redis-throttled-queue

Release 1.0.0

Ionel Cristian Mărieș

Feb 14, 2024

CONTENTS

CHAPTER

ONE

OVERVIEW

docs
tests
package

Queue system with key-based throttling implemented over Redis.

- Free software: BSD 2-Clause License

1.1 Installation

```
pip install redis-throttled-queue
```

You can also install the in-development version with:

```
pip install https://github.com/ionelmc/python-redis-throttled-queue/archive/main.zip
```

1.2 Documentation

https://python-redis-throttled-queue.readthedocs.io/

1.3 Development

To run all the tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER

TWO

INSTALLATION

At the command line:

```
pip install redis-throttled-queue
```

CHAPTER

THREE

USAGE

To use redis-throttled-queue in a project:

```
import redis_throttled_queue
```

CHAPTER

FOUR

REFERENCE

4.1 redis_throttled_queue

class redis_throttled_queue.**AsyncThrottledQueue**(**args*, ***kwargs*)

Asyncio variant of the queue.

__init__(**args*, ***kwargs*)

Overrides certain options because they cannot work anymore: `validate_version=False`, `register_library=False`.

async cleanup()

Asyncio variant for `cleanup`.

async pop(*window: str | bytes | int = Ellipsis*) → str | bytes | None

Asyncio variant for `pop`.

async push(*name: str*, *data: str | bytes*, ***, *priority: int = 0*)

Asyncio variant for `push`.

async classmethod register_library(*redis_client: Redis*)

You have to call this manually.

async size()

Asyncio variant for `__len__`.

async validate_version()

You have to call this manually.

enum redis_throttled_queue.**Resolution**(*value*)

Member Type
`int`

Valid values are as follows:

SECOND = <Resolution.SECOND: 1>

MINUTE = <Resolution.MINUTE: 60>

class redis_throttled_queue.**ThrottledQueue**(*redis_client: Redis*, *prefix: str*, *limit: int = 10*, *resolution=Resolution.SECOND*, *validate_version=True*, *register_library=True*)

Queue system with key-based throttling implemented over Redis.

Publishers push given a key.

Consumers pop one item at a time for the first key that has not exceeded the throttling limit withing the resolution window.

__init__(*redis_client: Redis*, *prefix: str*, *limit: int = 10*, *resolution=Resolution.SECOND*, *validate_version=True*, *register_library=True*)

Parameters

- **redis_client** – An instance of `StrictRedis`.
- **prefix** – Redis key prefix.
- **limit** – Throttling limit. The queue won't retrieve more items in the given resolution for a given *key*.
- **resolution** – Resolution to use. This decides how many time window keys you will have in Redis.

__len__()

Get queue length. :return:

cleanup()

Cleanup all associated redis data to this queue.

classmethod ensure_supported_redis(*info: dict*)

Redis version validator (must be >=7). Called from `__init__`, if enabled.

property idle_seconds: float

Idle time counter.

last_activity: float

limit: int

pop(*window: str | bytes | int = Ellipsis*) → str | bytes | None

Pop an item, if any available.

push(*name: str*, *data: str | bytes*, ***, *priority: int = 0*)

Push an item.

classmethod register_library(*redis_client: Redis*)

Registers the redis functions. Called from `__init__`, if enabled.

resolution: int

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.2 Documentation improvements

redis-throttled-queue could always use more documentation, whether as part of the official redis-throttled-queue docs, in docstrings, or even on the web in blog posts, articles, and such.

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/ionelmc/python-redis-throttled-queue/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

5.4 Development

To set up *python-redis-throttled-queue* for local development:

1. Fork python-redis-throttled-queue (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:YOURGITHUBNAME/python-redis-throttled-queue.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`).
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel*:

```
tox -p auto
```

CHAPTER

SIX

AUTHORS

- Ionel Cristian Mărieș - http://blog.ionelmc.ro

CHAPTER

SEVEN

CHANGELOG

7.1 1.0.0 (2022-11-15)

- Switched from eval scripts to redis functions. Minimum Redis server version becomes 7.0.
- Replaced unpack calls with direct indexing in the Lua functions.
- These changes improve the push operation by at least 6%.

7.2 0.6.0 (2022-07-06)

- Simplified `pop()` code to avoid the expensive scan operations. The `'...:names` key is now a sorted set and will be used as a template for the usage keys (`'...:usage:<window>'`).

7.3 0.5.0 (2022-06-28)

- Added support in a `AsyncThrottledQueue` class that only differs a bit from the regular `ThrottledQueue`:
  - `__len__` is removed, instead a awaitable `size()` method is available.
  - `__init__` doesn't validate version anymore, instead you can await on `validate_version()`.
  - `push()`, `pull()` and `cleanup()` are awaitable.
- Added a `validate_version` argument to `ThrottledQueue` (default: `True`).

7.4 0.4.4 (2022-05-09)

- Fixed missing usage key expiration when some queues are empty.

7.5 0.4.3 (2022-04-09)

- Fixed buggy counts when duplicate values are pushed. For now the highest priority will be used when two identical values would be pushed.

7.6 0.4.2 (2022-04-02)

- Refactored some duplicated code in the *pop* script.

7.7 0.4.1 (2022-03-31)

- Fixed bogus error in `cleanup()` when db is completely empty.

7.8 0.4.0 (2022-03-31)

- Added `last_activity` and `idle_seconds` attributes.
- Added a `cleanup()` method.

7.9 0.3.1 (2022-03-31)

- Renamed attributes (should be safe to mess with):
  - `_limit` becomes `limit`.
  - `_resolution` becomes `resolution`.

7.10 0.3.0 (2022-03-31)

- Allowed `pop(window)` using any window value (str/bytes/int recommended tho).

7.11 0.2.0 (2022-03-31)

- Fixed `__len__` (was returning a string).

7.12 0.1.0 (2022-03-30)

- First release on PyPI.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

Symbols

A

C

E

I

L

M

P

R

S

T

V