ReDe

A Redis Element Dehydration Module

Built For The Redis Modules Hackathon, November 2016

Quick Intro - Dehydration

- A dehydrator is a fancy time-queues
- Where built to solve the Contextual Completeness and Emergent Relevancy problems.
- Designed and developed @ Tamar Labs in mid 2015 (using Python+Redis).
- Provides these basic commands:

Quick Intro - Dehydration cont.

Push - Insert an element, it will need an id, the element itself and dehydration time in seconds.

Pull - Remove the element with the appropriate id before it expires.

Poll - Pull and return all the expired elements.

Other commands this module provides

TTN - Return the minimal time between now and the first expiration.

Look - Search the dehydrator for an element with the given id and if found return its payload (without pulling).

Update - Set the element represented by a given id, the current element will be returned, and the new element will inherit the current expiration.

Common Use Cases

Stream Coordination - Make data from one stream wait for the corresponding data from another (preferably using sliding-window style timing).

Event Rate Limitation - Delay any event beyond current max throughput to the next available time slot, while preserving order.

Self Cleaning Claims-Check - Store data for a well known period, without the need to search for it when it is expired or clear it from the data-store yourself.

Task Timer - Postpone actions and their respective payloads to a specific point in time.

Internals

- Hash map of doubly linked lists representing queues.
- Each queue represents a different ttl.
- So that pull order can be preserved within each queue regardless of expiration.
- Also, another Hash map is used for quick lookup of a specific items.

Command Performance

elements/sec by version

	v0.1.*	v0.2.*	v0.3.*
Push	16,000	23,000	22,000
Pull	19,500	31,000	31,500
Poll	1,700	265,000	305,000

Quick Start Guide

Prep.

Here's what you need to do to build this module:

1. Build Redis in a build supporting modules.
2. Build the module:

   ```
   make
   ```

3. Run Redis loading the module:

   ```
   /path/to/redis-server --loadmodule path/to/module.so
   ```

Now run redis-cli and try the commands:

```
127.0.0.1:9979> REDE.PUSH some_dehy id1 world 15
OK
127.0.0.1:9979> REDE.PUSH some_dehy id2 hello 1
OK
127.0.0.1:9979> REDE.PUSH some_dehy id3 goodbye 2
OK
127.0.0.1:9979> REDE.PULL some_dehy id3
"goodbye"
127.0.0.1:9979> REDE.POLL some_dehy
1) "hello"
127.0.0.1:9979> REDE.POLL some_dehy
(empty list or set)
127.0.0.1:6379> REDE.LOOK some_dehy id2
(nil)
127.0.0.1:6379> REDE.LOOK some_dehy id1
"world"
127.0.0.1:6379> REDE.PULL some_dehy id2
(nil)
127.0.0.1:6379> REDE.TTN some_dehy
8
```

This `(empty list or set)` reply from `REDE.POLL` means that the there are no more items to pull right now, so we'll have to wait until enough time passes for our next element to expire. using `REDE.TTN` we can see this will be in 8 seconds (in this example we waited a bit between commands). Once 8 seconds will pass we can run:

```
127.0.0.1:9979> REDE.POLL some_dehy
1) "world"
127.0.0.1:9979> REDE.TEST
PASS
(15.00s)
127.0.0.1:9979> DEL some_dehy
OK
```

External usage examples

helloworld.py
usage examples of most commands

test.py
functional and performance tests for the module.

More Resources

- Our homepage: https://tamarlabs.github.io/ReDe
- Our Git Repo: https://github.com/tamarlabs/ReDe
- Command documentation and examples: https://github.com/tamarlabs/ReDe/blob/master/Commands.md
- A detailed desciption of the Algorithm: https://github.com/tamarlabs/ReDe/blob/master/Algorithm.md
- The Article "Fast Data": https://goo.gl/DDFFPO
- The Null Terminator Blog: www.nullterminator.org

Enjoy!

adam@tamarlabs.com