redislabs
home of redis

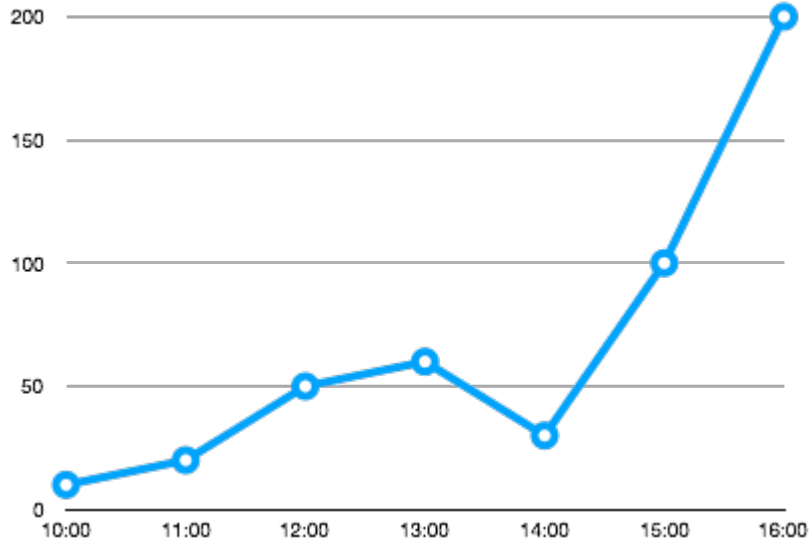
redisday
TLV 2018

Redis as a time-series DB

Danni Moiseyev, Redis Labs

What is a time series?

- What do you mean when you say time series?
- An ordered list of samples, each sample is a timestamp and value.
- Usually used for metrics
- Downsampled for longer retention time



Time-series are “easy” to work with

- Queries are usually a range query over time
- Can be compressed by using known techniques
- Timestamps are easy to index
- Append data only

Why store time series in Redis?

- People are already doing that
- Serving fast queries for time series
  - Dashboards
  - Mobile/IOT
  - Fast alerts on data
- Fast ingestion rate
  - use Redis as the first line metrics ingester
- Ease of deployment
  - Just run Redis with the module

Introducing redis-timeseries

- Started as an internal need for time series within Redislabs
- Custom data structure in Redis
- Built-in Redis facilities:
  - Persistence (AOF and RDB)
  - Replication
- Each key is a time series

Commands: TS.ADD

- Add a new sample:

  TS.ADD KEY TIMESTAMP VALUE

- Usage:

  $ TS.ADD temperature 1519820907 11.2

  $ TS.ADD temperature 1519820910 14

  $ TS.ADD temperature 1519820915 18.2

- Caveats:
  - You can’t add older samples than your latest timestamp

Commands: TS.INCRBY / TS.DECRBY

- Add a new sample:

  TS.INCRBY KEY VALUE [RESET TIME_SECONDS]

- Usage:

  $ TS.INCRBY active_machine 3

  $ TS.DECRBY active_machine 1

  $ TS.INCRBY active_users 1 RESET 5

- Caveats:
  - The timestamp that are used are based on the redis host clock

Command: TS.RANGE

- Range query over a specific key

  TS.RANGE KEY [START TIMESTAMP] [END TIMESTAMP] [AGGREGATION TIME_BUCKET]

  - Aggregations are one of: avg, max, min, count, sum.

- Example:

  $ TS.RANGE temperature 1519820907 1519830907

- Example - Aggregate to average of 5 mins (300 seconds):

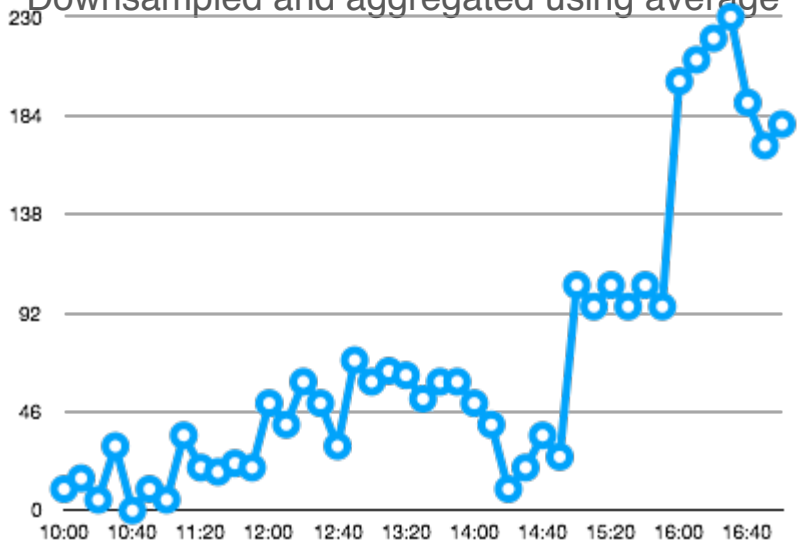
  $ TS.RANGE temperature 1519820907 1519830907 avg 300

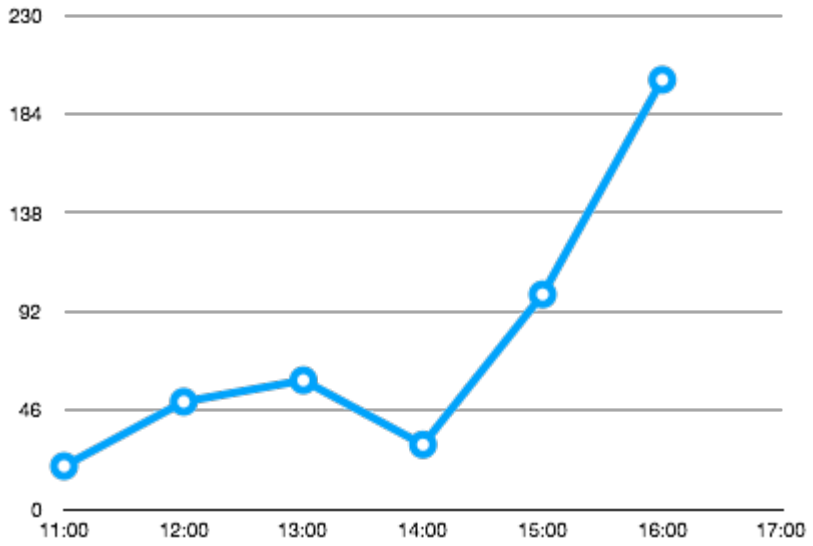
Downsampling rules

- What is downsampling?

Raw Data

Downsampled and aggregated using average





Downsampling rules - cont'd

- Each key can have multiple rules
- Each rule contains:
  - Destination key
  - Aggregation type: avg, min, max, count, sum
  - Time bucket in seconds
- Example:
  - $ TS.CREATERULE temperature AVG 60 temperature_AVG_60

    The destination key is temperature_AVG_60 will contain the aggregated data using average over 60 seconds bucket.
- Downsampling rules have very little performance impact

Global configuration

- Load the module with default configuration for downsampling rules
- Allows Calling to TS.ADD without creating the key
- Global Downsampling rules
  - Aggregation : time_bucket : retention time
  - Example: max:1m:1d;min:10s:1h;avg:2h:10d;avg:3d:100d
- Global retention policy per key

Differences between Sortedset and time series

- Ranged Aggregations are not possible in sortedset
- Automatic downsampling and retention time
- Memory optimized
- Less traffic between client <-> Redis

Differences between Streams and time series

- Streams are optimized for general event logging
- Time series are optimized for numeric streams
- Automatic Downsampling
- Aggregated queries
- (Future) Data compression
- (Future) Using some parts of redis streams

QA

PRs are welcome

https://github.com/danni-m/redis-timeseries


Thank You!
redisday
TLV 2018
redislabs
home of redis