distcache Documentation

Release 0.1

Wasim

Jul 30, 2020

Contents

Distcache is a open-source distributed in-memory cache and database. Operations are mostly asynchronous to achieve high performance. It is implemented purely in python without any external dependency which should make it easier to install and get started with. One of design goal of this project is ease of use and less cognitive load to users of similar caching/database systems like Redis, Memcached.

CHAPTER 1

Features

1. **Data types supported:**

    - All basic data types and their combination. For instance, int, str, dict, set, tuple, list, etc and

    objects that have only these types as their attributes are supported. - You can even read any object (image, pdfs, etc) in binary format and save them as key, value pair.

2. Key cache operations are logged so when the server fails, the cache can be reconstructed from the log files.
3. The APIs are similar to Memcached and Redis to reduce cognitive when migrating between platforms.
4. Since, distcache has pure python implementation the installation process should painless. It's makes it easier to get started up and running.
5. Its' architecture assumes that the cache clients and servers can fail and plans for it. The impact is minimal on adding and removing cache servers.
6. Snapshot the servers at regular intervals to avoid cold starts upon crash or planned shutdowns
7. Log replays also available for slow but complete reconstruction of the cache upon server crash, error or shutdown.
8. Thread safe increment and decrement operations on keys.
9. Health of the cache servers is monitored by the client.

CHAPTER 2

Install

```
pip install distcache
```

CHAPTER 3

Platform

- Linux
- Windows
- Python 2.7 to Python 3.5

CHAPTER 4

Quick Start

client.py

```
from distcache.cache_client import CacheClient

client = CacheClient()

# Cache operations
client.set("brazil", "football")
client.set("harry", "potter")
client.set(1, 2)
client.set(3, 6)
client.set("hey", "hola")
client.get("hey")
client.get(1)
client.set("hey", "there")
client.get("hey")
client.delete(3)
client.get(3)
client.get("brazil")
```

server.py

```
from distcache.cache_server import CacheServer

server = CacheServer('localhost', 2050)
```

Note: Make sure to include the address of the server in the self.server_pool list in config.py file.

Run server.py and client.py.

CHAPTER 5

APIs

All distcache APIs

CHAPTER 6

Usage

1. You have multiple servers serving users. And you need to increment user id across the server such that there is no duplicate. Similarly you can store product_id, session_id, user_id, etc.

2. 80% of the database access is generated by 20% of the queries. You should absolutely not be doing duplicate computations. Cache the results. And, the figures vary and you can still use caching service. It just makes things faster.