键空间通知（keyspace notification）

本文档翻译自： http://redis.io/topics/notifications 。

键空间通知功能目前仍在开发中，这个文档所描述的内容，以及功能的具体实现，可能会在未来数周内改变，敬请知悉。

功能概览

键空间通知使得客户端可以通过订阅频道或模式， 来接收那些以某种方式改动了 Redis 数据集的事件。

以下是一些键空间通知发送的事件的例子：

- 所有修改键的命令。
- 所有接收到 LPUSH 命令的键。
- 0 号数据库中所有已过期的键。

事件通过 Redis 的订阅与发布功能（pub/sub）来进行分发， 因此所有支持订阅与发布功能的客户端都可以在无须做任何修改的情况下， 直接使用键空间通知功能。

因为 Redis 目前的订阅与发布功能采取的是发送即忘（fire and forget）策略， 所以如果你的程序需要可靠事件通知（reliable notification of events）， 那么目前的键空间通知可能并不适合你： 当订阅事件的客户端断线时， 它会丢失所有在断线期间分发给它的事件。

未来将会支持更可靠的事件分发， 这种支持可能会通过让订阅与发布功能本身变得更可靠来实现， 也可能会在 Lua 脚本中对消息（message）的订阅与发布进行监听， 从而实现类似将事件推入到列表这样的操作。

事件的类型

对于每个修改数据库的操作，键空间通知都会发送两种不同类型的事件。

比如说，对 0 号数据库的键 mykey 执行 DEL 命令时， 系统将分发两条消息， 相当于执行以下两个 PUBLISH 命令：

```
PUBLISH __keyspace@0__:mykey del
PUBLISH __keyevent@0__:del mykey
```

订阅第一个频道 __keyspace@0__:mykey 可以接收 0 号数据库中所有修改键 mykey 的事件， 而订阅第二个频道 __keyevent@0__:del 则可以接收 0 号数据库中所有执行 del 命令的键。

以 keyspace 为前缀的频道被称为键空间通知（key-space notification）， 而以 keyevent 为前缀的频道则被称为键事件通知（key-event notification）。

当 del mykey 命令执行时：

- 键空间频道的订阅者将接收到被执行的事件的名字，在这个例子中，就是 del 。
- 键事件频道的订阅者将接收到被执行事件的键的名字，在这个例子中，就是 mykey 。

配置

因为开启键空间通知功能需要消耗一些 CPU ， 所以在默认配置下， 该功能处于关闭状态。

可以通过修改 `redis.conf` 文件， 或者直接使用 `CONFIG SET` 命令来开启或关闭键空间通知功能：

- 当 `notify-keyspace-events` 选项的参数为空字符串时，功能关闭。
- 另一方面，当参数不是空字符串时，功能开启。

`notify-keyspace-events` 的参数可以是以下字符的任意组合， 它指定了服务器该发送哪些类型的通知：

字符	发送的通知
K	键空间通知，所有通知以 `__keyspace@<db>__` 为前缀
E	键事件通知，所有通知以 `__keyevent@<db>__` 为前缀
g	`DEL` 、 `EXPIRE` 、 `RENAME` 等类型无关的通用命令的通知
$	字符串命令的通知
l	列表命令的通知
s	集合命令的通知
h	哈希命令的通知
z	有序集合命令的通知
x	过期事件：每当有过期键被删除时发送
e	驱逐(evict)事件：每当有键因为 `maxmemory` 政策而被删除时发送
A	参数 `g$lshzxe` 的别名

输入的参数中至少要有一个 `K` 或者 `E` ， 否则的话， 不管其余的参数是什么， 都不会有任何通知被分发。

举个例子， 如果只想订阅键空间中和列表相关的通知， 那么参数就应该设为 `Kl` ， 诸如此类。

将参数设为字符串 `"AKE"` 表示发送所有类型的通知。

命令产生的通知

以下列表记录了不同命令所产生的不同通知：

- DEL 命令为每个被删除的键产生一个 `del` 通知。
- RENAME 产生两个通知：为来源键（source key）产生一个 `rename_from` 通知，并为目标键（destination key）产生一个 `rename_to` 通知。
- EXPIRE 和 EXPIREAT 在键被正确设置过期时间时产生一个 `expire` 通知。当 EXPIREAT 设置的时间已经过期，或者 EXPIRE 传入的时间为负数值时，键被删除，并产生一个 `del` 通知。
- SORT 在命令带有 `STORE` 参数时产生一个 `sortstore` 事件。如果 `STORE` 指示的用于保存排序结果的键已经存在，那么程序还会发送一个 `del` 事件。
- SET 以及它的所有变种（SETEX 、 SETNX 和 GETSET）都产生 `set` 通知。其中 SETEX 还会产生 `expire` 通知。
- MSET 为每个键产生一个 `set` 通知。
- SETRANGE 产生一个 `setrange` 通知。
- INCR 、 DECR 、 INCRBY 和 DECRBY 都产生 `incrby` 通知。
- INCRBYFLOAT 产生 `incrbyfloat` 通知。
- APPEND 产生 `append` 通知。
- LPUSH 和 LPUSHX 都产生单个 `lpush` 通知，即使有多个输入元素时，也是如此。
- RPUSH 和 RPUSHX 都产生单个 `rpush` 通知，即使有多个输入元素时，也是如此。
- RPOP 产生 `rpop` 通知。如果被弹出的元素是列表的最后一个元素，那么还会产生一个 `del` 通知。
- LPOP 产生 `lpop` 通知。如果被弹出的元素是列表的最后一个元素，那么还会产生一个 `del` 通知。
- LINSERT 产生一个 `linsert` 通知。
- LSET 产生一个 `lset` 通知。
- LTRIM 产生一个 `ltrim` 通知。如果 LTRIM 执行之后，列表键被清空，那么还会产生一个 `del` 通知。
- RPOPLPUSH 和 BRPOPLPUSH 产生一个 `rpop` 通知，以及一个 `lpush` 通知。两个命令都会保证 `rpop` 的通知在 `lpush` 的通知之前分发。如果从键弹出元素之后，被弹出的列表键被清空，那么还会产生一个 `del` 通知。

- HSET 、 HSETNX 和 HMSET 都只产生一个 hset 通知。
- HINCRBY 产生一个 hincrby 通知。
- HINCRBYFLOAT 产生一个 hincrbyfloat 通知。
- HDEL 产生一个 hdel 通知。如果执行 HDEL 之后，哈希键被清空，那么还会产生一个 del 通知。
- SADD 产生一个 sadd 通知，即使有多个输入元素时，也是如此。
- SREM 产生一个 srem 通知，如果执行 SREM 之后，集合键被清空，那么还会产生一个 del 通知。
- SMOVE 为来源键（source key）产生一个 srem 通知，并为目标键（destination key）产生一个 sadd 事件。
- SPOP 产生一个 spop 事件。如果执行 SPOP 之后，集合键被清空，那么还会产生一个 del 通知。
- SINTERSTORE 、 SUNIONSTORE 和 SDIFFSTORE 分别产生 sinterstore 、 sunionostore 和 sdiffstore 三种通知。如果用于保存结果的键已经存在，那么还会产生一个 del 通知。
- ZINCRBY 产生一个 zincr 通知。（译注：非对称，请注意。）
- ZADD 产生一个 zadd 通知，即使有多个输入元素时，也是如此。
- ZREM 产生一个 zrem 通知，即使有多个输入元素时，也是如此。如果执行 ZREM 之后，有序集合键被清空，那么还会产生一个 del 通知。
- ZREMRANGEBYSCORE 产生一个 zrembyscore 通知。（译注：非对称，请注意。）如果用于保存结果的键已经存在，那么还会产生一个 del 通知。
- ZREMRANGEBYRANK 产生一个 zrembyrank 通知。（译注：非对称，请注意。）如果用于保存结果的键已经存在，那么还会产生一个 del 通知。
- ZINTERSTORE 和 ZUNIONSTORE 分别产生 zinterstore 和 zunionstore 两种通知。如果用于保存结果的键已经存在，那么还会产生一个 del 通知。
- 每当一个键因为过期而被删除时，产生一个 expired 通知。
- 每当一个键因为 maxmemory 政策而被删除以回收内存时，产生一个 evicted 通知。

所有命令都只在键**真的**被改动了之后，才会产生通知。

比如说，当 SREM 试图删除不存在于集合的元素时，删除操作会执行失败，因为没有真正的改动键，所以这一操作不会发送通知。

如果对命令所产生的通知有疑问， 最好还是使用以下命令， 自己来验证一下：

```
$ redis-cli config set notify-keyspace-events KEA
$ redis-cli --csv psubscribe '__key*__:*'
Reading messages... (press Ctrl-C to quit)
"psubscribe","__key*__:*",1
```

然后， 只要在其他终端里用 Redis 客户端发送命令， 就可以看到产生的通知了：

```
"pmessage","__key*__:*","__keyspace@0__:foo","set"
"pmessage","__key*__:*","__keyevent@0__:set","foo"
...
```

过期通知的发送时间

Redis 使用以下两种方式删除过期的键：

- 当一个键被访问时，程序会对这个键进行检查，如果键已经过期，那么该键将被删除。
- 底层系统会在后台渐进地查找并删除那些过期的键，从而处理那些已经过期、但是不会被访问到的键。

当过期键被以上两个程序的任意一个发现、 并且将键从数据库中删除时， Redis 会产生一个 expired 通知。

Redis 并不保证生存时间（TTL）变为 0 的键会立即被删除： 如果程序没有访问这个过期键， 或者带有生存时间的键非常多的话， 那么在键的生存时间变为 0 ， 直到键真正被删除这中间， 可能会有一段比较显著的时间间隔。

因此， Redis 产生 expired 通知的时间为过期键被删除的时候， 而不是键的生存时间变为 0 的时候。

讨论

1条评论 redis 命令参考 登录

推荐 推文 分享 评分最高

加入讨论…

通过以下方式登录 或注册一个 DISQUS 帐号

姓名

名扬天下 • 4年前

支持一下

27 • 回复 • 分享 ›

在 REDIS 命令参考 上还有

SLOWLOG — Redis 命令参考

1条评论 • 4年前

Gong Cheng — 非常感谢，这个功能非常强大

GEORADIUS — Redis 命令参考

2条评论 • 3年前

马腾飞 — 那有没有取最近前10个的接口呢

Pub/Sub（发布/订阅） — Redis 命令参考

1条评论 • 4年前

yangxiaoming — 很棒1

HyperLogLog — Redis 命令参考

3条评论 • 4年前

kyoloro — 谢谢 大大 我明白了

订阅 在您的网站上使用 Disqus添加 Disqus添加

赞助商

我们正在寻找赞助商，有意对这个网站进行赞助的朋友请联系 huangz1990@gmail.com 。