讲师介绍

Hash　　QQ：805921455

从事Java软件研发十年。

前新浪支付核心成员、

咪咕视讯(中国移动)项目经理、

对分布式架构、高性能编程有深入的研究。

明天，你一定会感谢今天奋力拼搏的你

生产环境缓存爬坑记
-缓存失效解决方案

分布式高并发—缓存技术

目录

课程安排

01

Redis的持久化机制

RDB、AOF持久化机制、为什么持久化和主从都可能出现数据丢失

02

淘汰策略导致数据失效

Redis中内存管理、Key的淘汰及过期处理

03

缓存击穿、缓存雪崩的解决方案

缓存击穿原因、雪崩危害、解决方案

04

总结

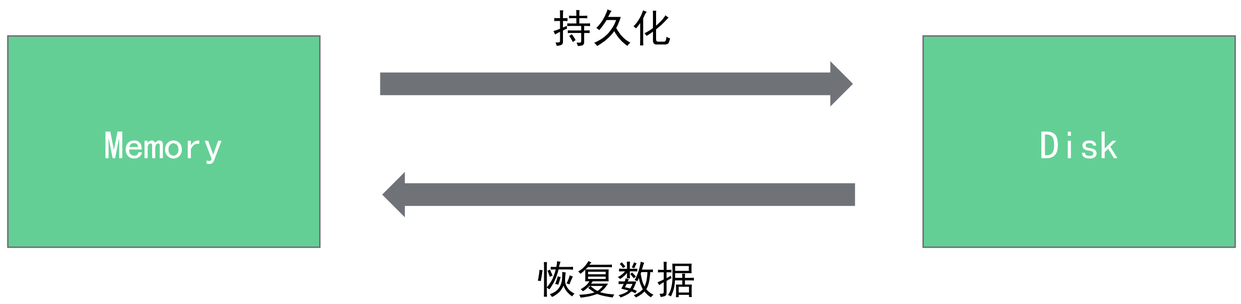
会总结、会学习

目录

Redis的持久化机制

持久化介绍

Redis的数据都存放在内存中，如果没有配置持久化，redis重启后数据就全丢失了，于是需要开启redis的持久化功能，将数据保存到磁盘上，当redis重启后，可以从磁盘中恢复数据。



持久化的方式

RDB 持久化

RDB 持久化方式能够在指定的时间间隔对你的数据进行快照存储

AOF（append only file）持久化

AOF 持久化方式记录每次对服务器写的操作，当服务器重启的时候会重新执行这些命令来恢复原始的数据

RDB 方式

- 客户端直接通过命令BGSAVE或者SAVE来创建一个内存快照
  - BGSAVE 调用fork来创建一个子进程，子进程负责将快照写入磁盘，而父进程仍然继续处理命令。
  - SAVE 执行SAVE命令过程中，不再响应其他命令。

- 在redis.conf中调整save配置选项，当在规定的时间内，Redis发生了写操作的个数满足条件会触发发生BGSAVE命令

```
# 900秒之内至少一次写操作
save 900 1
# 300秒之内至少发生10次写操作
save 300 10
# 60秒之内发生至少10000次
save 60 10000
```

RDB 优点和缺点

优点	缺点
对性能影响最小	同步时丢失数据
RDB文件进行数据恢复比使用AOF要快很多	如果数据集非常大且CPU不够强（比如单核CPU），Redis在fork子进程时可能会消耗相对较长的时间，影响Redis对外提供服务的能力。

AOF 持久化方式

- **记录每次服务受到的写**

  BGREWRITEAOF命令可以触发日志重写或自动重写，废除对同一个Key历史的无用命令，重建当前数据集所需的最短命令序列。

  意外中断，如果最后的命令只写了一部分，恢复时则会跳过它，执行后面完整的命令。

- 开启AOF持久化

```
appendonly yes
```

- AOF策略调整

```
#每次有数据修改发生时都会写入AOF文件，非常安全非常慢
appendfsync always
```

```
#每秒钟同步一次，该策略为AOF的缺省策略，够快可能会丢失1秒的数据
appendfsync everysec
```

```
#不主动fsync,由操作系统决定，更快，更不安全的方法
appendfsync no
```

AOF优点和缺点

优点	缺点
最安全	文件体积大
容灾	性能消耗比RDB高
易读，可修改	数据恢复速度比RDB慢

Redis丢失数据的可能性

持久化丢失的可能

RDB方式

快照产生的策略，天生就不保证数据安全

AOF持久化策略

默认每秒同步一次磁盘，可能会有1秒的数据丢失

每次修改都同步，数据安全可保证，但Redis高性能的特性全无

主从复制丢失的可能

异步复制，存在一定的时间窗口数据丢失

网络、服务器问题，存在一定数据的丢失

总结：持久化和主从都可能出现数据丢失

目录

淘汰策略导致数据失效

内存分配

不同数据类型的大小限制

- Strings类型：一个String类型的value最大可以存储512M。
- Lists类型：list的元素个数最多为2^32-1个，也就是4294967295个。
- Sets类型：元素个数最多为2^32-1个，也就是4294967295个。
- Hashes类型：键值对个数最多为2^32-1个，也就是4294967295个

```
# 最大内存控制
maxmemory 最大内存阈值
maxmemory-policy 到达阈值的执行策略
```

内存压缩

```
#配置字段最多512个
hash-max-zipmap-entries 512
#配置value最大为64字节
hash-max-zipmap-value 64
#配置元素个数最多512个
list-max-ziplist-entries 512
#配置value最大为64字节
list-max-ziplist-value 64
```

```
#配置元素个数最多512个
set-max-intset-entries 512
#配置元素个数最多128个
 zset-max-ziplist-entries 128
#配置value最大为64字节
zset-max-ziplist-value 64
```

大小超出压缩范围，溢出后Redis将自动将其转换为正常大小

过期数据的处理策略

主动处理（ redis 主动触发检测key是否过期）每秒执行10次。过程如下：

1. 从具有相关过期的密钥集中测试20个随机密钥
2. 删除找到的所有密钥已过期
3. 如果超过25%的密钥已过期，请从步骤1重新开始

被动处理：

1. 每次访问key的时候，发现超时后被动过期，清理掉

数据恢复阶段过期数据的处理策略

- RDB方式

过期的key不会被持久化到文件中。

载入时过期的key，会通过redis的主动和被动方式清理掉。

- AOF方式

当 redis 使用 AOF 方式持久化时，每次遇到过期的 key redis 会追加一条 DEL 命令 到 AOF 文件，

也就是说只要我们顺序载入执行 AOF 命令文件就会删除过期的键。

注意： 过期数据的计算和计算机本身的时间是有直接联系的！

LRU算法

LRU（Least recently used，最近最少使用）：根据数据的历史访问记录来进行淘汰数据

- 核心思想：如果数据最近被访问过，那么将来被访问的几率也更高。
- 注意：Redis的LRU算法并非完整的实现，完整的LRU实现是因为这需要太多的内存。
- 方法：通过对少量keys进行取样(50%)，然后回收其中一个最好的key。
- 配置方式： maxmemory-samples 5

LFU算法

LFU（Least Frequently Used）根据数据的历史访问频率来淘汰数据

- 核心思想：如果数据过去被访问多次，那么将来被访问的频率也更高。
- Redis实现的是近似的实现，每次对key进行访问时，用基于概率的对数计数器来记录访问次数，同时这个计数器会随着时间推移而减小。
- Morris counter算法依据：

  https://en.wikipedia.org/wiki/Approximate_counting_algorithm
- 启用LFU算法后，可以使用热点数据分析功能。（ redis-cli --hotkeys ）

Redis内存回收策略

配置文件中设置：maxmemory-policy noeviction

动态调整：config set maxmemory-policy noeviction

回收策略	说明
noeviction	客户端尝试执行会让更多内存被使用的命令直接报错
allkeys-lru	在所有key里执行LRU算法
volatile-lru	在所有已经过期的key里执行LRU算法
volatile-lfu	使用过期集在密钥中使用近似LFU进行驱逐
allkeys-lfu	使用近似LFU逐出任何键
allkeys-random	在所有key里随机回收
volatile-random	在已经过期的key里随机回收
volatile-ttl	回收已经过期的key，并且优先回收存活时间（TTL）较短的键

什么样的数据适合缓存

三个维度评判数据是否合适缓存

维度	适合缓存	不适合缓存
访问频率	访问频率高	访问频率低
读写比	读多写少	写多读少
一致性要求	一致性要求低	一致性要求高

目录

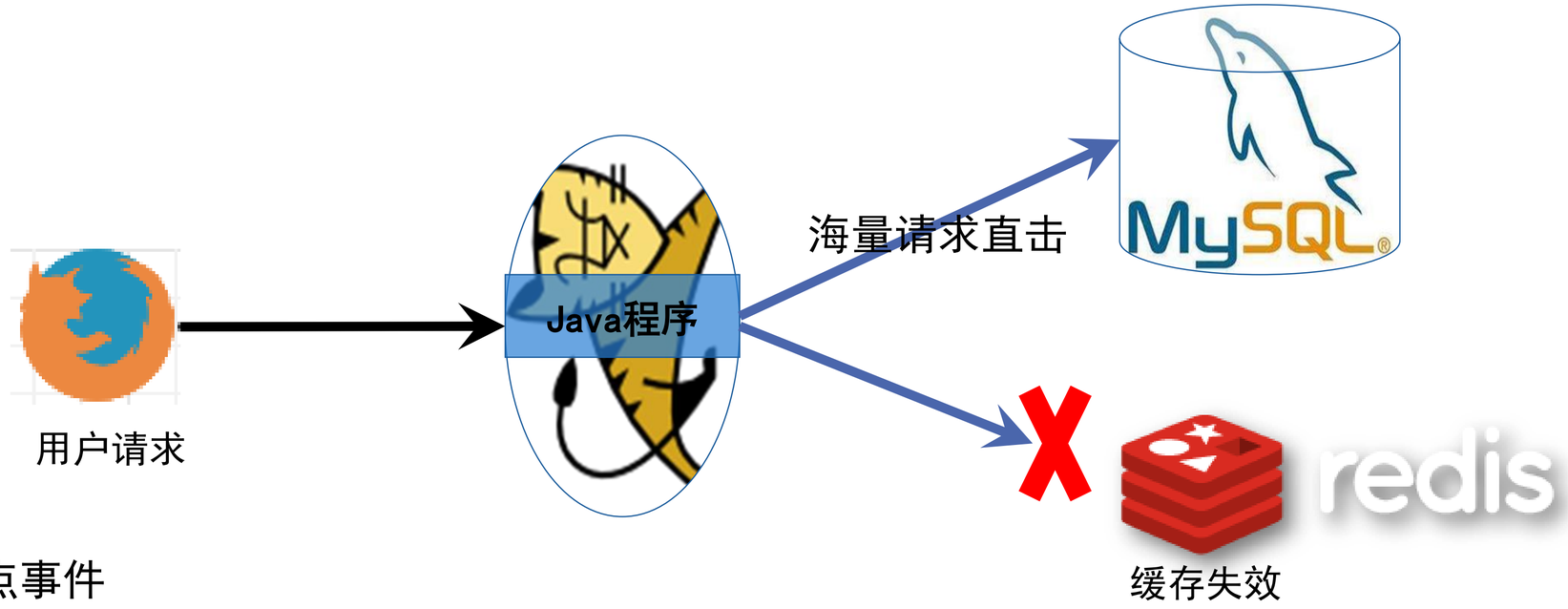
缓存穿透、缓存雪崩的解决方案

缓存雪崩原因-缓存穿透

缓存失效的两种情况：

1. 高峰期大面积缓存Key失效。（所有请求全部访问后端数据库）
2. 局部高峰期，热点缓存Key失效。（导致海量的请求直击数据库）

缓存数据有效期到来的那一瞬间

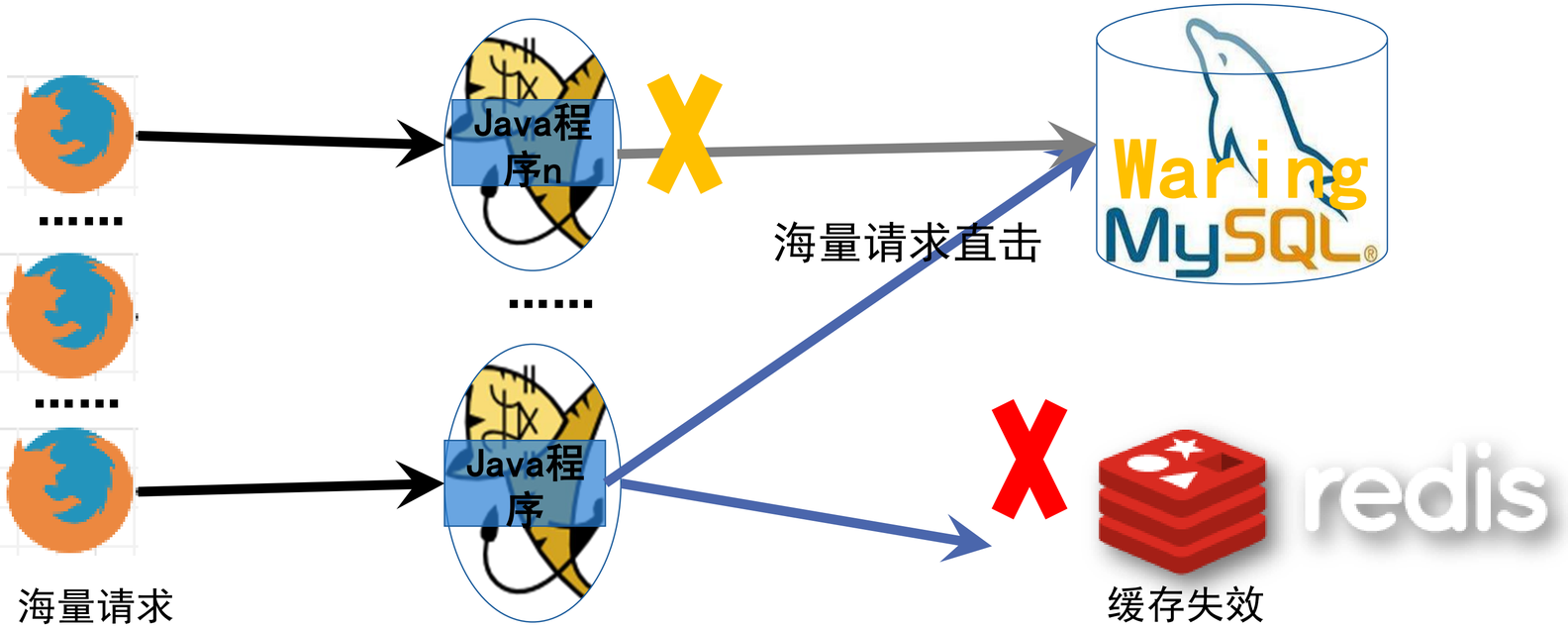


1. 突发重要热点事件
2. 春节发红包
3. 电商降价、抢购、促销活动
4. ……

缓存雪崩风险

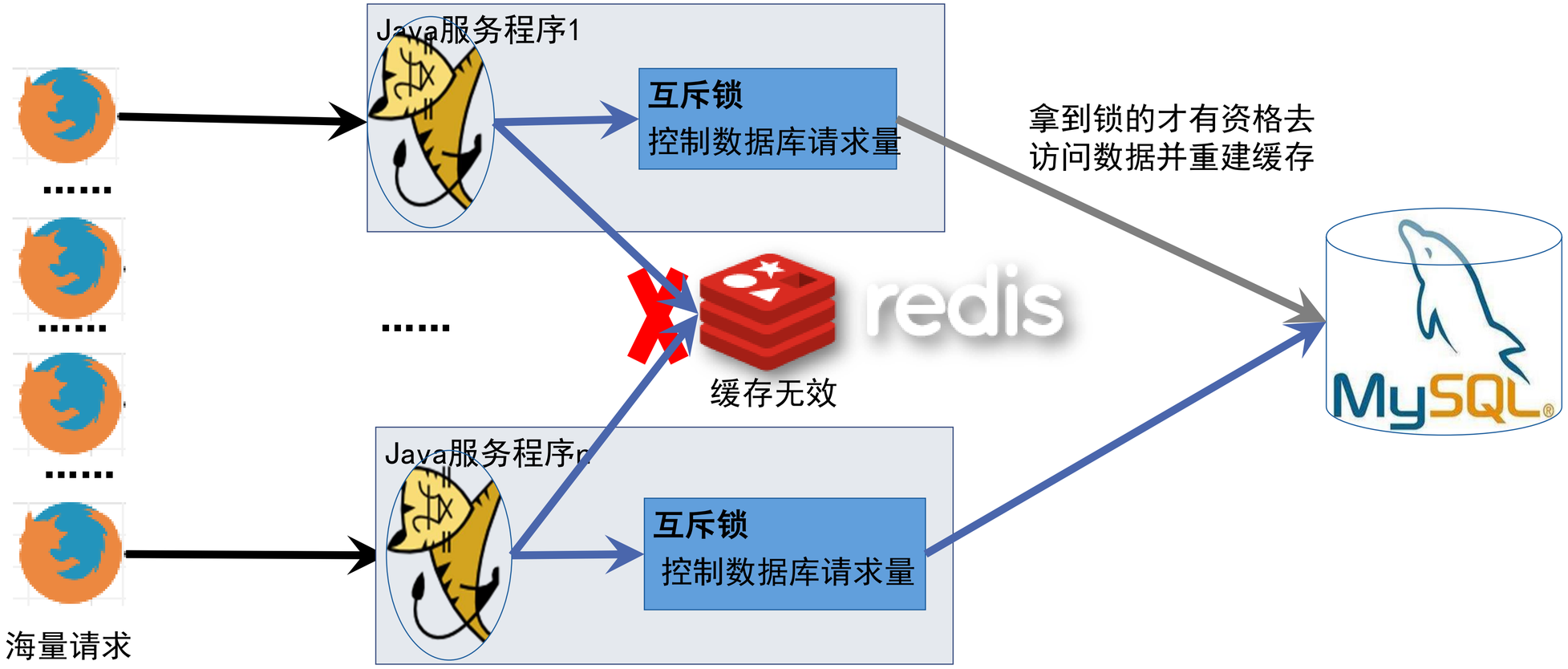
缓存雪崩：因为缓存服务挂掉或者热点缓存失效，从而导致海量请求去查询数据库，导致数据库连接不够用或者数据库处理不过来，从而导致整个系统不可用。

数据库服务器压力大，依赖数据库的其他系统也会面临崩溃风险。



雪崩的解决方案-互斥锁

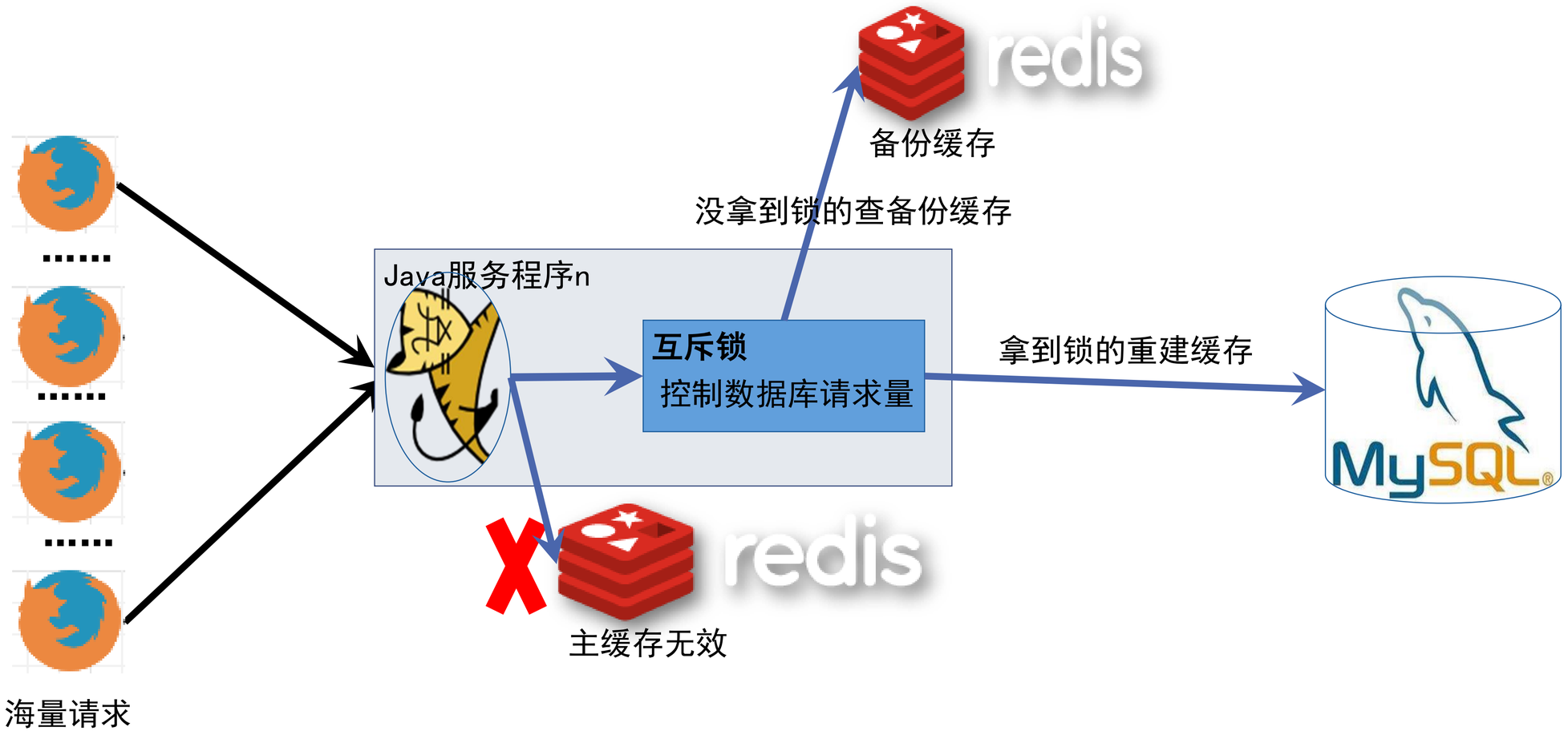
任由洪水猛兽般的流量轰击数据库？



雪崩的解决方案-缓存降级

拿到锁的线程负责更新缓存，其他请求读取备份缓存数据或者执行降级策略；

备份缓存通常是不设置过期时间的，异步更新的缓存。



雪崩的解决方案-缓存降级

- 优点：
  - 灵活多变，根据业务需要进行调整；
  - 使用方便

- 缺点：
  - 降级策略的选择对开发人员的要求高，需要能掌控业务；
  - 为了保证备份缓存的数据一致性，增加了维护的复杂度；

目录

总结

本节总结

Redis持久化机制各自的特点

Redis内存管理机制

Redis Key淘汰策略、过期机制

Redis缓存穿透的原因

Redis缓存雪崩的危害

缓存雪崩的解决方案

谢谢观看