与「十」俱进，码出未来

2018源创会年终盛典

主办方 开源中国 oschina.net

Redis 5.0 新特性解读

张冬洪 （极数云舟）

目录

- 个人介绍
- Redis 5.0 新特性介绍
- Redis 5.0 Stream 数据类型小试牛刀

个人介绍

- 极数云舟对外合作部总监、极数学院联合创始人、极数云舟数据库架构师
- Redis中国用户组（CRUG）联合创始人兼主席
- 曾服务于世纪佳缘、新媒传信和新浪微博，任高级DBA
- 阿里云MVP、腾讯云TVP

Redis 5.0 新特性介绍

Redis 特点

- 单进程
- 高可用
- 支持持久化
- 分布式
- 丰富的数据结构
- 高性能
- 低延迟
- 自定义功能

Redis 4.0 新特性

- 支持Module，方便扩展自定义新功能
- PSYNC 2.0，优化主从切换全量复制问题
- 提供LFU（Last Frequently Used）算法，优化缓存剔除
- 非阻塞DEL、FLUSHALL、FLUSHDB，避免阻塞
- 提供AOF-RDB混合持久化策略，数据更安全可靠
- 提供memory命令，内存监控统计更全面
- 交换数据库功能swapdb，减少select操作命令
- Redis Cluster兼容NAT和Docker
- 引入Jemalloc库，优化内存访问

Redis 5.0 新特性

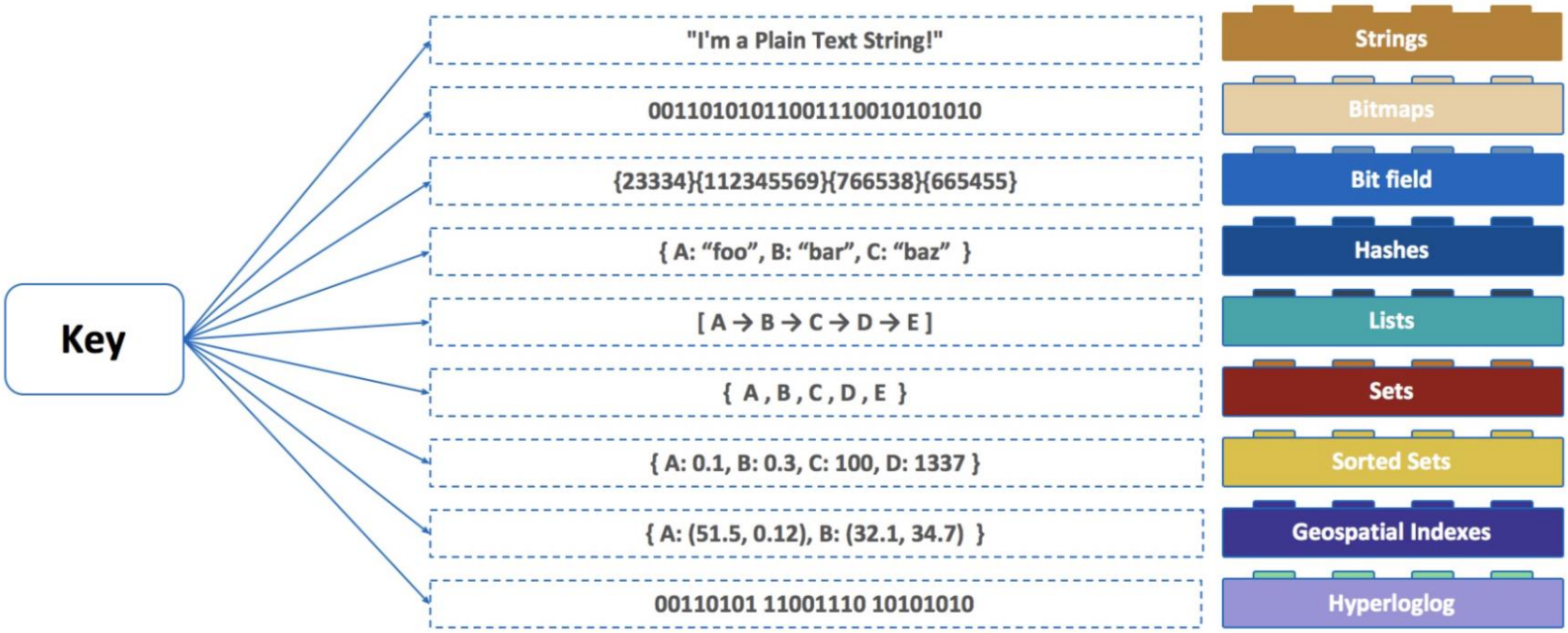
1. The new Stream data type. https://redis.io/topics/streams-intro
2. New Redis modules APIs: Timers and Cluster API.
3. RDB now store LFU and LRU information.
4. The cluster manager was ported from Ruby (redis-trib.rb) to C code inside redis-cli. Check `redis-cli --cluster help` for more info.
5. New sorted set commands: ZPOPMIN/MAX and blocking variants.
6. Active defragmentation version 2.
7. Improvemenets in HyperLogLog implementations.
8. Better memory reporting capabilities.
9. Many commands with sub-commands now have an HELP subcommand.
10. Better performances when clients connect and disconnect often.
11. Many bug fixes and other random improvements.
12. Jemalloc was upgraded to version 5.1

https://raw.githubusercontent.com/antirez/redis/5.0/00-RELEASENOTES

Redis 5.0

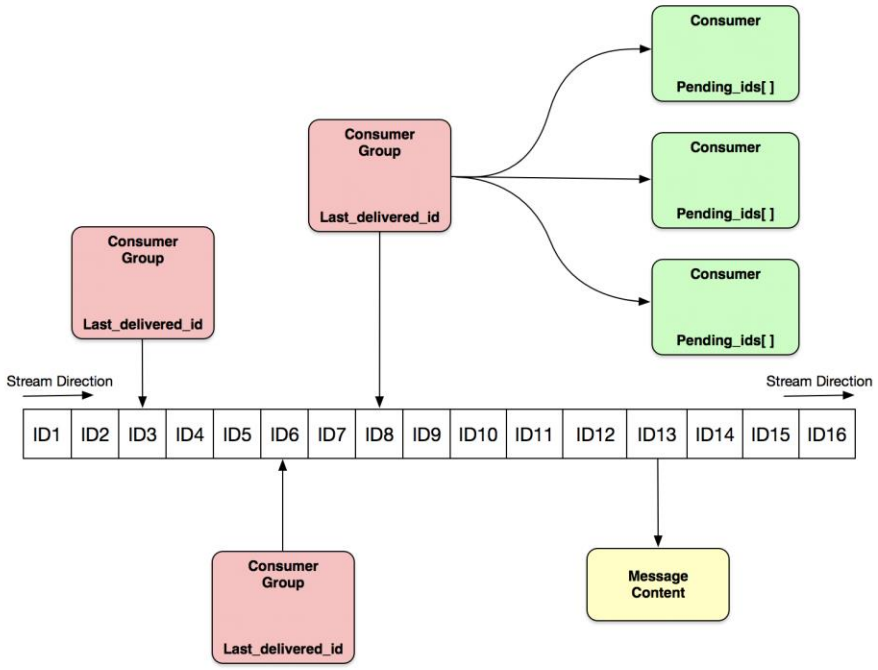
Stream 数据类型小试牛刀

Redis 数据结构



Stream 它是一个新的强大的支持多播的可持久化的消息队列

基本概念



- Redis Stream：一个消息链表、每个stream都有唯一名称

  就是Redis key，在首次使用xadd指令时自动创建

- 消息：由唯一的ID和对应的内容组成
- 持久化：消息是持久化的，存储方式跟其他类型一样
- 消息ID： timestampInMillis-sequence
  - 1527846880572-5 表示该毫秒内产生的第5条消息
  - 消息ID可以由服务器产生，也可以由客户端指定
  - 形式必须是：整数 – 整数，后加入的消息ID要大
  - 很多操作都跟该ID有关系，故重要
- 消息内容：键值对，如：name donghong

基本使用 - 独立消费（类普通消息队列list）

- xadd　追加消息
- xdel　删除消息，这里的删除仅仅是设置了标志位，不影响消息总长度（已修复）
- xrange 获取消息列表，会自动过滤已经删除的消息
- xlen　获取消息长度
- del　删除Stream
- xread　消费指令，消费消息；当stream 没有消息时，可以阻塞等待

DEMO

- 追加消息

```
127.0.0.1:6379>
127.0.0.1:6379> xadd hello * name zhangdh age 29
1533213308815-0
127.0.0.1:6379> xadd hello * name ziwen age 27
1533213332255-0
127.0.0.1:6379> xadd hello * name changjin age 28
1533213357975-0
127.0.0.1:6379>
```

*号表示服务器自动生成ID，后面顺序跟着key/value　　也可以指定ID

```
127.0.0.1:6379> xadd hello 1533218826424-1 name ziwen age 28
1533218826424-1
127.0.0.1:6379> xadd hello 1533218826424-2 name donghong age 28
1533218826424-2
```

- 获取消息长度

```
127.0.0.1:6379> xlen hello
(integer) 3
127.0.0.1:6379>
```

- 获取消息内容

```
127.0.0.1:6379> xrange hello - +
1) 1) 1533213308815-0
   2) 1) "name"
      2) "zhangdh"
      3) "age"
      4) "29"
2) 1) 1533213332255-0
   2) 1) "name"
      2) "ziwen"
      3) "age"
      4) "27"
3) 1) 1533213357975-0
   2) 1) "name"
      2) "changjin"
      3) "age"
      4) "28"
127.0.0.1:6379>
```

- `-` 最小值
- `+` 最大值
- 可以查询指定Id内容或者范围查找

- 删除消息和Stream

```
127.0.0.1:6379> xdel hello 1533213332255-0
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379> xlen hello
(integer) 2
127.0.0.1:6379>
127.0.0.1:6379> xrange hello - +
1) 1) 1533213308815-0
   2) 1) "name"
      2) "zhangdh"
      3) "age"
      4) "29"
2) 1) 1533213357975-0
   2) 1) "name"
      2) "changjin"
      3) "age"
      4) "28"
127.0.0.1:6379> del hello
(integer) 1
127.0.0.1:6379>
127.0.0.1:6379> xlen hello
(integer) 0
```

- 消费消息

```
127.0.0.1:6379> xread count 2 streams hello 0-0
1) 1) "hello"
   2) 1) 1) 1533213759858-0
         2) 1) "name"
            2) "zhangdh"
            3) "age"
            4) "29"
      2) 1) 1533213766069-0
         2) 1) "name"
            2) "ziwen"
            3) "age"
            4) "27"
127.0.0.1:6379> xread count 2 streams hello - +
(error) ERR Unbalanced XREAD list of streams: for each stream key an ID or '$' must be specified.
127.0.0.1:6379> xread count 1 streams hello $
(nil)
```

- 从stream头部读取两条消息
- 参数设为0-0，表示读取所有的PEL消息
- 参数$，表示从stream尾部读取一条消息

- 阻塞消费消息

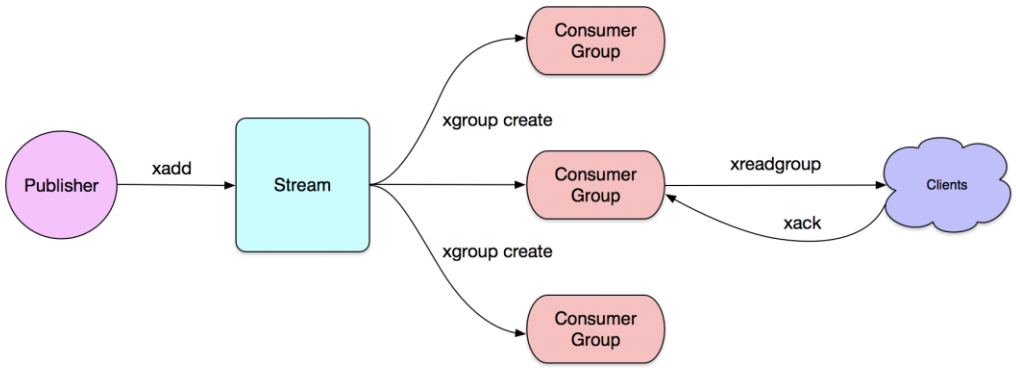
```
127.0.0.1:6379> xread block 0 count 1 streams hello $
1) 1) "hello"
   2) 1) 1) 1533214435838-0
         2) 1) "name"
            2) "changjin"
            3) "age"
            4) "28"
(181.28s)
127.0.0.1:6379> xread block 1000 count 1 streams hello $
(nil)
(1.02s)
```

..edis/redis5.0 (zsh)

```
127.0.0.1:6379> xadd hello * name changjin age 28
1533214435838-0
```

- block 0表示永远阻塞，直到消息到来
- block 1000表示阻塞1s，如果1s内没有任何消息到来，就返回nil
- 阻塞解除后，返回新的消息内容，还显示了一个等待时间

基本概念



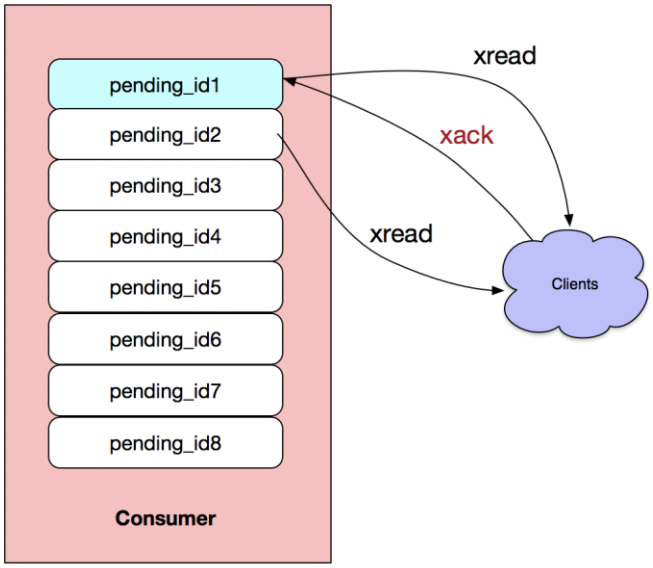
- 每个Stream都可以挂多个消费组
- 每个消费组会有个游标last_delivered_id在Stream数组之上往前移动，表示当前消费组已经消费到哪条消息了

- 每个消费组都有一个Stream内唯一的名称
- 消费组不会自动创建，它需要单独的指令xgroup create进行创建

- 需要指定从Stream的某个消息ID开始消费，这个ID用来初始化last_delivered_id变量

- 每个消费组都是独立的，互不影响，也就是说同一份Stream内部的消息会被每个消费组都消费到

基本概念



- 同一个消费组(Consumer Group)可以挂接多个消费者(Consumer)
- 这些消费者之间是竞争关系，任意一个消费者读取了消息都会使游标last_delivered_id往前移动。
- 每个消费者有一个组内唯一名称。

- 消费者(Consumer)内部会有个状态变量pending_ids，它记录了当前已经被客户端读取的消息，但是还没有ack。如果客户端没有ack，这个变量里面的消息ID会越来越多，一旦某个消息被ack，它就开始减少。这个pending_ids变量在Redis官方被称之为PEL，也就是Pending Entries List，这是一个很核心的数据结构，它用来确保客户端至少消费了消息一次，而不会在网络传输的中途丢失了没处理。

基本使用 - 消费组模式

- xgroup 用于创建，销毁和管理消费组
- xgroup create 创建消费组，需要传递起始消息Id参数用来初始 last_delivered_id
- xinfo groups 获取Stream的消费组信息
- xinfo stream 获取Stream 的信息
- xinfo consumers 获取指定消费组的消费者信息
- xreadgroup 用于通过消费组从Stream中读取消息
- xack 允许客户端将待处理的消息标记为正确处理了
- xpending 用于查询消费组中待处理的消息条目

DEMO

- 创建消费组

  - 表示从头开始消费

```
127.0.0.1:6379> xgroup create hello cg1 0-0
OK
```

  - $表示从尾部开始消费，只接受新消息，当前Stream消息会全部忽略

```
127.0.0.1:6379> xgroup create hello cg3 $
OK
```

- 从Stream里消费信息

```
127.0.0.1:6379> xreadgroup group cg1 c1 count 1 streams hello >
1) 1) "hello"
   2) 1) 1) 1533213759858-0
         2) 1) "name"
            2) "zhangdh"
            3) "age"
            4) "29"
127.0.0.1:6379> xreadgroup group cg1 c1 count 1 streams hello >
1) 1) "hello"
   2) 1) 1) 1533213766069-0
         2) 1) "name"
            2) "ziwen"
            3) "age"
            4) "27"
127.0.0.1:6379> xreadgroup group cg1 c1 count 1 streams hello >
1) 1) "hello"
   2) 1) 1) 1533214435838-0
         2) 1) "name"
            2) "changjin"
            3) "age"
            4) "28"
127.0.0.1:6379> xreadgroup group cg1 c1 count 1 streams hello >
(nil)
```

- *>号表示从当前消费组的last_delivered_id后面开始读*
- *每当消费者读取一条消息，last_delivered_id变量就会前进*

- 获取消费组信息

```
127.0.0.1:6379> XINFO groups hello
1) 1) name
   2) "cg1"
   3) consumers
   4) (integer) 1
   5) pending
   6) (integer) 7
   7) last-delivered-id
   8) 1533218826424-1
2) 1) name
   2) "cg2"
   3) consumers
   4) (integer) 0
   5) pending
   6) (integer) 0
   7) last-delivered-id
   8) 1533214435838-0
```

- 获取Stream的信息

```
127.0.0.1:6379> xinfo stream hello
 1) length
 2) (integer) 3
 3) radix-tree-keys
 4) (integer) 1
 5) radix-tree-nodes
 6) (integer) 2
 7) groups
 8) (integer) 2
 9) last-generated-id
10) 1533214435838-0
11) first-entry
12) 1) 1533213759858-0
    2) 1) "name"
       2) "zhangdh"
       3) "age"
       4) "29"
13) last-entry
14) 1) 1533214435838-0
    2) 1) "name"
       2) "changjin"
       3) "age"
       4) "28"
```

- 消息消费确认

```
127.0.0.1:6379> xinfo consumers hello cg1
1) 1) name
   2) "c1"
   3) pending
   4) (integer) 7
   5) idle
   6) (integer) 173230
127.0.0.1:6379> xack hello cg1 1533218826424-1
(integer) 1
127.0.0.1:6379> xinfo consumers hello cg1
1) 1) name
   2) "c1"
   3) pending
   4) (integer) 6
   5) idle
   6) (integer) 247837
```

- 查看消费者信息

```
127.0.0.1:6379> xinfo CONSUMERS hello cg1
1) 1) name
   2) "alice"
   3) pending
   4) (integer) 0
   5) idle
   6) (integer) 131581200
2) 1) name
   2) "c1"
   3) pending
   4) (integer) 6
   5) idle
   6) (integer) 131562297
```

- 查看待处理的消息情况

```
127.0.0.1:6379> XPENDING hello cg1
1) (integer) 6
2) 1533213759858-0
3) 1533218826424-0
4) 1) 1) "c1"
      2) "6"
```

- 待处理消息中的最小和较大消息ID
- 最后是消费者列表和待处理消息数

- 上限流

```
127.0.0.1:6379> XADD hello MAXLEN 2 * value 1
1533352347318-0
127.0.0.1:6379> XADD hello MAXLEN 2 * value 2
1533352356702-0
127.0.0.1:6379> XADD hello MAXLEN 2 * value 3
1533352358057-0
127.0.0.1:6379> XLEN hello
(integer) 2
127.0.0.1:6379> xrange hello - +
1) 1) 1533352356702-0
   2) 1) "value"
      2) "2"
2) 1) 1533352358057-0
   2) 1) "value"
      2) "3"
```

- 消息持久化

```
⚡ root@zhangdonghongdeMacBook-Pro ~/redis/redis5.0 unstable ● tail -n 50 -f appendonly.aof
hello
$3
cg3
$1
$
*7
$4
XADD
$5
hello
$6
MAXLEN
$1
2
$15
1533352347318-0
$5
value
$1
1
```

- 持久存储到AOF和RDB文件中
- 消息重要时，必须将AOF与强fsync策略一起使用
- 异步复制，在故障转移之后，可能会丢失某些内容，具体取决于从服务器从主服务器接收数据的能力。

Q & A