Redis客户端使用指南

Summary

版本	发布的	更改由	注释
当前 (v. 3)	**2022-09-27 10:57**	donghong.huang@shopee.com	补充quick start
v. 2	2022-05-31 12:44	donghong.huang@shopee.com	
v. 1	2022-05-31 12:13	donghong.huang@shopee.com	

快速开始

使用步骤

redis的使用步骤，可以简单概括为以下4步：

1. 初始化。必须且只需初始化1次。
2. 获取连接。
3. 使用第2步获取到的连接执行redis命令并获取结果。
4. 关闭连接。本质上只是将连接归还给连接池，不一定会在物理上释放连接。

redis_example.go

```
import "git.garena.com/shopee/bg-logistics/go/gocommon/redis"

func main() {
        //1.
        err := redis.Init(
                10, //
                10, //
                "tcp", //"tcp"
                "127.0.0.1:6379", //redis
        )
        if err != nil {
                panic(err)
        }

        //2. redis
        conn := redis.GetRConn()
        defer conn.Close() //4.deferpanic

        //3. redisLINDEX
        data, err := conn.LIndex(context.Background(), "my.test.list", 2)
        if err != nil {
                panic(err)
        }
}
```

进阶使用

在chassis中使用redis

主要区别

在chassis中使用redis与直接使用redis的主要区别有2点:

1. chassis自动根据redis的配置完成redis客户端的初始化。
2. chassis会自动为每一个redis命令申请一个连接，并且在该redis命令响应后自动释放连接。

首先，在配置文件中配置redis客户端所需的一些信息:

chassis.yaml

```
chassis:
  application:
    name: example_CacheServer
  service:
    rest:
      listenAddress: 127.0.0.1:7334
  plugins:
    cache:
      default:
        type: redis
        maxIdleConns: 10
        maxActiveConns: 10
        dialAddress: "redis://127.0.0.1:6379/2" #dialAddressredis://[[user][:pass]@]host[:port][/db]
```

配置完成后，就可以在代码中使用redis。redis的管理组件在chassis中称为“CacheHandler”。

example.go

```
import (
        "context"
        "git.garena.com/shopee/bg-logistics/go/chassis"
        "git.garena.com/shopee/bg-logistics/go/chassis/handler"
)

func main() {
        handler.RegisterCacheHandler() //cache handlerchassis.Init
        err := chassis.Init(chassis.WithDefaultProviderHandlerChain(
                handler.CacheHandler, //cache handler
        ))
        if err != nil {
                panic(err)
        }
}

func DoSomeBusinessWithRedis(ctx context.Context) {
        data, err := chassis.RedisCacheFromContext(ctx). //redis
                 LIndex(context.Background(), "my.test.list", 2) //redis
}
```

带参数初始化redis

redis初始化时，除了上述固定的参数，还有一些可变的选项可以选择。

example.go

```
err := redis.Init(
                10, //
                10, //
                "tcp", //"tcp"
                "127.0.0.1:6379", //redis,
                 WithCodisDB(2), //redis
                 WithPassword("123456"), //redis""
                 WithWait(), //redis
                 WithShadowAddress("shadow.redis.cluster:6679"), //
        )
```

访问多个redis集群/实例

如果一个应用，需要访问多个不同的redis集群或者实例，那么，可以为每一个集群或者实例创建一个连接池。

example.go

```
poolA, err := redis.NewPool(10, 10, "tcp", "127.0.0.1:6379") //redis.Init
if err != nil {
        panic(err)
}

poolB, err := redis.NewPool(10, 10, "tcp", "127.0.0.1:6380") //redis.Init
if err != nil {
        panic(err)
}
```

等待redis空闲连接

默认情况

默认情况下，当活跃的连接数达到了初始化时指定的最大活跃连接数，当尝试获取新的连接时，都会得到一个bad conn。bad conn发送任何redis命令都会得到一个错误。

如果希望在并发比较高的情况下，允许等待，从而避免影响接口可用率，可以在初始化时使用WithWait参数。

example.go

```
err := redis.Init(10,10,"tcp","127.0.0.1:6379", WithWait())
if err != nil {
        panic(err)
}

conn := redis.GetRConn() //

ctx, cancel := context.WithTimeout(context.Background(), 3*time.Second)
conn = redis.GetCtxRConn(ctx) //ctx timedoutcanceled
```

附录

Redis命令中的key的识别规则

识别key的用途

Redis客户端识别redis命令中包含的key，目前主要有2个作用：

1. 内置的CAT监控上报的data里，会包含Redis命令中所有的key；
2. 处理由影子请求产生的Redis命令时，会自动为命令中所有的key增加“_shadow"后缀

目前，redis客户端能够正确识别key的redis命令，以及识别方式，如下表所示：

redis命令的范围

由于目前公司使用的是codis，因此以下命令只包含codis支持的命令

type	cmd	命令格式	命令key数量	key位置	备注
key	DEL	DEL key [key ...]	n	1	
key	DUMP	DUMP key	1	1	
key	EXISTS	EXISTS key	1	1	
key	EXPIRE	EXPIRE key seconds	1	1	
key	EXPIREAT	EXPIREAT key timestamp	1	1	
key	MOVE	MOVE key db	1	1	
key	PERSIST	PERSIST key	1	1	
key	PEXPIRE	PEXPIRE key milliseconds	1	1	
key	PEXPIREAT	PEXPIREAT key milliseconds-timestamp	1	1	
key	PTTL	PTTL key	1	1	
key	RESTORE	RESTORE key ttl serialized-value	1	1	
key	SORT	SORT key [BY pattern] [LIMIT offset count] [GET pattern [GET pattern ...]] [ASC | DESC] [ALPHA] [STORE destination]	1-2	1,k	至少包含一个key，另外一个key可选，由STORE参数指定
key	TTL	TTL key	1	1	
key	TYPE	TYPE key	1	1	
string	APPEND	APPEND key value	1	1	
string	BITCOUNT	BITCOUNT key [start] [end]	1	1	
string	DECR	DECR key	1	1	
string	DECRBY	DECRBY key decrement	1	1	
string	GET	GET key	1	1	
string	GETBIT	GETBIT key offset	1	1	
string	GETRANGE	GETRANGE key start end	1	1	
string	GETSET	GETSET key value	1	1	
string	INCR	INCR key	1	1	
string	INCRBY	INCRBY key increment	1	1	
string	INCRBYFLOAT	INCRBYFLOAT key increment	1	1	
string	MGET	MGET key [key ...]	n	1-n	
string	MSET	MSET key value [key value ...]	n	1,3,5,...	参数以k,v,k,v,...的形式出现，因此计数位置的都是key
string	PSETEX	PSETEX key milliseconds value	1	1	
string	SET	SET key value [EX seconds] [PX milliseconds] [NX|XX]	1	1	
string	SETBIT	SETBIT key offset value	1	1	

string	SETEX	SETEX key seconds value	1	1	
string	SETNX	SETNX key value	1	1	
string	SETRANGE	SETRANGE key offset value	1	1	
string	STRLEN	STRLEN key	1	1	
hash	HDEL	HDEL key field [field ...]	1	1	
hash	HEXISTS	HEXISTS key field	1	1	
hash	HGET	HGET key field	1	1	
hash	HGETALL	HGETALL key	1	1	
hash	HINCRBY	HINCRBY key field increment	1	1	
hash	HINCRBYFLOAT	HINCRBYFLOAT key field increment	1	1	
hash	HKEYS	HKEYS key	1	1	
hash	HLEN	HLEN key	1	1	
hash	HMGET	HMGET key field [field ...]	1	1	
hash	HMSET	HMSET key field value [field value ...]	1	1	
hash	HSET	HSET key field value	1	1	
hash	HSETNX	HSETNX key field value	1	1	
hash	HVALS	HVALS key	1	1	
hash	HSCAN	HSCAN key cursor [MATCH pattern] [COUNT count]	1	1	
list	LINDEX	LINDEX key index	1	1	
list	LINSERT	LINSERT key BEFORE|AFTER pivot value	1	1	
list	LLEN	LLEN key	1	1	
list	LPOP	LPOP key	1	1	
list	LPUSH	LPUSH key value [value ...]	1	1	
list	LPUSHX	LPUSHX key value	1	1	
list	LRANGE	LRANGE key start stop	1	1	
list	LREM	LREM key count value	1	1	
list	LSET	LSET key index value	1	1	
list	LTRIM	LTRIM key start stop	1	1	
list	RPOP	RPOP key	1	1	
list	RPOPLPUSH	RPOPLPUSH source destination	2	1-2	
list	RPUSH	RPUSH key value [value ...]	1	1	
list	RPUSHX	RPUSHX key value	1	1	
set	SADD	SADD key member [member ...]	1	1	
set	SCARD	SCARD key	1	1	
set	SDIFF	SDIFF key [key ...]	n	1-n	
set	SDIFFSTORE	SDIFFSTORE destination key [key ...]	n	1-n	
set	SINTER	SINTER key [key ...]	n	1-n	
set	SINTERSTORE	SINTERSTORE destination key [key ...]	n	1-n	
set	SISMEMBER	SISMEMBER key member	1	1	
set	SMEMBERS	SMEMBERS key	1	1	
set	SMOVE	SMOVE source destination member	2	1-2	
set	SPOP	SPOP key	1	1	

set	SRANDMEMBER	SRANDMEMBER key [count]	1	1	
set	SREM	SREM key member [member ...]	1	1	
set	SUNION	SUNION key [key ...]	n	1-n	
set	SUNIONSTORE	SUNIONSTORE destination key [key ...]	n	1-n	
set	SSCAN	SSCAN key cursor [MATCH pattern] [COUNT count]	1	1	
sorted_set	ZADD	ZADD key score member [[score member] [score member] ...]	1	1	
sorted_set	ZCARD	ZCARD key	1	1	
sorted_set	ZCOUNT	ZCOUNT key min max	1	1	
sorted_set	ZINCRBY	ZINCRBY key increment member	1	1	
sorted_set	ZRANGE	ZRANGE key start stop [WITHSCORES]	1	1	
sorted_set	ZRANGEBYSCORE	ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset count]	1	1	
sorted_set	ZRANK	ZRANK key member	1	1	
sorted_set	ZREM	ZREM key member [member ...]	1	1	
sorted_set	ZREMRANGEBYRANK	ZREMRANGEBYRANK key start stop	1	1	
sorted_set	ZREMRANGEBYSCORE	ZREMRANGEBYSCORE key min max	1	1	
sorted_set	ZREVRANGE	ZREVRANGE key start stop [WITHSCORES]	1	1	
sorted_set	ZREVRANGEBYSCORE	ZREVRANGEBYSCORE key max min [WITHSCORES] [LIMIT offset count]	1	1	
sorted_set	ZREVRANK	ZREVRANK key member	1	1	
sorted_set	ZSCORE	ZSCORE key member	1	1	
sorted_set	ZUNIONSTORE	ZUNIONSTORE destination numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX]	n	1,3-m	key的数量由numkeys参数指定
sorted_set	ZINTERSTORE	ZINTERSTORE destination numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX]	n	1,3-m	key的数量由numkeys参数指定
sorted_set	ZSCAN	ZSCAN key cursor [MATCH pattern] [COUNT count]	1	1	
script	EVAL	EVAL script numkeys key [key ...] arg [arg ...]	n	3-m	key的数量由numkeys参数指定
script	EVALSHA	EVALSHA sha1 numkeys key [key ...] arg [arg ...]	n	3-m	key的数量由numkeys参数指定
connection	AUTH	AUTH password	0	-	不包含任何key的命令，上报时会把所有参数上报，但不对参数做影子处理
connection	ECHO	ECHO message	0	-	同上
connection	PING	PING	0	-	同上
connection	QUIT	QUIT	0	-	同上
connection	SELECT	SELECT index	0	-	同上
server	INFO	INFO [section]	0	-	同上

redis客户端目前不支持的命令

不支持的命令

下表是Redis客户端不支持提取Key和对Key进行影子处理的命令，请不要在正式代码中使用

type	cmd		命令格式
key	KEYS	KEYS	KEYS pattern
key	MIGRATE	MIGRATE	MIGRATE host port key destination-db timeout [COPY] [REPLACE]
key	OBJECT	OBJECT	OBJECT subcommand [arguments [arguments]]
key	RANDOMKEY	RANDOMKEY	RANDOMKEY
key	RENAME	RENAME	RENAME key newkey
key	RENAMENX	RENAMENX	RENAMENX key newkey
key	SCAN	SCAN	SCAN cursor [MATCH pattern] [COUNT count]
string	BITOP	BITOP	BITOP operation destkey key [key ...]
string	MSETNX	MSETNX	MSETNX key value [key value ...]
list	BLPOP	BLPOP	BLPOP key [key ...] timeout
list	BRPOP	BRPOP	BRPOP key [key ...] timeout
list	BRPOPLPUSH	BRPOPLPUSH	BRPOPLPUSH source destination timeout
pub_sub	PSUBSCRIBE	PSUBSCRIBE	PSUBSCRIBE pattern [pattern ...]
pub_sub	PUBLISH	PUBLISH	PUBLISH channel message
pub_sub	PUBSUB	PUBSUB	PUBSUB &lt;subcommand&gt; [argument [argument ...]]
pub_sub	PUNSUBSCRIBE	PUNSUBSCRIBE	PUNSUBSCRIBE [pattern [pattern ...]]
pub_sub	SUBSCRIBE	SUBSCRIBE	SUBSCRIBE channel [channel ...]
pub_sub	UNSUBSCRIBE	UNSUBSCRIBE	UNSUBSCRIBE [channel [channel ...]]
transaction	DISCARD	DISCARD	DISCARD
transaction	EXEC	EXEC	EXEC
transaction	MULTI	MULTI	MULTI
transaction	UNWATCH	UNWATCH	UNWATCH
transaction	WATCH	WATCH	WATCH key [key ...]
script	SCRIPT_EXISTS	SCRIPT_EXISTS	SCRIPT EXISTS script [script ...]
script	SCRIPT_FLUSH	SCRIPT_FLUSH	SCRIPT FLUSH
script	SCRIPT_KILL	SCRIPT_KILL	SCRIPT KILL
script	SCRIPT_LOAD	SCRIPT_LOAD	SCRIPT LOAD script
server	BGREWRITEAOF	BGREWRITEAOF	BGREWRITEAOF
server	BGSAVE	BGSAVE	BGSAVE
server	CLIENT_GETNAME	CLIENT_GETNAME	CLIENT GETNAME
server	CLIENT_KILL	CLIENT_KILL	CLIENT KILL ip:port
server	CLIENT_LIST	CLIENT_LIST	CLIENT LIST
server	CLIENT_SETNAME	CLIENT_SETNAME	CLIENT SETNAME connection-name
server	CONFIG_GET	CONFIG_GET	CONFIG GET parameter
server	CONFIG_RESETSTAT	CONFIG_RESETSTAT	CONFIG RESETSTAT
server	CONFIG_REWRITE	CONFIG_REWRITE	CONFIG REWRITE
server	CONFIG_SET	CONFIG_SET	CONFIG SET parameter value
server	DBSIZE	DBSIZE	DBSIZE

server	DEBUG_OBJECT	DEBUG_OBJECT	DEBUG OBJECT key
server	DEBUG_SEGFAULT	DEBUG_SEGFAULT	DEBUG SEGFAULT
server	FLUSHALL	FLUSHALL	FLUSHALL
server	FLUSHDB	FLUSHDB	FLUSHDB
server	LASTSAVE	LASTSAVE	LASTSAVE
server	MONITOR	MONITOR	MONITOR
server	PSYNC	PSYNC	PSYNC <MASTER_RUN_ID> <OFFSET>
server	SAVE	SAVE	SAVE
server	SHUTDOWN	SHUTDOWN	SHUTDOWN
server	SLAVEOF	SLAVEOF	SLAVEOF host port
server	SLOWLOG	SLOWLOG	SLOWLOG subcommand [argument]
server	SYNC	SYNC	SYNC
server	TIME	TIME	TIME