

极客大学 Java 进阶训练营

第 21 课

分布式缓存-缓存技术



KimmKing

Apache Dubbo/ShardingSphere PMC

Apache Dubbo/ShardingSphere PMC

前某集团高级技术总监/阿里架构师/某银行北京研发中心负责人

阿里云 MVP、腾讯 TVP、TGO 会员

10 多年研发管理和架构经验

熟悉海量并发低延迟交易系统的设计实现

目录

1. 从数据的使用说起
2. 本地缓存
3. 远程缓存
4. 缓存策略
5. 缓存常见问题
6. 总结回顾与作业实践

第 21 课 1. 从数据的使用说起

我们把数据的使用频率和方式分个类

- **静态数据**：一般不变，类似于字典表
- **准静态数据**：变化频率很低，部门结构设置，全国行政区划数据等
- **中间状态数据**：一些计算的可复用中间数据，变量副本，配置中心的本地副本

- **热数据**：使用频率高
- **读写比较大**：读的频率 >> 写的频率

这些数据适合于使用缓存的方式访问

广义上来说，为了加速数据处理，让业务更快访问的临时存放冗余数据，都是缓存

狭义上，现在我们一般在分布式系统里把缓存到内存的数据叫做内存缓存

还有没有其他数据？

缓存无处不在

内存 ~ 可以看做是 CPU 和 磁盘之间的缓存

CPU与内存的处理速度也不一致，出现 L1&L2 Cache

网络处理，数据库引擎的各种Buffer，都可以看做是缓存

GUI的Double Buffer（双缓冲），是一个经典的性能优化方法

缓存的本质：

系统各级处理速度不匹配，导致利用空间换时间

缓存是提升系统性能的一个简单有效的办法

缓存加载时机

1、启动全量加载 ==> 全局有效，使用简单

2、懒加载

同步使用加载 ==>

- 先看缓存是否有数据，没有的话从数据库读取
- 读取的数据，先放到内存，然后返回给调用方

延迟异步加载 ==>

- 从缓存获取数据，不管是否为空直接返回 ==>
- 策略1异步）如果为空，则发起一个异步加载的线程，负责加载数据
- 策略2解耦）异步线程负责维护缓存的数据，定期或根据条件触发更新

缓存的有效性与数据同步

1. 为什么一般说变动频率大、一致性要求高的数据，不太适合用缓存？

变化大，意味着 **内存缓存数据** <--> **原始数据库数据**，一直有差异；
一致性要求高，意味着 只有使用**原始数据**，甚至加了**事务**，才是保险的。

2. 如何评价缓存的有效性？

读写比：对数据的写操作导致数据变动，意味着维护成本。 **N : 1**

命中率：命中缓存意味着缓存数据被使用，意味着有价值。 **90%+**

“计算机科学只存在两个难题: 缓存失效和命名。” ——Phil Karlton

对于 **数据一致性**，**性能**，**成本** 的综合衡量，是引入缓存的必须指标。

缓存使用不当导致的问题

1、系统预热导致启动慢

试想一下，一个系统启动需要预热半个小时。
导致系统不能做到快速应对故障宕机等问题。

2、系统内存资源耗尽

只加入数据，不能清理旧数据。
旧数据处理不及时，或者不能有效识别无用数据。

第 21 课 2. 本地缓存

最简单的本地缓存

```
public static final Map<String, Object> CACHE = new HashMap();
```

```
CACHE.put( "beijing" , "100001" );
```

```
String cityCode = (String) CACHE.get( "beijing" );
```

思考：还缺少什么？如何改进？

Hibernate/MyBatis都有Cache

一级缓存， session级别。

二级缓存， sessionFactory级别。

MyBatis:

```
<cache type="org.mybatis.caches.ehcache.LoggingEhcache" >  
    <property name="memoryStoreEvictionPolicy" value="LRU" /></cache>  
<select id="selectArticleListPage" resultMap="resultUserArticleList" useCache="false">
```

Hibernate:

```
<property name="hibernate.cache.provider_class">  
org.hibernate.cache.EhCacheProvider</property>  
<ehcache><diskStore path="/tmp/cache" /></ehcache>  
<cache usage="read-write" />  
<class name="Student" table="t_student" ><cache usage="read-write" /> </class >
```

Guava Cache

```
Cache<String,String> cache = CacheBuilder.newBuilder()  
    .maximumSize(1024)  
    .expireAfterWrite(60,TimeUnit.SECONDS)  
    .weakValues()  
    .build();  
cache.put("word","Hello Guava Cache");  
System.out.println(cache.getIfPresent("word"));
```

此外，还可以显式清除、统计信息、移除事件的监听器、自动加载等功能。

Spring Cache

- 1、基于注解和AOP，使用非常方便
- 2、可以配置Condition和SPEL，非常灵活
- 3、需要注意：绕过Spring的话，注解无效

核心功能：@Cacheable、@CachePut、@CacheEvict

参考：<https://developer.ibm.com/zh/articles/os-cn-spring-cache/>

第 21 课 3. 远程缓存

考虑一下本地缓存有什么缺点？

- 1、在多个集群环境同步？当集群规模增大，缓存的读写放大。
- 2、在JVM中长期占用内存？如果是堆内存，总是会影响GC。
- 3、缓存数据的调度处理，影响执行业务的线程，抢资源。

==> 集中处理缓存

聪明的你，思考一下：有什么缺点呢？

Redis/Memcached 缓存中间件

REmote DIctionary Server(Redis) 是一个由Salvatore Sanfilippo写的key-value存储系统。Redis是一个开源的使用ANSI C语言编写、遵守BSD协议、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API。

Memcached是以LiveJournal旗下Danga Interactive公司的Brad Fitzpatric为首开发的一款开源高性能，分布式内存对象缓存系统。

Redis 官网：<https://redis.io/>

Redis 在线测试：<http://try.redis.io/>

Redis 命令参考：<http://doc.redisfans.com/>

《Redis 设计与实现》：<http://redisbook.com/>

Memcached 官网：<https://memcached.org/>



Hazelcast/Ignite 内存网格



第 21 课 4. 缓存策略

容量

资源有限

- 缓存数据容量是必须要考虑的问题
- 思考系统的设计容量、使用容量、峰值，应该是我们做架构设计的一个常识

过期策略

- 按FIFO或LRU
- 按固定时间过期
- 按业务时间加权：例如 $3+5x$

第 21 课 5. 缓存常见问题

缓存穿透

问题：大量并发查询不存在的KEY，导致都直接将压力透传到数据库。

分析：为什么会多次透传呢？不存在一直为空。

需要注意让缓存能够区分KEY不存在和查询到一个空值。

解决办法：

- 1、缓存空值的KEY，这样第一次不存在也会被加载会记录，下次拿到有这个KEY。
- 2、Bloom过滤或RoaringBitmap 判断KEY是否存在。
- 3、完全以缓存为准，使用 延迟异步加载 的策略2，这样就不会触发更新。

缓存击穿

问题：某个KEY失效的时候，正好有大量并发请求访问这个KEY。

分析：跟前面一个其实很像，属于比较偶然的。

解决办法：

- 1、KEY的更新操作添加全局互斥锁。
- 2、完全以缓存为准，使用延迟异步加载 的策略2，这样就不会触发更新。

缓存雪崩

问题：当某一时刻发生大规模的缓存失效的情况，会有大量的请求进来直接打到数据库，导致数据库压力过大升值宕机。

分析：一般来说，由于更新策略、或者数据热点、缓存服务宕机等原因，可能会导致缓存数据同一个时间点大规模不可用，或者都更新。所以，需要我们的更新策略要在时间上合适，数据要均匀分散，缓存服务器要多台高可用。

解决办法：

- 1、更新策略在时间上做到比较均匀。
- 2、使用的热数据尽量分散到不同的机器上。
- 3、多台机器做主从复制或者多副本，实现高可用。
- 4、实现熔断限流机制，对系统进行负载能力控制。

第 21 课 6.总结回顾与作业实践

第 21 课总结回顾

从数据库到缓存

本地与远程缓存

缓存策略与使用

缓存常见的问题

第 21 课作业实践

- 1、（选做）按照课程内容，动手验证Hibernate和Mybatis缓存。
- 2、（选做）使用spring或guava cache，实现业务数据的查询缓存。
- 3、（挑战☆）编写代码，模拟缓存穿透，击穿，雪崩。
- 4、（挑战☆☆）自己动手设计一个简单的cache，实现过期策略。

2021，找到唯一。

在这个艰难的世界里，每天进步一点点，
坚守头顶的星空和心中的信念，成为更好的自己。

THANKS! |  极客大学