Redis as a Time Series DB

Josiah Carlson - @dr_josiah

Agenda

- Who are you?
- What is Redis? (3 minutes, optional)
- What is a time series database?
- Combining structures for success
- Analyzing/segmenting events
- Next steps
- Questions

Who are you?

- A guy who does a lot of stuff with Redis and Python
  - Book:
    - Redis in Action http://manning.com/carlson
    - Read for free http://bit.ly/ria-free
  - Libraries: https://github.com/josiahcarlson
    - rom, RPQueue, lua_call, parse-crontab, ...
  - Mailing list: https://groups.google.com/forum/#!forum/redis-db (not active here recently)
  - Blog: dr-josiah.com
  - Twitter: @dr_josiah

What is Redis?

- In-memory key/data structure server
  - Strings, lists, hashes, sets, sorted sets (plus integers and floats)
  - HyperLogLog, lexicographic prefix scans, geo index/search api (unreleased 3.2)
  - Pubsub, Lua scripting (like a stored procedure)
- Ops features:
  - Master/slave replication + sentinel for HA
  - Alternative “cluster” mode with sharding/slaving/failover
  - On-disk persistence; point-in-time or incremental

What is a time series database?

- Not about the database, but about the data and operations
  - Discrete events or samples of at least one value or metric over time
  - Sometimes the *event* is the value/metric
  - If it has a timestamp (or one is implied), it is part of a time series
- Examples of data:
  - Data gathered from sensors embedded inside IoT devices
    - Nest thermostat - measures temperature, analyzes your preferences...
  - Sell prices and volumes of traded stocks
  - Values and delivery locations of orders placed at an online retailer
  - Actions of users in a video game and their outcome

Combining structures for success

- Consider a log of user actions on a web site like:
  - **{‘ts’: 1458710360.679, ‘user’: 218946, ‘type’: ‘login’, ‘ip’: ‘172.32.5.6’}**
- We can get an easily sliced time series with individual events stored in hashes, and a sorted set for an index of timestamps:
  - Increment the **actions:** key, which becomes the event **id**
  - Store each event individually in its own HASH at key: **actions:<event_id>**
  - Add an **(<event_id>: <ts>)** member/score pair to a ZSET at key: **actions:ts**
- To use:
  - Scan over **actions:ts** using ZRANGEBYSCORE or ZRANGE
  - Use HGET/HMGET/HGETALL to fetch the actions for analysis

Combining structures for success (reading)

```
-- Many-caveats about using this, you probably don't want it (not cluster compatible)!
-- KEYS should be something like:
--     {'action:ts', 'action:'}
-- ARGV should be something like:
-- (gets all events from 2016-03-23 UTC)
--     {'1458691200', '1458777600'}

local ids = redis.call('ZRANGEBYSCORE', KEYS[1], unpack(ARGV))

local results = {}
for i, id in ipairs(ids) do
    local item = redis.call('HGETALL', KEYS[2]..id)
    if item then
        table.insert(results, item)
    end
end

return cjson.encode(results)
```

Combining structures for success (writing)

```
-- Many-caveats about using this, you probably don't want it (not cluster compatible)!
-- KEYS should be something like:
--     {'action:'}
-- ARGV should be something like:
--     {'<json-encoded data, with ts>'}

local new_id = redis.call('INCR', KEYS[1])
local dest = KEYS[1] .. new_id

local data = cjson.decode(ARGV[1])
data.id = new_id

for k, v in pairs(data) do
    redis.call('HSET', dest, k, v)
end

-- note:    ZADD       <key>        <score> <member>
redis.call('ZADD', KEYS[1] .. 'ts', data.ts, new_id)
```

Analyzing/segmenting events

We now have events, but want to (for example) count the number of events of different types in a time range, equivalent to:

```
SELECT type, COUNT(*)
FROM actions
WHERE ...
GROUP BY type
```

- We can use Lua to scan over the items in the time range, counting them
- We can create additional ZSETs, one for each event type...
  - **`actions:login:ts`**, **`actions:logout:ts`**, ...
  - Stores **`(<event_id>: <ts>)`** pairs, like **`actions:ts`**
  - Can use ZCOUNT to get a count without scanning*

Analyzing/segmenting events (continued)

- Or if you needed to index events by user id
  - Add (**<event_id>, <user_id>**) member/score pairs to **actions:user**
  - Easy to get total count/user with ZCOUNT (**ZCOUNT actions:user <user_id> <user_id>**)
  - Can partially induce a secondary timestamp index over the same data; to fetch a specific time range for one user
    - Use ZRANGEBYSCORE and ZREVRANGEBYSCORE on **actions:ts** for start/end of id range for a pair of timestamps (assuming synchronized clocks, or using Redis 3.2-unreleased Lua side-effects propagation)
    - Either:
      - Use ZRANGEBYSCORE on **actions:user** to scan ids and filter
      - Use ZRANGEBYSCORE + ZREVRANGEBYSCORE + ZRANGE to bisect (speed optimization, requires left padding of members with zeros)

Combining structures for success (revisited)

- If you know you never need to extract individual fields, and you know that your events are unique...
  - Use a single ZSET with **(<encoded json event>: <ts>)** pairs
- If you are planning on bucketing based on time slot, only need a list of items
  - Use LISTs keyed as **actions:<time slice>**
  - Just RPUSH **<encoded json event>** onto these LISTs
- Approximate counts? HyperLogLog
- If you plan your metrics, you can explicitly count them in real time
  - INCR, DECR, INCRBY, INCRBYFLOAT, HINCRBY, HINCRBYFLOAT, ZINCRBY, all very fast
  - Lua scripting can be a huge win here

Combining structures for success (revisited)

- Redis doesn’t need to be 100% of this
  - If you only need counts over arbitrary time ranges, just **`action:ts`** or the **`action:login:ts`** and related sorted sets may be enough
  - Redis can be your counters/aggregate/sorted set storage, even if it’s not your event storage
- HASHes + ZSETs can solve just about anything, alternatives **can** offer:
  - Better performance
  - More convenient access to necessary data
  - Lower memory utilization

Next steps

Time series is a specific example of data modeling...

- My data modeling talk from Redisconf 2015
  - Get an idea of the general process involved
- Specific examples:
  - For indexing/search: chapter 7 of RiA, my Redisconf 2012 talk
  - Other data modeling examples: the rest of RiA, my blog
  - Survey of data modeling examples: my Python with Redis talk from July 2012
- Google will know more

Questions?

Josiah Carlson - @dr_josiah