queries while remaining fast, reliable, and scalable even with a huge number of data sources. Collecting, storing, accessing, and analyzing this data with traditional or other NoSQL databases is just not possible.

WHY RIAK TS FOR TIME SERIES DATA?

Riak TS is a key/value store optimized for fast reads and writes of time series data. Riak TS is built to handle the unique needs of time series applications ensuring high availability, data accuracy, and scale. To accomplish this, Riak TS automatically co-locates, replicates, and distributes data across the cluster.

Its unique masterless architecture enables linear scale using commodity hardware, so you can easily add capacity as your time series data grows. Riak TS provides flexibility to support both structured and semi-structured data as well as the ability to write range queries.

RIAK TS ARCHITECTURE

OPTIMIZED FOR TIME SERIES DATA

Riak TS is specifically architected to optimize performance for time series data. The masterless architecture ensures speed, fault tolerance, and availability. It also co-locates related time series data on nodes for fast reads and writes. By co-locating and ordering data based on a composite key, Riak TS can service queries from a minimum number of partitions. This allows the system to avoid the expensive coverage queries that would be necessary if you were to use secondary indexes to perform the same query. Riak TS also performs data filtering on the storage backend which provides another level of efficiency and minimizes network overhead by not transferring extraneous data. You can also configure automatic and efficient data expiry. Let's look further into this architecture and time series optimizations.

MASTERLESS

At its core, Riak TS is a key/value database, built from the ground up to safely distribute data across a cluster of servers, called nodes. A Riak TS cluster is a group of nodes that are in constant communication to ensure data availability and partition tolerance.

Riak TS has a masterless architecture in which every node in a cluster is capable of serving read and write requests. All nodes are homogeneous with no single master or point of failure. Any node selected can serve an incoming request, regardless of data locality, providing data availability even when hardware, or the network itself, is experiencing failure conditions. This ensures your IoT or time series application is always available for both read and write operations, which is especially important when ingesting potentially millions of time series data points.

THE RIAK® RING

The basis of the Riak TS masterless architecture, replication, and fault tolerance is the Riak Ring. This Ring is a managed collection of partitions that share a common hash space. The hash space is called a Ring because the last value in the hash space is thought of as being adjacent to the first value in the space. Replicas of data are stored in the "next N partitions" of the hash space, following the partition to which the key hashes.

The Ring is also used as shorthand for the "Ring State." The Ring State is a data structure that gets communicated and stays in sync between all nodes, so each node knows the state of the entire cluster. If a node gets a request for an object managed by another node, it consults the Ring State and forwards the request to the proper node, effectively proxying the request as the coordinating node. If a node is taken offline permanently or a new server is added, the other nodes adjust, balancing the partitions around the cluster, then updating the Ring State. YYou can read more about nodes, vnodes, clusters, the Ring, and Ring State at docs.basho.com.

NODES AND VNODES

Each node in a Riak TS cluster manages one or many virtual nodes, or vnodes. Each vnode is a separate process, which is assigned a partition of the Ring and is responsible for a number of operations in a Riak cluster. These operations include storing objects, handling incoming read/write requests from other vnodes, and interpreting causal context metadata for objects.

This uniformity of Riak TS vnode responsibility provides the basis for the fault tolerance and scalability found in Riak TS. If your cluster has 64 partitions and you are running three nodes, two of your nodes will have 21 vnodes, while the third node holds 22 vnodes.

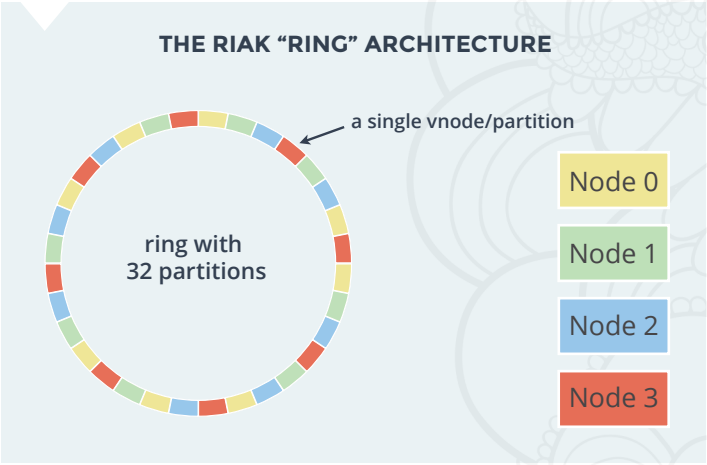


Fig 1: Shows a Riak TS ring with 32 partitions call vnodes

The concept of vnodes is important as we look at data replication. No single vnode is responsible for more than one replica of an object. Each object belongs to a primary vnode, and is then replicated to neighboring vnodes located on separate nodes in the cluster.

With Riak TS, time series data is written such that data from a desired time period is written to the same partition and therefore is co-located with data from the same time period. This data co-location based on time quantization enables faster reads and queries so that your time series data is faster to analyze and easier to manage.

The time quantization grouping of time series data is based on both a configurable composite partition key and local key. The partition key is a three-part key composed of a data family object identifier label, a series identifier, and a time quantum. The data family label qualifier is applied so that completely different kinds of time series data, whether short-lived weather data (e.g. quantum = 1 day) or long-lived access logs (e.g. quantum = 7 years), can be very simply and independently managed. The quantum is the amount of time that you want similar data from the same data family and series to be written to the same partition.

The local key is used for locality, sort order, and faster access of grouped (time quantized) data leveraging LevelDB's sort ordered data storage. It enables queries to run faster by ensuring that grouped data can be read all at once and processed together.

INTELLIGENT REPLICATION

Riak TS automatically distributes data across nodes in a Riak TS cluster and yields a near-linear performance increase as you add capacity. Data is distributed evenly across nodes using hashing with time series data co-located for performance. When you add (or remove) machines, data is rebalanced automatically in a non-blocking operation. Riak TS will continue servicing read and write requests, allowing the ability to perform scaling operations without the need for downtime. Any new machines that are added to a cluster claim a portion of the overall dataset, which is then redistributed, with the resulting cluster status shared across all vnodes.

Replication ensures that there are multiple copies of data across multiple servers. By default, Riak TS makes 3 replicas, each belonging to a unique vnode. The primary benefits derived from replication are high availability and low latency, even under peak load. If a server in the cluster becomes unavailable, other servers that contain the replicated data remain available to serve requests and the whole system remains highly available.

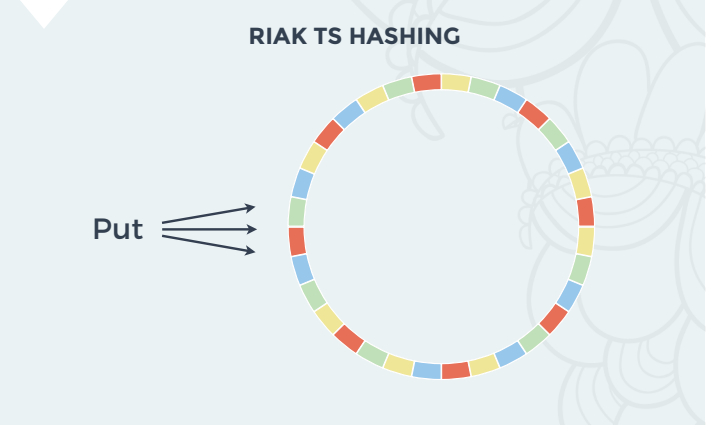


Fig 2: Shows data from the same quantum is written to the same partition. Queries for the time series data from the same quantum are read from the same partition for faster access and faster queries

To further understand intelligent replication in Riak TS, it is helpful to understand how data is distributed across the cluster. Riak TS chooses one vnode to exclusively host a range of keys, and the other vnodes host the remaining non-overlapping ranges. With partitioning, the total capacity can increase by simply adding commodity servers.

Since replication improves availability and partitions allow us to increase capacity, Riak TS combines both partitions and replication to work together. Time series data that is grouped and stored together is partitioned as well as replicated across multiple nodes to create a horizontally scalable system.

EVENTUAL CONSISTENCY

One of the important differences and key advantages of Riak TS is the concept of eventual consistency. Eventual consistency, means that not all of the assigned nodes for a write operation in a distributed system need to have that write confirmed before the distributed system considers that write to be complete. This allows for a higher degree of workload concurrency and data availability.

Riak TS is the most highly resilient NoSQL database to use for your time series applications. By default, Riak TS replicas are "eventually consistent," meaning that while data is always available, not all replicas have the most recent update at the exact same time, causing brief periods— generally on the order of milliseconds—of inconsistency while all state changes are synchronized. Riak TS is designed to deliver maximum data availability.

NODE OR NETWORK FAILURE RECOVERY

Due to replication and eventual consistency, it then becomes important to understand the concept of quorums, which define the success or failures of reads and writes. A quorum is defined by the number of replicas that you define. By default, a quorum is half of all the total replicated nodes plus one more (a majority).

You can set a value to give you control over how many replicas must respond to a read or write request for the request to succeed ("R value" for reads and a "W value" for writes). If this value is not specified, Riak TS defaults to requiring a quorum, where the majority of nodes must respond.

A "strict quorum" is where a request would fail (and deliver an error message to the client) if the required number of primary nodes cannot accept requests. Using the strict quorum would work for most requests. However, at any moment, a node could go down or the network could partition, triggering the unavailability of the required number of nodes. Therefore, to provide resilience, Riak TS defaults to what is known as a sloppy quorum, meaning that if any primary (expected) node is unavailable, the next available node in the cluster will accept requests. That node will then update the primary node when it comes back online. This ability to easily handle node failures is known as a hinted handoff.

Hinted handoff ensures that if a node goes down, a neighboring node will take over its storage operations. When the failed node returns, the updates received by the neighboring node are handed back to it. This ensures that availability for writes and updates is maintained automatically, minimizing the operational burden of failure conditions.

RIAK TS DEVELOPMENT CONCEPTS

READING AND WRITING DATA

Riak TS stores data as a combination of keys and values in buckets. Key/Value stores are conceptually like hash tables, where values are stored and accessed by an immutable key. Riak TS functions like a very large hash space, also known as a hash table, a map, a dictionary, or an object.

Riak TS has two types of keys:

- Partition keys — determine where the data is placed on the cluster
- Local keys —determine where the data is written in the partition

Partition keys use time quantization to group data that will be queried together in the same physical part of the cluster. Time quantization says "group data by 15-minute clumps, or 10-second clumps, or 60-day clumps" depending on how quickly your time series data come in and how you need to analyze them.

The quantization is configurable on a table level. In order to query Riak TS data, data is structured using a specific schema. The schema defines what data can be stored in a Riak TS table and what type it has. Queries can then be written against that schema and the Riak TS query system can validate and execute them.

We have combined the definition of the various keys and the data schema into a single SQL statement. The query language is SQL, so you will use the field names and the table name in those queries; SQL conventions such as case sensitivity hold.

BUCKETS AND TABLES

Riak TS buckets have a one-to-one mapping with tables. Buckets provide logical namespaces so that identical keys in different buckets will not conflict.

There is also a level of classification that exists above buckets, called "bucket types." Bucket types are groups of buckets with a similar set of properties. The benefit of bucket types is that a group of distinct buckets can share properties. Riak TS enables you to define and configure tables of time series data as a bucket type, and write data to these tables. The schema of Riak TS tables are generated as bucket properties and installed as bucket types, which allows you to structure data as it is coming in and store both structured and semi-structured data. Creating a table will specify the schema and compile it, which will create a bucket. The bucket information is passed around the ring.

DATA MODELING

Riak TS provides incredible flexibility for storing time series data. The DDL for table and field definitions enables you to "type" time series structured and semi-structured data, which means that developers can define their own schema. Riak TS allows you to define a schema with a SQL CREATE TABLE statement. Tables can include any number of columns with data types including integer, boolean, timestamp, sint64, and double. Every table must include a timestamp column as well as a primary key definition. The primary key usually uses time quantization to allow the user to specify the extent to which ordered ranges of data are partitioned together in a cluster. The quantum function groups data into chunks based on a user-defined time period such as seconds, minutes, hours, days. As a rule, primary keys are defined by a combination of the quantum function as well as a series ID. The "bucket type" is then created and includes the table definition as one of its properties.

Once the schema is defined, you can use the SHOW TABLES command to return a list of all tables in the schema, and use DESCRIBE to dynamically discover your schema, including the quantum.

By applying a schema and typing your time series data, you can use Riak TS to collect, store, access, and analyze this data while enabling your applications to scale out, up, and down predictably and linearly as your data grows — ensuring that reads and writes to the database are fast, reliable, and scalable.

RANGE QUERIES

Riak TS enables single-key DELETEs and GETs, which allow you to read and modify data, but for most time series data it is easier to query using SQL. SQL queries can consist of standard select statements that allow you to specify a time range, and a series ID, as well as which subset of your columns that you would like to return. You can use optional secondary fields and apply standard logical operators (`=, !=, >, <, <=, >=`) and boolean operators AND and OR to filter the results set.

To enable you to query time series data in Riak TS, the data is stored in a schema. The schema defines its type and which data can be stored in the time series bucket. Queries can then be written and validated against that schema. Range queries and data co-location in Riak TS make queries on your data faster and easier to run.

DATA EXPIRY

Data expiry, or global object expiration, allows you to remove aged and unnecessary data from the database. Once configured, data that only needs to be persisted for a limited time will be automatically and efficiently deleted.

SPARK CONNECTOR

The Spark Connector efficiently retrieves data from Riak TS to Spark for in-memory distributed processing, then the results can be stored back in Riak TS as needed. The ability to persist these results to Riak TS provides flexibility for future data processing or analysis. The seamless integration of Riak TS with Apache Spark ensures easier and faster operational analysis of time series data.

EXAMPLE - CREATING A TABLE AND RANGE QUERY FOR WATER DATA IN RIAK TS

CREATE TABLES TO STORE TIME SERIES DATA

Riak TS buckets are used to define tables. When you create a new bucket, you define it with a DDL:

```
CREATE TABLE WaterData (
    waterdistrict varchar not null,
    location varchar not null,
    time timestamp not null,
    temp double not null,
    salinity double not null,
    PRIMARY KEY (( waterdistrict, location,
quantum(time,15, m)), waterdistrict, location, time)
```

San Francisco Water Data

Site	ISO 8601 Timestamp	Temperature (C)	Specific Conductance (ÂµS/cm)	Salinity (PSU)	Turbidity (FNU)
PIER_17	2015-09-18T00:00:00-08:00	17.8	48500	31.6	3
PIER_17	2015-09-18T00:15:00-08:00	17.8	48500	31.6	2.8
PIER_17	2015-09-18T00:30:00-08:00	17.7	48400	31.6	2.8
PIER_17	2015-09-18T00:45:00-08:00	17.8	48500	31.6	3.6
PIER_17	2015-09-18T01:00:00-08:00	17.7	48500	31.6	2
PIER_17	2015-09-18T01:15:00-08:00	17.3	48500	31.6	2.8
PIER_17	2015-09-18T01:30:00-08:00	17.4	48500	31.6	2.4
PIER_17	2015-09-18T01:45:00-08:00	17.3	48500	31.6	3.8
PIER_17	2015-09-18T02:00:00-08:00	17.4	48400	31.6	3
PIER_17	2015-09-18T02:15:00-08:00	17.4	48500	31.6	2.3
PIER_17	2015-09-18T02:30:00-08:00	17	48600	31.7	1.4
PIER_17	2015-09-18T02:45:00-08:00	17	48600	31.7	2.1
PIER_17	2015-09-18T03:00:00-08:00	16.9	48700	31.8	2.4
PIER_17	2015-09-18T03:15:00-08:00	16.8	48700	31.8	5.6

QUERY USING A TIMESTAMP OR DATE/TIME FORMAT

SELECT statement using timestamps (as long integers):

```
SELECT * from WaterData
WHERE waterdistrict = 'San Francisco County' AND
    location = 'Pier 17' AND
    time >= 144378190000 AND
    time <= 144379690000 AND
    temp > 18 OR salinity > 32
GROUP BY waterdistrict;
```

SELECT statement using ISO 8601 date/time format:

```
SELECT * from WaterData
WHERE waterdistrict = 'San Francisco County' AND
    location = 'Pier 17' AND
    time >= '1974-07-30 01:03:10' AND
    time <= '1974-07-30 01:28:10' AND
    temp > 18 OR salinity > 32
GROUP BY waterdistrict;
```

RIAK TS ENTERPRISE

Riak TS Enterprise is a commercially distributed product built on Riak (Apache 2.0-license) that extends Riak TS with Multi-cluster Replication and 24x7 support.

MULTI-CLUSTER REPLICATION

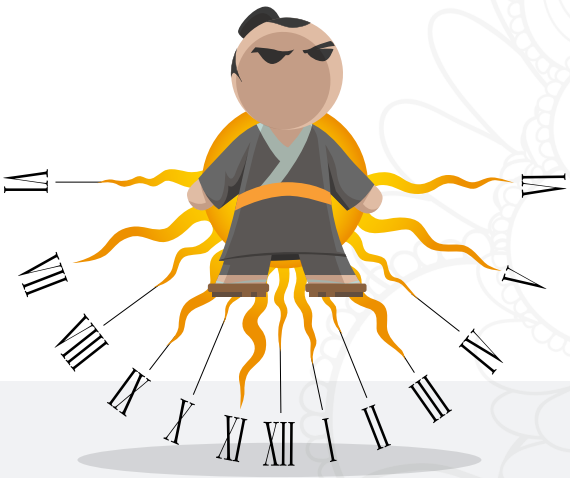
Riak TS Enterprise offers Multi-cluster Replication so that data stored in Riak TS can be replicated for backup-clusters, analysis clusters, or for multiple datacenters. With Riak TS Enterprise, data can be replicated across the datacenter or across geographic areas, providing disaster recovery, data locality, compliance with regulatory requirements, the ability to "burst" peak loads into public cloud infrastructure, and more.

In Multi-cluster Replication, one cluster acts as a primary, or source, cluster. The primary cluster handles replication requests from one or more secondary clusters (often located in datacenters in other regions or countries). If the datacenter with the primary cluster goes down, a secondary cluster can take over as the primary cluster ensuring your data is always available.

NEXT STEPS

Integrate Riak TS into your IoT and Big Data applications to make your time series data actionable using familiar querying tools and a highly available, fault-tolerant, enterprise-grade, scalable datastore.

If you are interested in evaluating Riak TS, please contact us. We would be happy to arrange a tech talk with your team, or answer any questions about our products. Our Professional Services Team can assist you in planning, setting up, and optimizing your Deployment.



ABOUT BASHO TECHNOLOGIES

Basho, the creator of the world's most resilient databases, is dedicated to developing disruptive technology that simplifies enterprises' most critical distributed systems data management challenges. Basho has attracted one of the most talented groups of engineers and technical experts ever assembled devoted exclusively to solving some of the most complex distributed systems challenges presented by Big Data and IoT.

Basho's database, Riak® KV, the industry leading distributed NoSQL database, is used by fast growing Web businesses and by one-third of the Fortune 50 to power their critical Web, mobile and social applications. Built on the same foundation, Basho introduced Riak TS, which is the first enterprise-ready NoSQL database specifically optimized to store, query and analyze time series data. Basho also provides Riak integrations for a variety of Big Data technologies like Apache Spark, Redis, Mesos, and Apache Solr.

For more information visit Basho.com which is full of interesting use cases, customer case studies and product detail, or docs.basho.com for technical documentation.

basho

SEATTLE - HEAD OFFICE
10900 NE 8th Street
Suite 1580
Bellevue, WA 98004
617.714.1700

WASHINGTON, D.C.
12930 Worldgate Drive
Suite 120
Herndon, VA 20170
617.714.1700

LONDON
Fourth Floor
South Warwick House
65/66 Queen Street
London, EC4R 1EB
+44 020 3201 0032

PARIS
6th Floor
105 rue Anatole France
Levallois-Perret, Paris
92300 France
+33 1 73 44 66 71

TOKYO
Basho Japan KK
NK7 Building 3rd Floor 2-9
Yotsuya Shinjuku-ku
Tokyo, Japan 160-0004
03-5953-1780

08/2016