《Java性能优化与面试21讲》

李国

— 拉勾教育出品 —

08 |案例分析：Redis 如何助力秒杀业务

前言

什么叫分布式缓存呢？

它是一种**集中管理的思想**

前言

Redis 支持非常丰富的数据类型

包括**字符串（string）**、**列表（list）**、**集合（set）**、**有序集合（zset）**、**哈希表（hash）**等常用的数据结构

也支持一些其他的比如**位图（bitmap）**一类的数据结构

前言

	Redis	MC
是否多线程	否	是
数据类型	数据类型丰富	字符串类型
数据保存	数据可持久化至硬盘	断电后数据会丢失
性能表现	存储小数据时性能高	存储大数据（如超 100kb）性能高
数据划分	使用基于哈希槽（slot）的划分方式	客户端实现的一致性哈希（Consistency Hashing）

SpringBoot 如何使用 Redis

使用 SpringBoot 可以很容易地对 Redis 进行操作

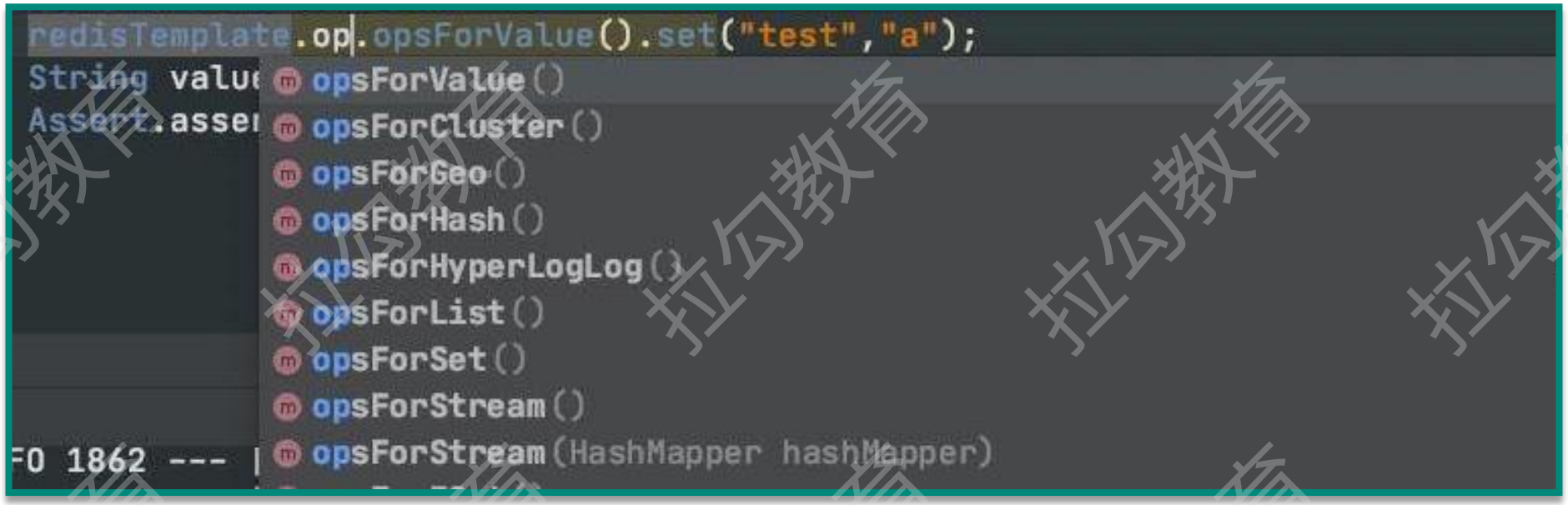
Java 的 Redis的客户端

常用的有三个：**jedis**、**redisson** 和 **lettuce**，Spring 默认使用的是 lettuce

SpringBoot 如何使用 Redis


```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

SpringBoot 如何使用 Redis



SpringBoot 如何使用 Redis


```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-cache</artifactId>
</dependency>
```

SpringBoot 如何使用 Redis

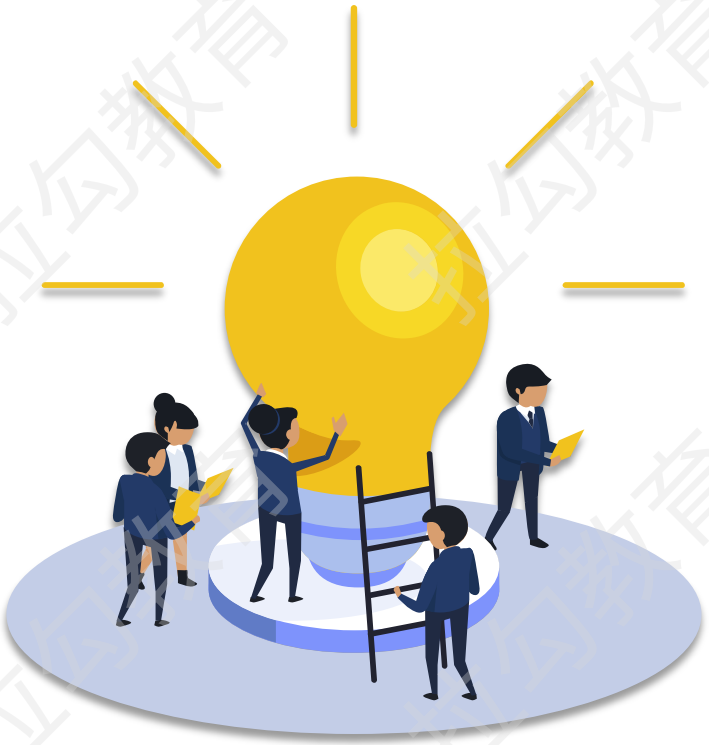
使用 spring-cache 有三个步骤：

- 在启动类上加入 @EnableCaching 注解
- 使用 CacheManager 初始化要使用的缓存框架，使用 @CacheConfig 注解注入要使用的资源
- 使用 @Cacheable 等注解对资源进行缓存

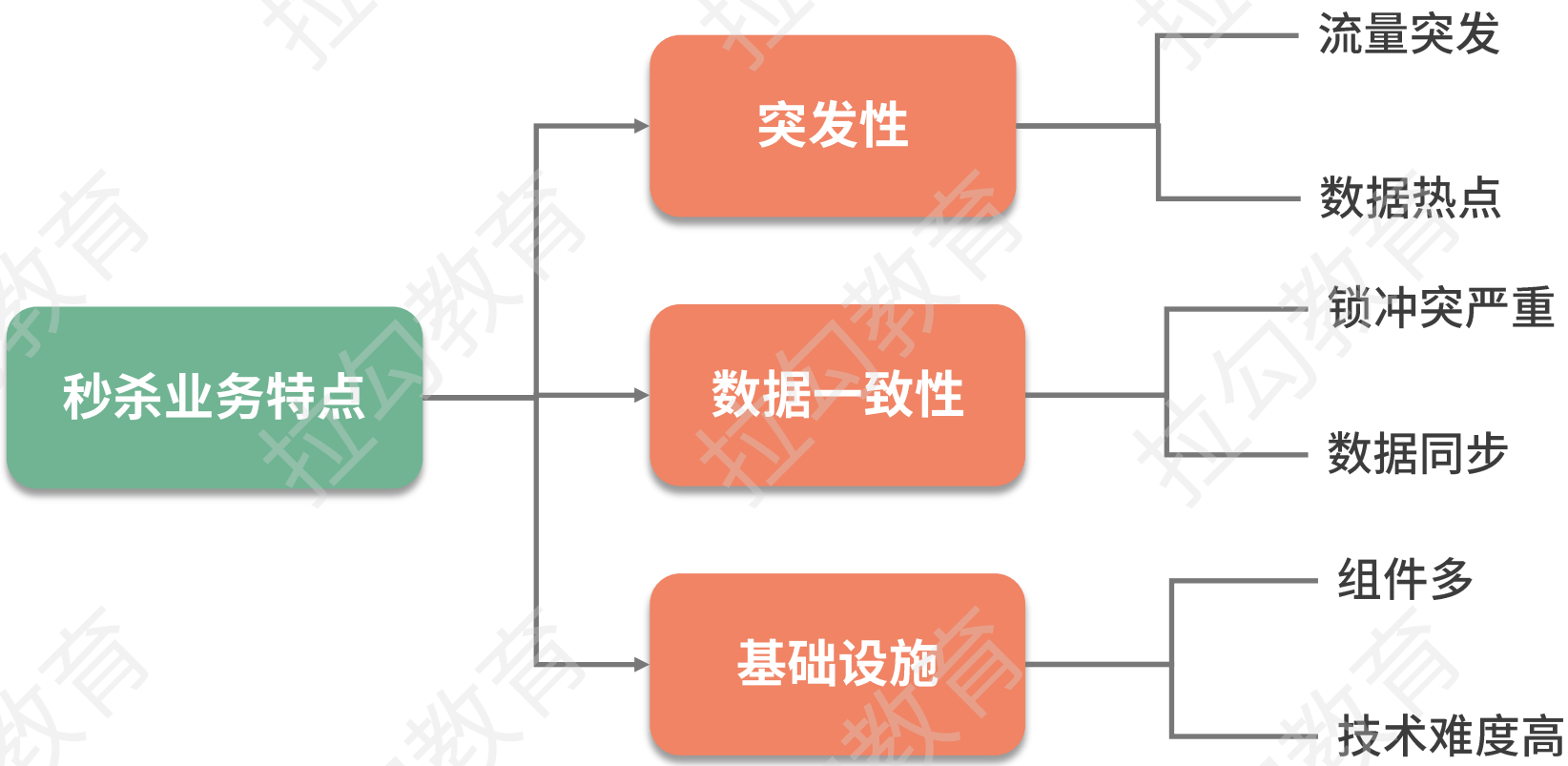
SpringBoot 如何使用 Redis

针对缓存操作的注解，有三个：

- @Cacheable 表示如果缓存系统里没有这个数值，就将方法的返回值缓存起来
- @CachePut 表示每次执行该方法，都把返回值缓存起来
- @CacheEvict 表示执行方法的时候，清除某些缓存值



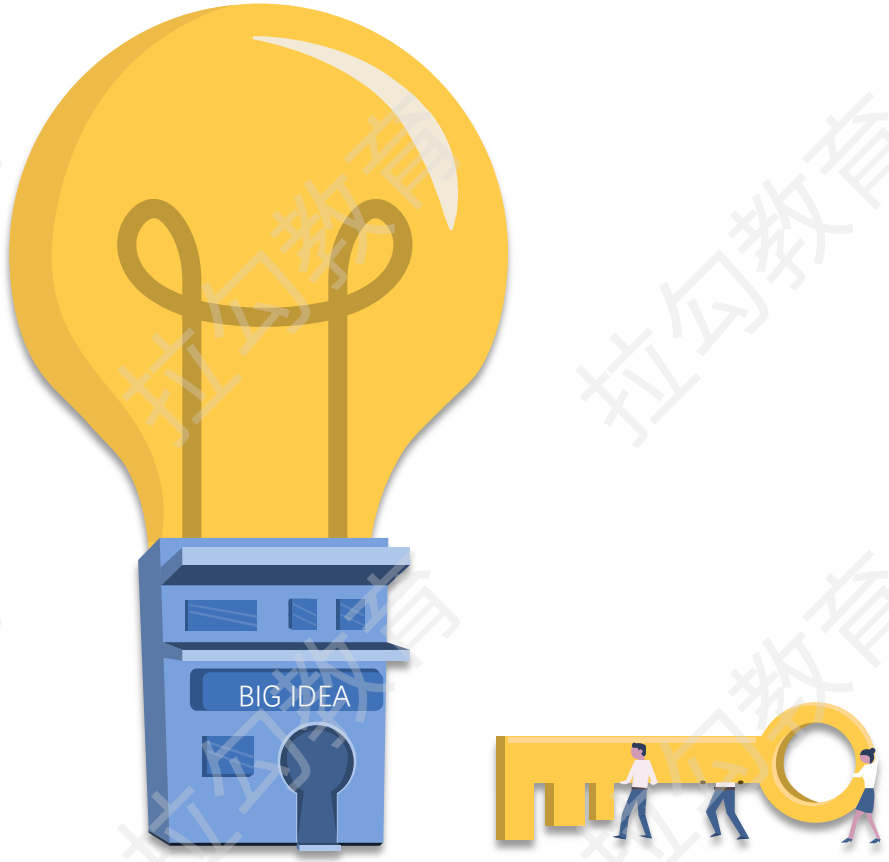
秒杀业务介绍



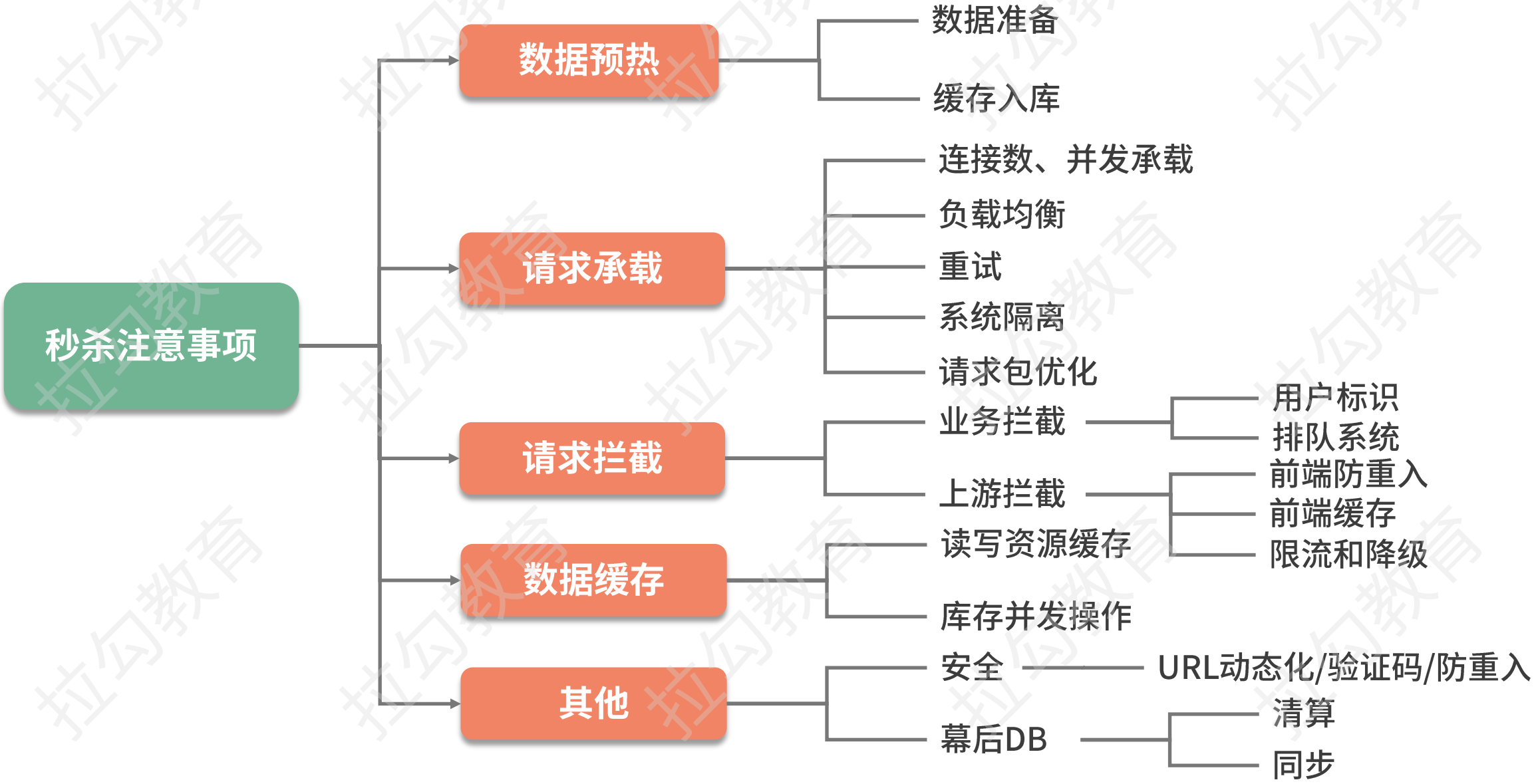
秒杀业务介绍

处理秒杀业务有三个绝招：

1. 选择速度最快的内存作为数据写入
2. 使用异步处理代替同步请求
3. 使用分布式横向扩展



Lua 脚本完成秒杀



Lua 脚本完成秒杀


```
seckill:goods:${goodsId}{
    total: 100,
    start: 0,
    alloc:0
}
```

Lua 脚本完成秒杀


```
static final String goodsId = "seckill:goods:%s";

String getKey(String id) {
    return String.format(goodsId, id);
}
public void prepare(String id, int total) {
    String key = getKey(id);
    Map<String, Integer> goods = new HashMap<>();
    goods.put("total", total);
    goods.put("start", 0);
    goods.put("alloc", 0);
    redisTemplate.opsForHash().putAll(key, goods);
}
```

Lua 脚本完成秒杀


```
local falseRet = "0"
local n = tonumber(ARGV[1])
local key = KEYS[1]
local goodsInfo =
redis.call("HMGET",key,"total","alloc")
local total = tonumber(goodsInfo[1])
local alloc = tonumber(goodsInfo[2])
if not total then
   return falseRet
end
if total >= alloc + n  then
   local ret = redis.call("HINCRBY",key,"alloc",n)
   return tostring(ret)
end
return falseRet
```

Lua 脚本完成秒杀

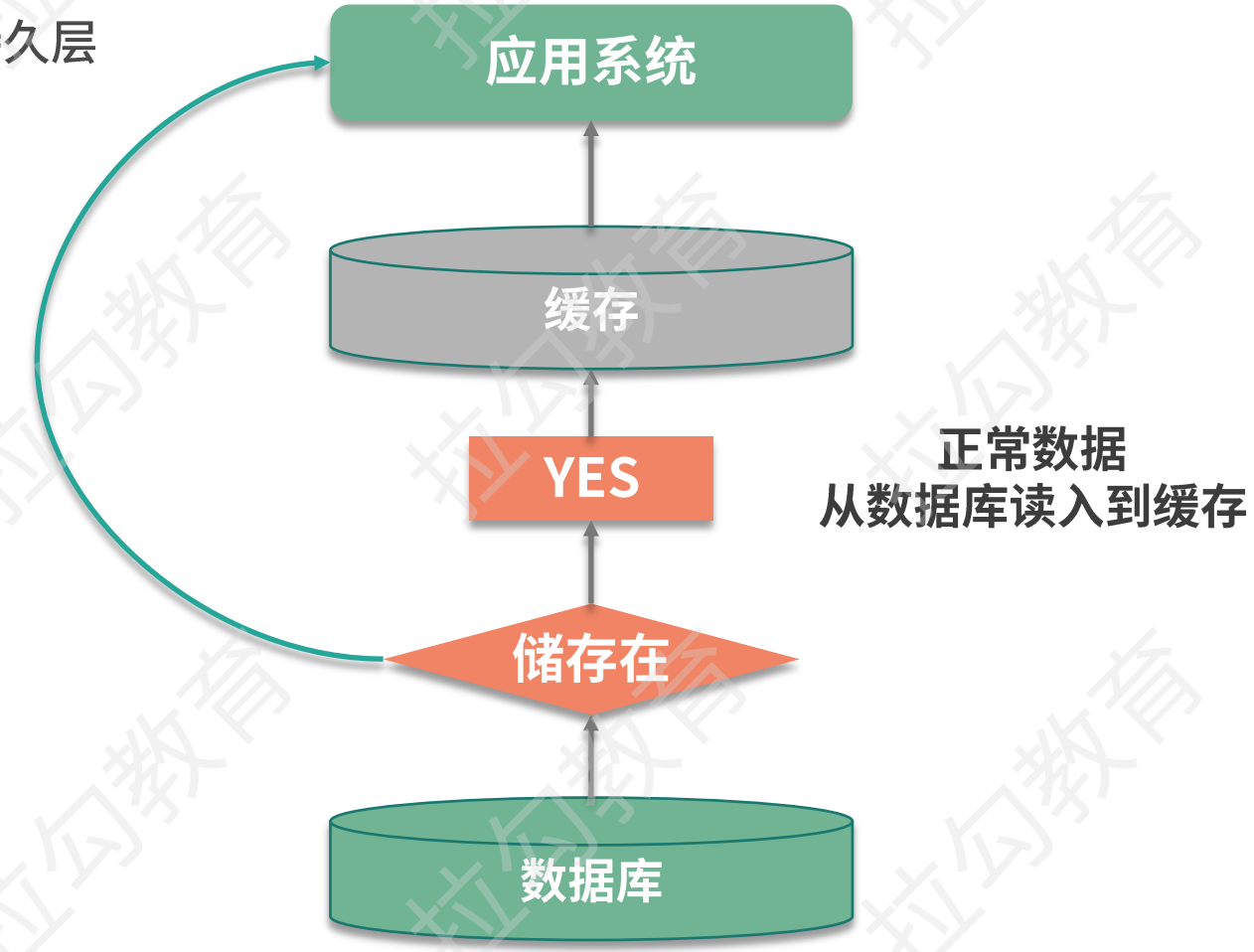

```
public int secKill(String id, int number) {
    String key = getKey(id);
    Object alloc =  redisTemplate.execute(script, Arrays.asList(key), String.valueOf(number));
    return Integer.valueOf(alloc.toString());
}
```

Lua 脚本完成秒杀

缓存穿透、击穿和雪崩

缓存穿透

如果命中率很低，那么压力就会集中在数据库持久层



缓存穿透、击穿和雪崩

缓存击穿

指的也是用户请求落在数据库上的情况，大多数情况，是由于缓存时间批量过期引起的

一般会对缓存中的数据，设置一个过期时间

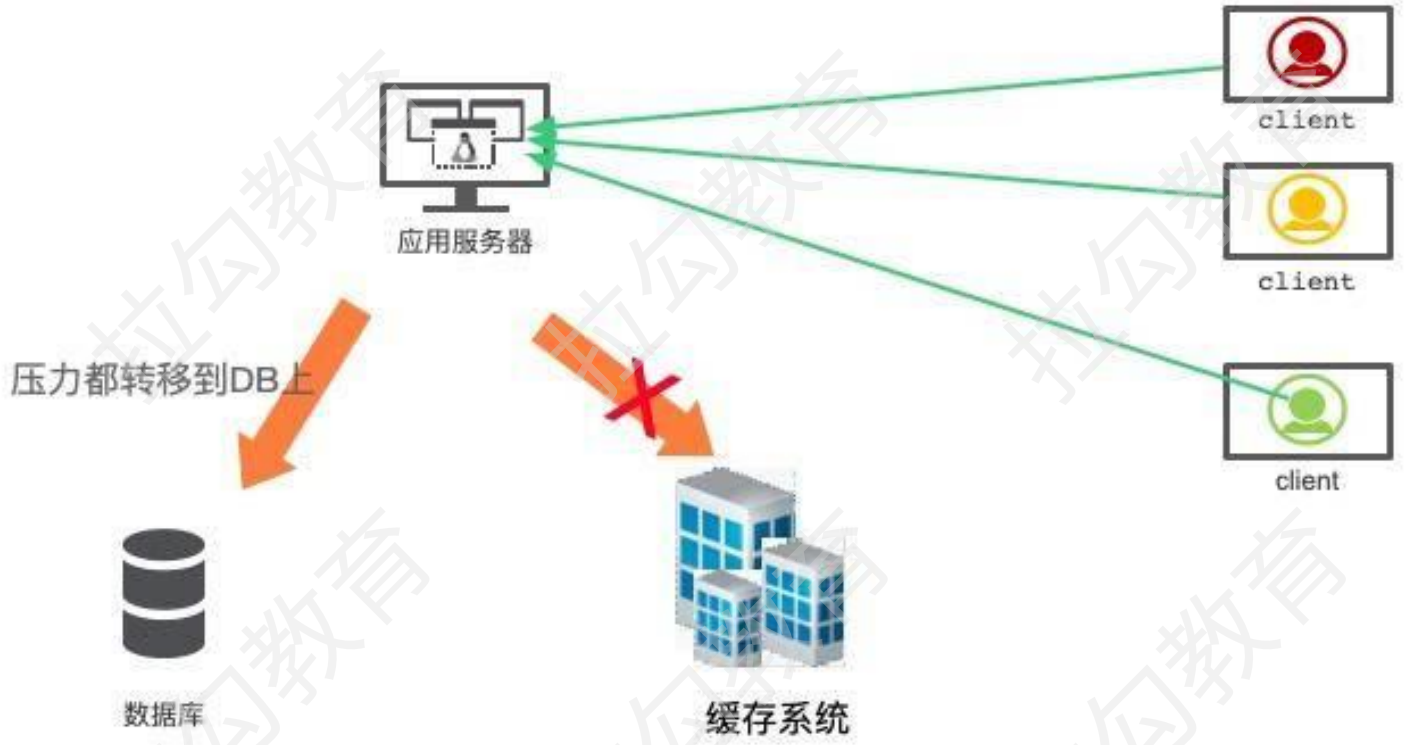
如果在某个时刻从数据库获取了大量数据，并设置了同样的过期时间

它们将会在同一时刻失效，造成和缓存的击穿

缓存穿透、击穿和雪崩

缓存雪崩

缓存是用来对系统加速的，后端的数据库只是数据的备份，而不是作为高可用的备选方案



缓存一致性

常用的操作有四个：写入、更新、读取、删除

- **写入**

  缓存和数据库是两个不同的组件，只要涉及到双写，就存在只有一个写成功的可能性，造成数据不一致

- **更新**

  更新的情况类似，需要更新两个不同的组件

- **读取**

  读取要保证从缓存中读到的信息是最新的，是和数据库中的是一致的

- **删除**

  当删除数据库记录的时候，如何把缓存中的数据也删掉？

缓存一致性

推荐使用触发式的缓存一致性方式，使用懒加载的方式，可以让缓存的同步变得非常简单：

- 当读取缓存的时候，如果缓存里没有相关数据，则执行相关的业务逻辑，构造缓存数据存入到缓存系统
- 当与缓存项相关的资源有变动，则先删除相应的缓存项，然后再对资源进行更新

  这时即使是资源更新失败，也是没有问题的

缓存一致性

小结

对于分布式缓存来说，Redis 是现在使用最广泛的

先简单介绍它和 Memcached 的一些区别

介绍了 SpringBoot 项目中 Redis 的使用方式

然后以秒杀场景为主，学习了库存扣减这一个核心功能的 Lua 代码

这段代码主要是把条件判断和扣减命令做成了原子性操作

小结

Redis 的 API 使用非常简单，速度也很快，但同时它也引入了很多问题

如果不能解决这些异常场景，那么 Redis 的价值就大打折扣

所以主要谈到了缓存的穿透、击穿以及雪崩的场景，并着重介绍了一下缓存的一致性和解决的思路

Next：第08讲《案例分析：池化对象的应用场景》

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容

L / A / G / O / U