# Spring Boot 进阶

缓存

# 课程内容

- Java Cache
  - 缓存相关概念
  - Spring Boot 自动配置缓存
- Cache 实战
  - 本地缓存 Guava/EHCache
  - 分布式缓存 Redis
  - 混合使用 Guava&Redis
- 高并发场景
  - 缓存穿透
  - 缓存数据一致性

# 缓存相关概念介绍

命中率

容量

淘汰策略

预热策略

# Spring Boot 自动配置缓存

- 引入缓存
  - 在pom.xml中引入cache依赖
- 开启缓存功能
  - 在Spring Boot主类中增加@EnableCaching注解
- 操作缓存
  - @Cacheable
  - @CachePut
  - @CacheEvict

# Spring Boot 自动配置缓存

- CacheManager
  - Generic
  - JCache (JSR-107)
  - **EhCache 2.x**
  - Hazelcast
  - Infinispan
  - **Redis**
  - **Guava**
  - Simple

- 强制指定
  - spring.cache.type=none
  - spring.cache.type=ehcache
  - spring.cache.type=guava
  - spring.cache.type=redis

# 本地缓存-EHCache

- 集成
  - 在pom.xml中引入EHCache依赖
- 配置
  - src/main/resources/ehcache.xml
  - <ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:noNamespaceSchemaLocation="http://www.ehcache.org/ehcache.xsd">

    <cache name="user" maxEntriesLocalHeap="10" timeToLiveSeconds="1"
    ></cache>
    </ehcache>

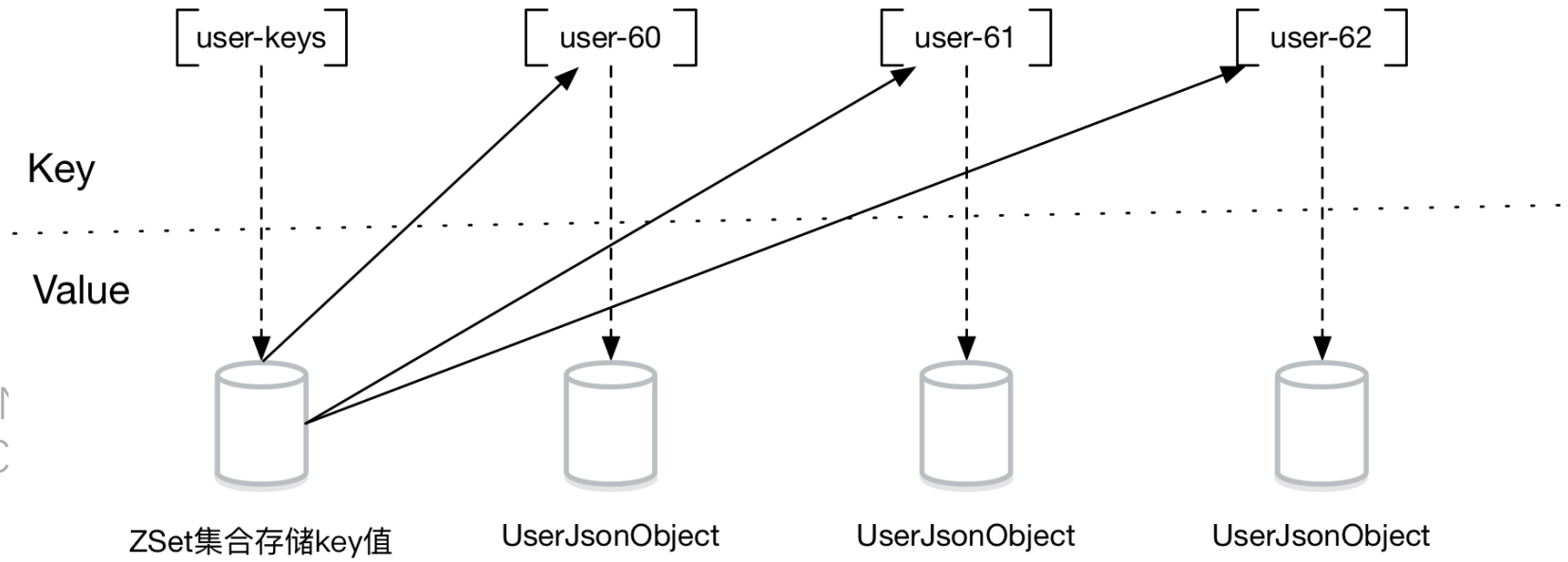
# 本地缓存-Guava Cache

- 集成
  - 在pom.xml中引入Guava Cache依赖
- 配置
  - spring.cache.guava.spec=maximumSize=500,expireAfterWrite=1s

# 分布式缓存-Redis

- 集成
  - 在pom.xml中引入Redis依赖
- 配置

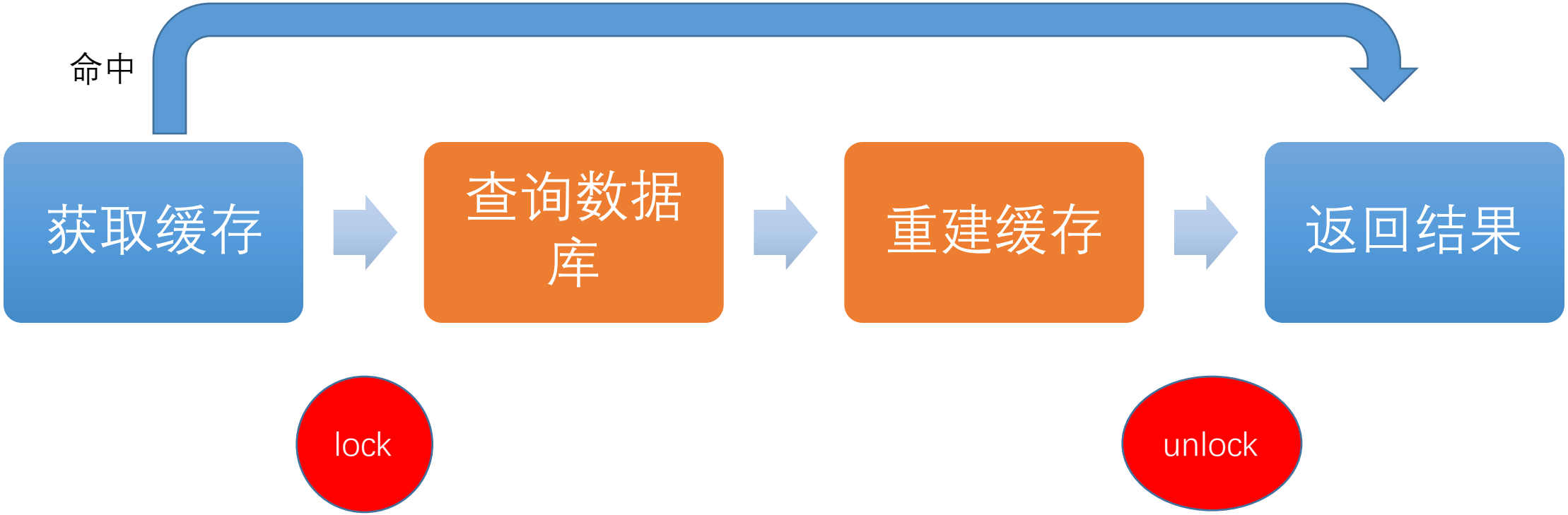
```
spring.redis.database=0
spring.redis.host=localhost
#spring.redis.password=  # Login
spring.redis.pool.max-active=8
spring.redis.pool.max-idle=8
spring.redis.pool.max-wait=-1
spring.redis.pool.min-idle=0
spring.redis.port=6379
#spring.redis.sentinel.master=  # M
#spring.redis.sentinel.nodes=  # C
#spring.redis.timeout=0
```



# 混合使用Guava&Redis

- 配置CacheManager
  - GuavaCacheConfig
  - RedisCacheConfig
- 注解
  - @Primary

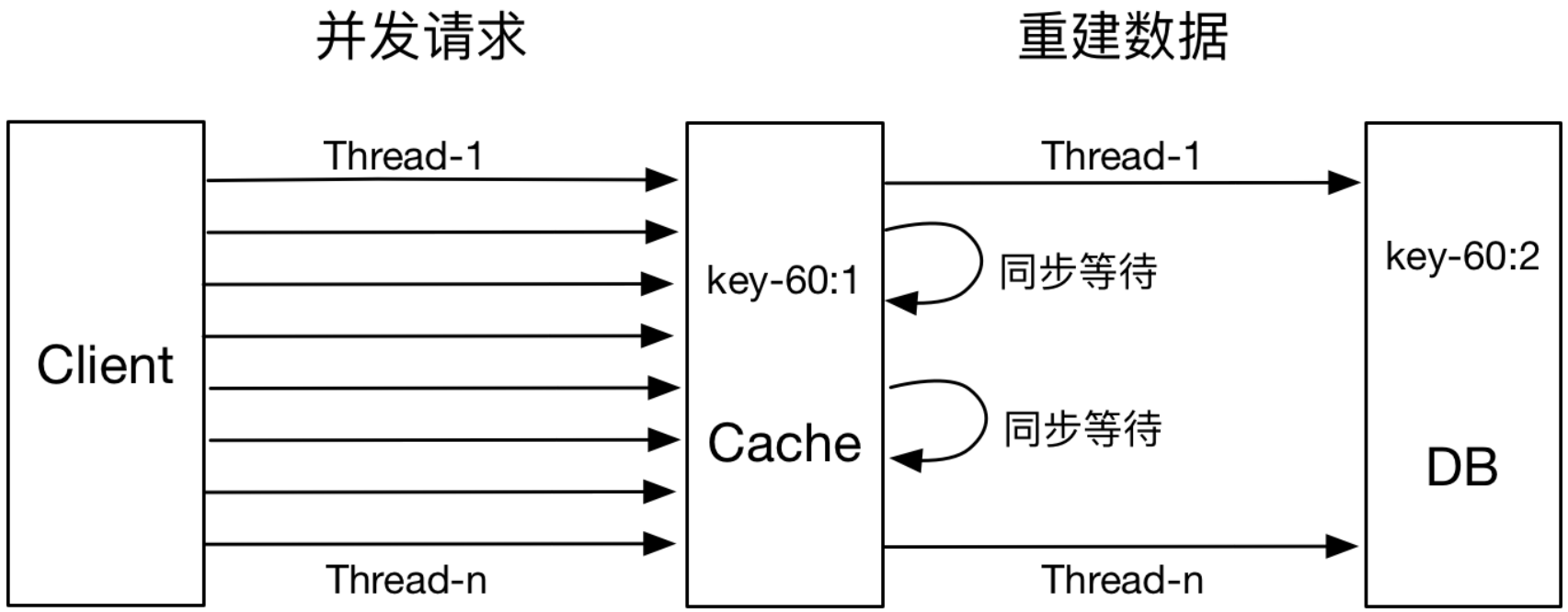
# 高并发场景-缓存穿透问题



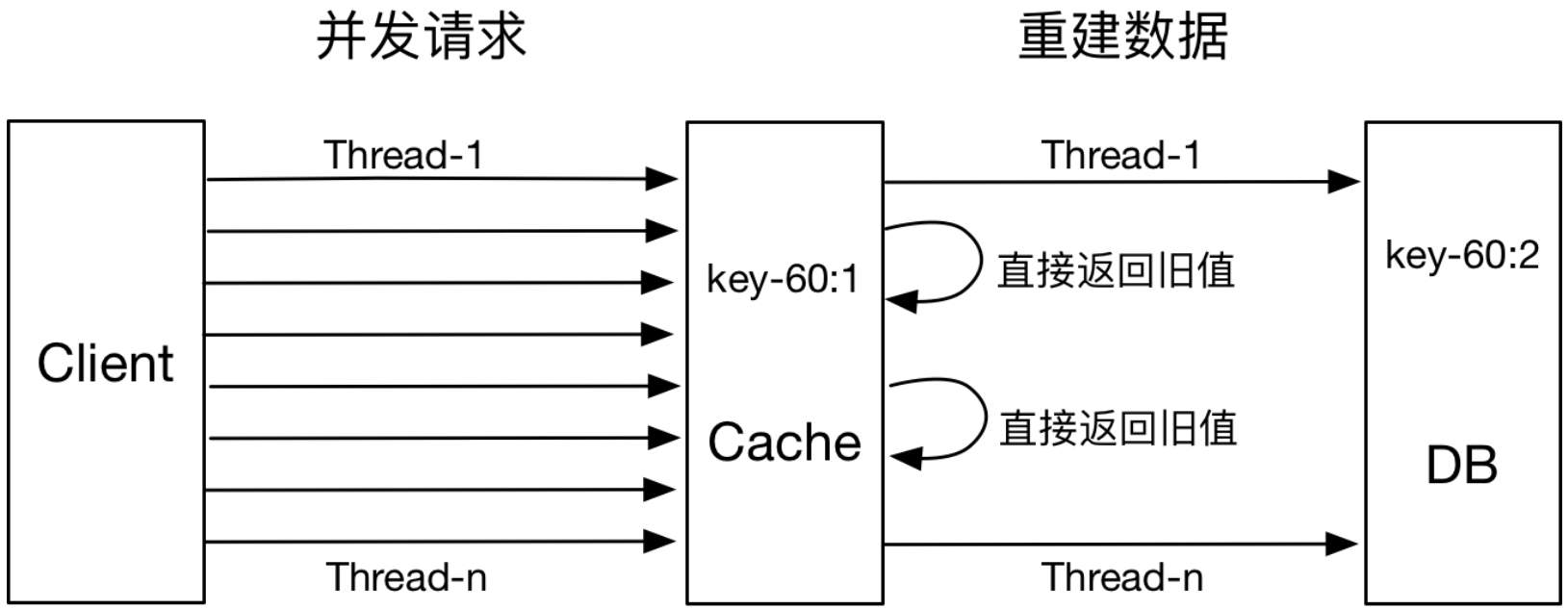
# 高并发场景-缓存穿透问题

- 问题1
  - 对于同一个key，只让一个线程去重建缓存，其他线程等待，导致系统不稳定，如何优化？
- 问题2
  - 如果是一万个key，怎么办？
- 思路
  - 两阶段过期：更新时间，小于过期时间
  - 尽量避免同时失效，散列
  - 极端情况，对重建缓存的线程进行限流

# 同步加锁

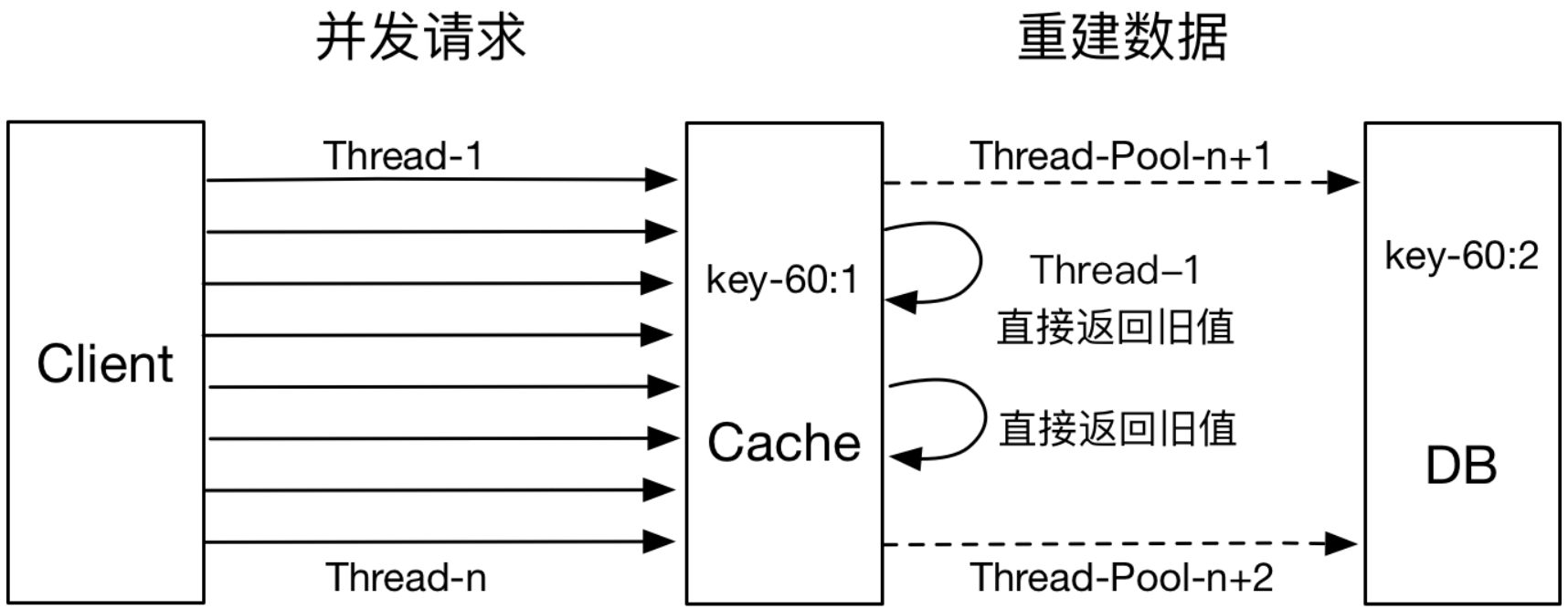

并发请求
重建数据
Client
Thread-1
Thread-n
key-60:1
Cache
同步等待
同步等待
Thread-1
Thread-n
key-60:2
DB

# 两阶段过期：更新时间，小于过期时间




key-60:1
expired-time:5s
refresh-time:3s

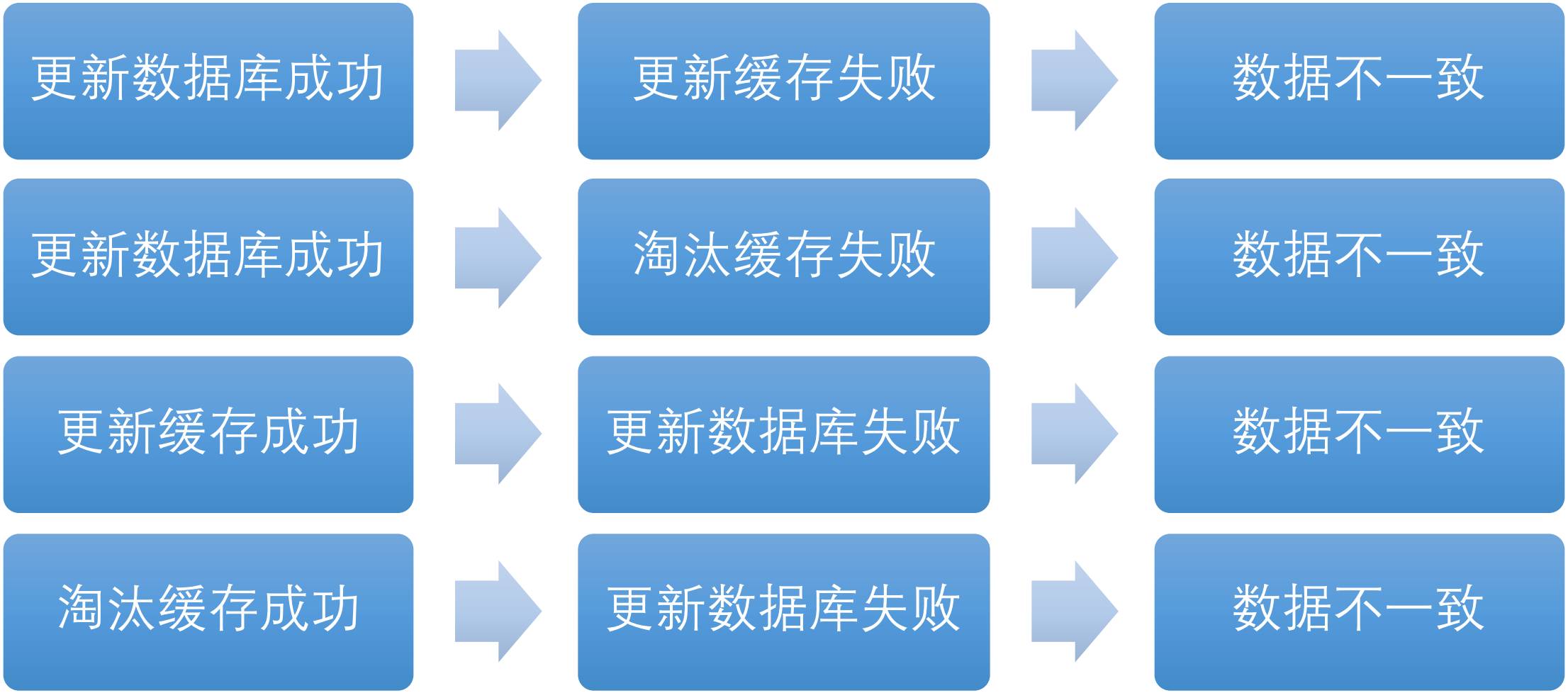
# 两阶段过期：更新时间，小于过期时间




key-60:1
expired-time:5s
refresh-time:3s

Thread-Pool-Size:100

# 高并发场景-缓存数据一致性问题



# 高并发场景-缓存数据一致性问题

- 如何避免
  - 先DB后缓存-->极端情况：周期内不一致
  - 不建议过多的重试

# 总结-注意事项

- 要缓存的 Java 对象必须实现 Serializable 接口
- 缓存的生命周期我们可以配置，然后托管 Spring CacheManager，不要试图通过 redis-cli 命令行去管理缓存。
- CacheManager 必须设置缓存过期时间，否则缓存对象将永不过期，这样做的原因如上，避免一些野数据“永久保存”。此外，设置缓存过期时间也有助于资源利用最大化，因为缓存里保留的永远是热点数据。
- 缓存适用于**读多写少**的场合，查询时缓存命中率很低、写操作很频繁等场景不适宜用缓存。
- **代码地址： git@gitee.com:fairyside/springboot.git**

# 谢谢

Java新兵

扫一扫二维码，加入该群。

**欢迎大家加入QQ群“Java新兵”，参与讨论。**

**群号:654287735**