链滴

后端架构学习（2）- redis

作者：ccran

原文链接：https://ld246.com/article/1590212036319

来源网站：链滴

后端架构学习（2）- redis

之前主要记录了docker部署一个最简单的后端，本次加入redis。

架构会随着场景的变更而变换；因此，这里假设一个很简单的场景：

- 实现商品秒杀
- 商品信息存储在数据库中
- 功能为每天拿前一天销量TOP100商品作为秒杀商品

V2架构

1. 技术栈

1.1 简介

- SpringBoot
- MySQL
- docker
- redis

在V1架构的基础上加入了redis，redis是一个基于内存的高性能key*-*value数据库。

1.2 使用

在使用上：**类比redis为Java的Map**,redis有以下几种数据类型作为value

- String: 字符串 ~ Map<String,String>
- Hash: 散列 ~ Map<String,Object>
- List: 列表 ~ Map<String,ArrayList>
- Set: 集合 ~ Map<String,HashSet>
- Sorted Set: 有序集合 ~ Map<String,TreeSet>

String

```
> set name ccran
OK
> get name
ccran
> set age 18 EX 5
OK
> get age
18
> keys *
name
> get age
null
```

set命令格式：**set key value**

EX设置过期时间，5S后该key-value失效

keys * 可以获取所有的key值

Hash

```
> hmset person name "ccran" age 18
OK
> hgetall person
name
ccran
age
18
```

hmset格式： **HMSET key field value [field value ...]**

hgetall获取key对应value对象的所有字段和值

List

```
> lpush skill redis
1
> lpush skill mysql
2
> lrange skill 0 5
mysql
redis
```

```
> rpush skill java
3
> lrange skill 0 -1
mysql
redis
java
```

lpush列表头添加对象，rpush列表尾添加对象

lrange取列表范围元素，命令格式为 **lrange key start stop**，包含start以及stop

Set

```
> sadd num 1
1
> sadd num 2
1
> sadd num 2
0
> smembers num
1
2
```

集合里面没有重复元素

Sorted Set

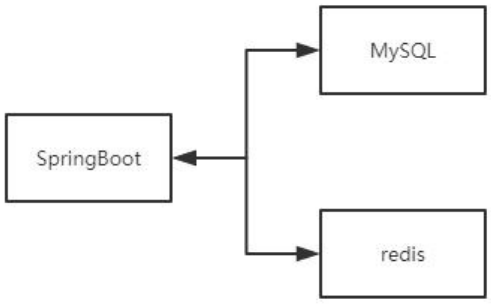
```
> zadd company 1 google
1
> zadd company 2 apple
1
> zadd company 2 apple
0
> zadd company 1 apple
0
> zrange company 0 -1 withscores
apple
1
google
1
```

zadd命令格式：**ZADD key score member [[score member] [score member] ...]**

根据score排序，value不能重复，但是score可以重复

最后贴出Redis命令参考： http://doc.redisfans.com/

2. 架构



在V1架构的基础上增加了SpringBoot与redis的交互

2.1 秒杀实现

V1架构秒杀实现的流程：秒杀页面=>秒杀接口=>后端<=>MySQL

1. 用户进入秒杀页面
2. 前端调用秒杀接口
3. 后端通过ORM框架生成SQL语句交给MySQL并等待数据返回
4. MySQL解析语句，定位数据位置，最差情况需要读磁盘
5. 数据返回给后端，返回给接口，前端展示页面

V2架构秒杀实现的流程：在第5步，**数据返回给后端以后，写入redis，并设置过期时间为1天，以后次查询如果redis有数据则直接拿redis数据。**

V1架构与V2架构比较：

1. V1架构每次需要去数据库进行查询；虽然MySQL可能会缓存查询，预读数据页，但最差还是可能在磁盘IO，预估20ms左右
2. V2架构只有第一次需要去数据库进行查询；后续所有访问全部到redis，由于是在内存中，预估2m左右

V1架构如果能支持2000并发的话，V2架构则能支持20000并发了。
mile

2.2 redis vs map

为什么不用java map呢？（个人理解）

- map会占据堆内存；增大垃圾回收压力，增大整个后端压力
- 如果开多个后端实例，会造成数据冗余；
- 不便于后期分布式拓展；
- redis功能更丰富；如过期，否则我们只能手动管理
- 数据与业务解耦；可能有其他后端业务需要，类比于使用MySQL存储数据，而不是后端自身组织数据
- ...

3. 部署

首先拉取镜像

```
docker pull redis:5.0
```

然后运行镜像，生成容器

```
docker run -p 6379:6379  -d --name redis redis:5.0
```

4. 总结

1. 可以缓存访问频繁的数据，提高并发量。

2. 解决问题需要找到问题的关键，并且使用合适的组件来充分发挥其优势。

3. 多学习，很多问题可以采用相似的方案解决；比如很多地方都用到了缓存，浏览器、数据库、DNS CPU等