**Redis Selective Backup and Restore (RSBR)**

This design and document can serve as either alternative or supplemental to Redis sentinel. The purpose of this design is to backup important data selectively instead of backup the whole database like Redis sentinels. The purpose of this design is to enable database resorting based on the timestamp of the data. Intuitively speaking, it should function like windows system recovery based on a date.

Schema design:

In this example, we use port 6379 as the main database port and 6380 as backup database port. The backup can be executed in two ways: backup data to 6380 when the data is created at 6379, or backup data to 6380 after data is initiated in 6379. Here we only discuss the later case where backup is NOT executed at the same time as it is stored. This way is preferred since concurrent requests tend to be less stable than sequential operation.

6380 backup database will not save the exact data info as the data appeared in 6379. Instead, it will create backup type based on the timestamp the backup action is taken. To futher demonstrate this, we take this example:

original data in 6379:

|  |  |  |
| --- | --- | --- |
| Key | Value | Type |
| test | hello | string |

backup schema in 6380:

|  |  |  |
| --- | --- | --- |
| Key | Value | Type |
| test: 1568659962728 | hello | string |
| test: stamp | 1568659962728 | list |

The number 1568659962728 is timestamp hash created based on the current time, this can be replaced by any form of timestamp. However, it has to follow the rule that it is always incrementing. Basically, the backup data will be created with time stamp in 6380 alone with a list tracking all the related time stamp. This schema can be extended to store multiple copy of data based on different time. For example:

Action on 6379:

->set test HELLO

backup action executed to 6380

->set test BYE

backup action executed to 6380

->set test GOOD

backup action executed to 6380

schema in 6380:

|  |  |  |
| --- | --- | --- |
| Key | Value | Type |
| test: 1568659724506 | HELLO | string |
| test: 1568659758045 | BYE | string |
| test: 1568659962728 | GOOD | string |
| Test:stamp | [1568659724506, 1568659758045, 1568659962728] | list |

Here we define the maximum of copy of one data as buffer limit. The default buffer limit in the code implementation is set to 3. Here we have 3 copies of data backup in different time and now they can be used to restore data based on time. When the data is backup for 4th time, the earliest backup (test: 1568659724506) will be popped so the buffer limit is kept to 3.

…