大数据库系统

3.5 Redis消息订阅功能

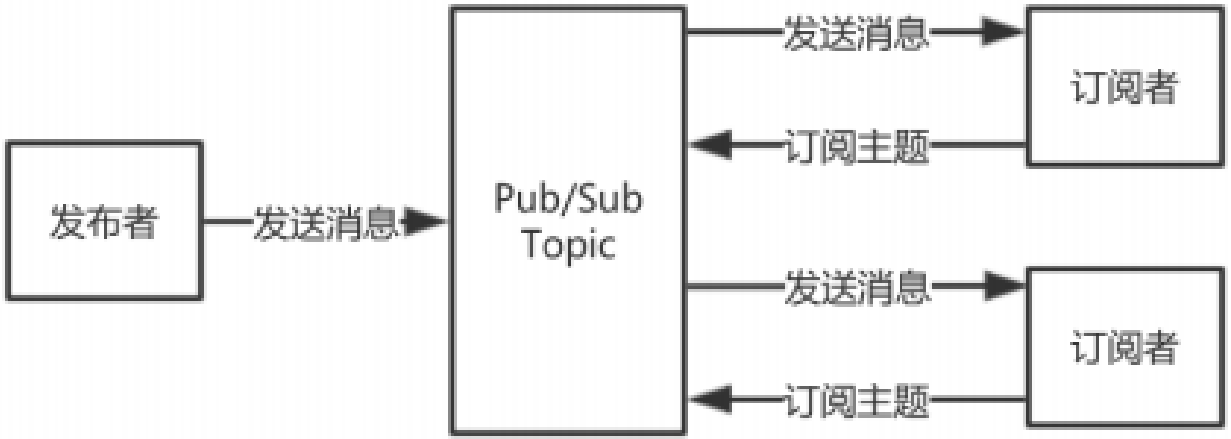
3.4.2 Redis消息订阅功能

◆ 消息订阅、发布模式

- 消息订阅发布模式是一种常用的设计模式，它具有一对多的依赖关系
- 主要包含3个角色
  - 主题（Topic）
  - 订阅者（Subscriber）
  - 发布者（Publisher）
- 多个订阅者对象同时监听某个发布者发布的主题对象[1]
- 当这个主题的状态发生变化时，所有订阅者对象都会收到通知，使它们自动更新自己的状态

3.4.2 Redis消息订阅功能

- 消息订阅发布模式
  - 主题就是一条消息内容
  - 发布者与订阅者具有一对多的关系，它们之间存在依赖性，订阅者必须订阅主题后才能接收到发布者发布的消息[2]
  - 例：网游服务器管理者发布消息，比如过多久要维护，大家做好准备



3.4.2 Redis消息订阅功能

- PUBLISH命令

  PUBLISH channel message

  消息发布者将消息发送给指定的频道，返回一个整数，表示接收到这条消息的客户端的数量

- SUBSCRIBE命令

  SUBSCRIBE channel [channel …]

  客户端订阅指定的消息频道channel

- UNSUBSCRIBE命令

  UNSUBSCRIBE channel [channel …]

  取消订阅指定的频道

例：开启两个终端，终端1发布频道news，并在频道里发布消息，终端2收听频道news的消息

1、终端1将“today is sunshine”发布到频道news

```
redis 127.0.0.1:6379> publish news 'today is sunshine'
(integer) 0
redis 127.0.0.1:6379>
```

终端1

2、开启另一个终端，订阅news频道，可以看到执行subscribe后一直处于等待接收消息的状态

```
redis 127.0.0.1:6379> subscribe news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
```

终端2

为什么没有得到“today is sunshine”？

因为“today is sunshine”在订阅前发送的

3、在终端1在发布“still sunshine”，发现返回值为1，说明有一个用户接收了这个消息

```
redis 127.0.0.1:6379> publish news 'still sunshine'
(integer) 1
redis 127.0.0.1:6379>
```

终端1

4、观察终端2，成功接收到频道news的“still sunshine”

```
redis 127.0.0.1:6379> subscribe news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
1) "message"
2) "news"
3) "still sunshine"
```

终端2

刚刚讲到，可以多个用户订阅1个频道

例2：再打开一个终端，订阅news频道

```
redis 127.0.0.1:6379> subscribe news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
```

终端3

回到终端1，在news频道发布“always sunshine”

```
redis 127.0.0.1:6379> publish news 'always sunshine'
(integer) 2
redis 127.0.0.1:6379>
```

终端1

返回值变为了2，说明两个用户收到消息

```
redis 127.0.0.1:6379> subscribe news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
1) "message"
2) "news"
3) "still sunshine"
1) "message"
2) "news"
3) "always sunshine"
```

终端2

```
redis 127.0.0.1:6379> subscribe news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
1) "message"
2) "news"
3) "always sunshine"
```

终端3

3.4.2 Redis消息订阅功能

◆ PSUBSCRIBE命令

PSUBSCRIBE pattern [pattern …]

根据指定的模式来订阅符合这个模式的频道，可以使用通配符订阅频道

- h?llo：订阅hello、hallo和hxllo频道（?表示单个任意字符）
- h*llo：订阅hllo和heeeello频道（*表示多个任意字符，包括空字符）
- h[ae]llo：订阅hello和hallo频道，但是不能订阅hillo频道（选择[和]之间的任意一个字符）
- h\?llo:表示消息订阅者只能订阅h?llo频道

注意：这种方式是订阅符合模式的所有频道[3]

例：启动终端4，订阅包含“new”开头的所有频道

```
redis 127.0.0.1:6379> psubscribe new*
Reading messages... (press Ctrl-C to quit)
1) "psubscribe"
2) "new*"
3) (integer) 1
```

终端4

回到终端1，在news频道发布消息，返回3，说明3个用户接到消息

```
redis 127.0.0.1:6379> publish news 'three'
(integer) 3
redis 127.0.0.1:6379>
```

终端1

再在终端1，在newstop频道里发布消息，此时只有1个用户收到

```
redis 127.0.0.1:6379> publish newstop 'four'
(integer) 1
redis 127.0.0.1:6379>
```

查看各个终端

```
redis 127.0.0.1:6379> subscribe news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
1) "message"
2) "news"
3) "still sunshine"
1) "message"
2) "news"
3) "always sunshine"
1) "message"
2) "news"
3) "three"
```

终端2

```
redis 127.0.0.1:6379> subscribe news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
1) "message"
2) "news"
3) "always sunshine"
1) "message"
2) "news"
3) "three"
```

终端3

```
redis 127.0.0.1:6379> psubscribe new*
Reading messages... (press Ctrl-C to quit)
1) "psubscribe"
2) "new*"
3) (integer) 1
1) "pmessage"
2) "new*"
3) "news"
4) "three"
1) "pmessage"
2) "new*"
3) "newstop"
4) "four"
```

终端4

总结

- 主要内容

1、消息订阅、发布模式

2、相关命令

PUBLISH
SUBSCRIBE
PSUBSCRIBE
UNSUBSCRIBE

大数据库系统

3.6 Redis持久化

3.5 Redis的持久化

- 主要内容

  3.5.1 RDB持久化

  3.5.2 AOF持久化

3.5 Redis的持久化

◆ 为什么需要持久化

- Redis作为数据库存储数据，希望长久保存数据，就需要持久化[4]

什么是持久化?

采用某种方式将数据从内存保存到硬盘中，使得服务器重启之后，可以根据硬盘中保存的数据进行恢复，这个过程就是持久化

3.5 Redis的持久化

- Redis支持两种持久化方式
  - RDB持久化

    根据指定的规则“定时”将内存中的数据（快照）保存到硬盘中
  - AOF持久化

    将每次执行的命令及时形成日志保存到硬盘中

这两种持久化方式可以单独使用，也可以将二者结合起来使用[5]

3.5.1 RDB持久化

◆ RDB持久化

- 每隔N分钟或者N次写操作后，Redis会自动将内存中的所有数据生成一份副本，经过压缩成二进制文件存储在硬盘目录
- 这个过程被称为“快照”（红色部分可以自定义）
- 快照文件可以还原当时的数据库状态
- 由于快照文件保存在硬盘上，就算服务器停止服务，也可以利用RDB文件来还原数据库状态。

3.5.1 RDB持久化

- 快照的实现过程

1、Redis调用执行fork函数复制一份当前进程（父进程）的副本（子进程）[6]

2、父进程继续处理来自客户端的命令请求，而子进程则将内存中的数据写到硬盘上的一个临时RDB文件中

3、当子进程把所有数据写完后，也就表示快照生成完毕，此时旧的RDB文件将会被这个临时RDB文件替换，这个旧的RDB文件也会被删除

3.5.1 RDB持久化

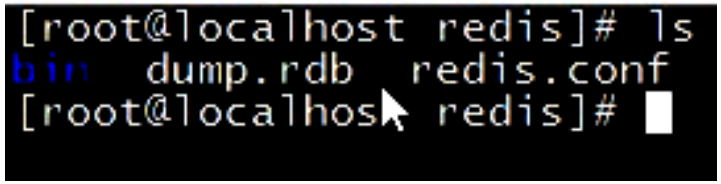
- 快照的实现过程
  - 在执行fork函数的过程中，父、子进程共享同一内存数据
  - 当父进程要修改某个数据时，操作系统会将这个共享内存数据另外复制一份给子进程使用，以此来保证子进程的正确运行[7]

如果写操作比较多，内存使用量变大（复制多）

3.5.1 RDB持久化

◆ 配置RDB持久化

- 默认配置下，在Redis所在目录可以看到Redis的配置文件“redis.conf”和一个dump.rdb文件



- RDB的一些配置可以在redis.conf中修改
- 而dump.rdb则是目前redis根据redis.conf的配置生成的一个rdb文件

进入redis.conf文件学习下几个RDB的配置项

vim /usr/local/redis/redis.conf

3.5.1 RDB持久化

◆ save

```
save 900 1
save 300 10
save 60 10000
```

- save规定了RDB重写时机和条件，它们之间是“或”的关系，每次只有一个快照条件会被执行（从下往上看易于理解）
  - save 900 1：上次RDB结束后900秒内有1个或1个以上的键被修改就会进行快照
  - save 300 10：上次RDB结束后300秒内有10个或10个以上的键被修改就会进行快照
  - save 60 10000：上次RDB结束后60秒内有10000个或10000个以上的键被修改就会进行快照
- 如果将所有save命令屏蔽，则关闭RDB功能

3.5.1 RDB持久化

- stop-writes-on-bgsave-error

当RDB过程中或执行BGSAVE命令出现错误时，Redis是否终止执行写命令

```
stop-writes-on-bgsave-error yes
```

- 参数的值默认被设置为yes，当导出RDB过程中出错时，Redis将拒绝新的写入操作
- 当设置为No时，即使RDB过程中出错，服务器也会继续执行写命令

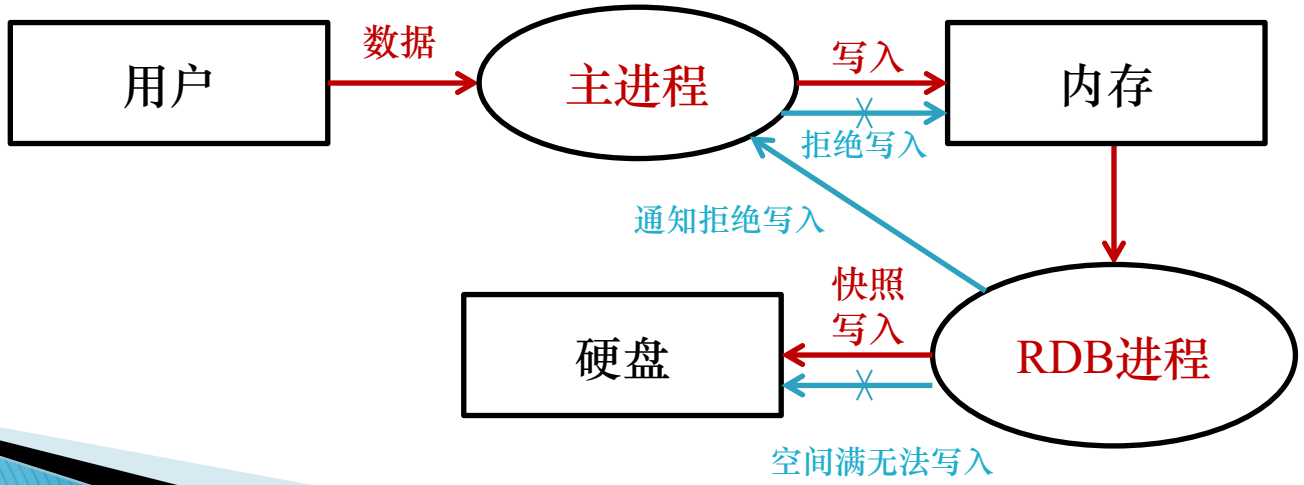
3.5.1 RDB持久化

怎么理解stop-writes-on-bgsave-error的作用呢?

在向硬盘写入RDB时，redis会专门创建一个进程负责写入RDB，另一个进程负责接收用户命令完成写操作。

假定写RDB需要20s的时间，但是，就在这20s内，创建快照出错了，比如磁盘满了，此时无法向硬盘写入快照，

如果该选项为YSE时则会拒绝新的写入，避免大量数据丢失，也可以避免数据两边不一致[8]



3.5.1 RDB持久化

- rdbcompression

  是否开启RDB压缩文件，

  默认为yes表示开启，不开启则设置为no

```
rdbcompression yes
```

  - 由于计算机内存空间普遍较大（32G，64G……）而且将越来越大，因此可以选择将RDB文件进行压缩，以减少磁盘空间的占用

3.5.1 RDB持久化

- rdbchecksum

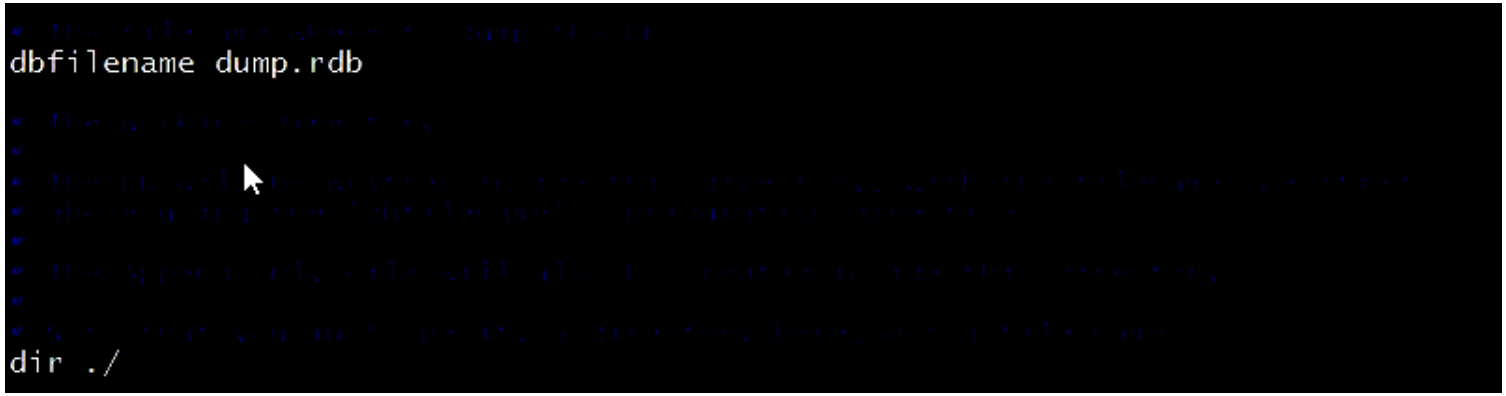
  是否开启RDB文件的校验，在服务器进行RDB文件的写入与读取时会用到它，默认设置为yes

```
rdbchecksum yes
```

  - 在服务器重启后，在从磁盘将RDB文件导入内存前检查RDB文件是否完整
  - 如果将它设置为no，则在服务器对RDB文件进行写入与读取时，可以提升性能，但是无法确定RDB文件是否已经被损坏

- dbfilename

  用于设置RDB文件名，默认为dump.rdb



- dir

  指定RDB文件所在目录，默认为Redis的根目录：./

  可以通过命令来修改它，命令格式如下：

  CONFIG SET dbfilename RDB文件名

  CONFIG SET dir RDB文件路径

  注意：

  1、高版本的aof文件与rdb文件放在同一个目录，均由“dir”配置项配置目录

  2、如果更改了AOF与RDB存放目录，不要忘了需要更改目录的权限，可使用chmod 777 更改目录权限，例如：sudo chmod 777 /usr/local/redis/rdbaof/

3.5.1 RDB持久化

- 实例

通过一个实例了解RDB持久化恢复数据库里的数据

1、关闭redis进程：pkill -9 redis，修改redis.conf文件，将rdb文件路径设置为“/var/rdb”，并将创建RDB条件改为60秒内3000次修改生成RDB，其他配置项均为默认

```
save 900 1
save 300 10
save 60 3000

dir /var/rdb
```

创建“/var/rdb”目录

```
[root@localhost redis]# ls /var/
cache  db  empty  games  lib  local  lock
[root@localhost redis]# mkdir rdb
```

为了防止混淆，删掉原先默认的RDB文件

```
[root@localhost redis]# rm dump.rdb
rm: remove regular file `dump.rdb'? y
[root@localhost redis]# ls
bin  redis.conf
[root@localhost redis]#
```

2、进入/usr/local/redis目录，开启redis服务端

./src/redis-server redis.conf

客户端连接至服务器，并写入键值site

```
redis 127.0.0.1:6379> set site www.zixue.it
OK
redis 127.0.0.1:6379>
```

问：此时服务器会有RDB导出吗?

```
[root@localhost redis]# ls /var/rdb/
[root@localhost redis]#
```

没有，为什么?

3、为了让服务器端产生RDB文件，可以通过redis-benchmark程序，短时间内进行10000次修改，达到RDB生成条件

进入/usr/local/redis目录，执行redis-benchmark程序

./src/redis-benchmark –n 10000

等待一段时间后，运行结束，每秒1799次请求，肯定可以达到60秒内3000次修改的要求

```
99.73% <= 105 milliseconds
99.75% <= 106 milliseconds
99.76% <= 120 milliseconds
100.00% <= 120 milliseconds
1799.53 requests per second


[root@localhost redis]#
```

进入“/var/rdb/”目录下，观察是否产生了RDB文件

```
[root@localhost redis]# ls /var/rdb/
dump.rdb
[root@localhost redis]#
```

服务器已经产生了RDB文件

为了做对比测试，我们再回到客户端，往服务器再写入一个新的键值address

```
redis 127.0.0.1:6379> set address bj
OK
redis 127.0.0.1:6379>
```

4、使用pkill命令强行关闭服务器的redis进程，模拟断电

```
[root@localhost redis]# pkill -9 redis
[root@localhost redis]#
```

5、再次重启redis服务器

6、启动客户端，连上服务器，分别查询两个键值是否存在

```
redis 127.0.0.1:6379> get site
"www.zixue.it"
```

Site键值内容存在，那address键值是否存在?

```
redis 127.0.0.1:6379> get address
(nil)
redis 127.0.0.1:6379>
```

address键值并不存在，为什么?

存address之前刚进行RDB，因此刚放address时没达到生成RDB条件

3.5.1 RDB持久化

- RDB持久化的优点

1、RDB文件是一个经过压缩的二进制文件（需要开启rdbcompression），文件紧凑，体积较小，非常适用于进行数据库数据备份

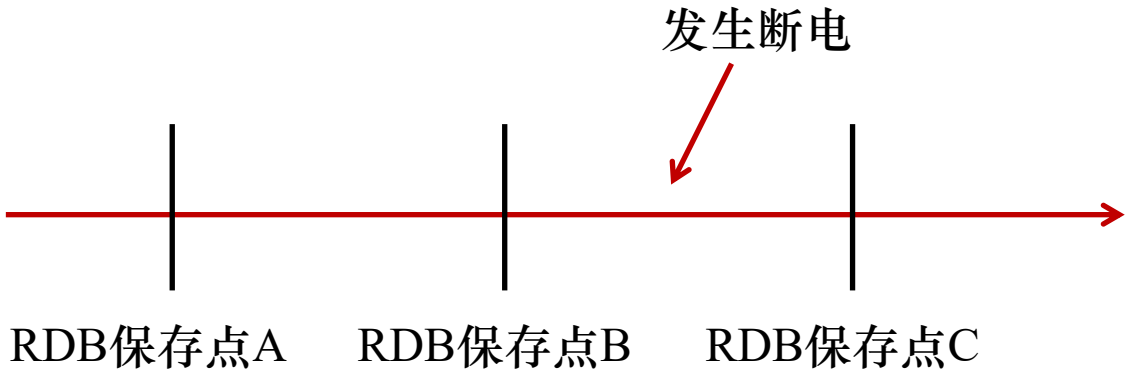
2、 RDB持久化适用于灾难恢复，而且恢复数据时的速度快（因为是内存快照，将快照文件直接读到内存里）

3、并行性好，单独的进程保存RDB快照，保存RDB的过程中，其他进程继续处理其他相关的操作

3.5.1 RDB持久化

◆ RDB的缺陷

1、从上一个实例可以看出，如果在两个RDB保存点之间发生了断电，那么将会损失两个保存点之间时间段写入的数据



- 在大数据时代，几分钟的数据丢失在商业中很有可能是巨大的经济损失
- 适用于对数据完整性和一致性不太高的应用

3.5.1 RDB持久化

- RDB持久化的缺陷

2、当数据量非常庞大时，在生成和保存RDB文件的时候，操作比较耗时，将会占用太多CPU时间，从而影响服务器的性能

3、RDB文件存在兼容性问题，老版本的Redis不支持新版本的RDB文件

3.5.1 RDB持久化

- RDB适用场景
  - ✓ 如果可以接受十几分钟或更多的数据丢失，选择RDB对Redis的性能更加有利
  - ✓ 由于RDB文件小(如果开启了压缩功能)、恢复快，因此灾难恢复（有快速恢复要求的）常用RDB文件；

- 针对RDB方式的劣势

总结

- Redis持久化
  - 什么是持久化
  - Redis持久化的两种方式
  - RDB持久化
    - ✓ RDB是什么
    - ✓ 快照的过程
    - ✓ RDB配置项
    - ✓ 通过实例理解RDB过程
    - ✓ RDB的优缺点
    - ✓ RDB适用场景