优质IT资源微信x923713

《手写OS操作系统》小班二期招生，全程直播授课，大牛带你掌握硬核技术！ 点此查

免费课 实战课 体系课 慕课教程 专栏 手记 企业服务

从所有教程的词条中查询

bobby · 更新于 2022-11-16

上一节 6. redlock详解 1. 表结构设计和... 下一节

常见的分布式锁实现方案

1. 基于mysql的
    1. 悲观锁
    2. 乐观锁
2. 基于redis的分布式锁
3. 基于zookeeper的分布式锁

1. 基于mysql来实现

1. 悲观锁

悲观锁与乐观锁是人们定义出来的概念，你可以理解为一种思想，是处理并发资源的常用手段。
不要把他们与mysql中提供的锁机制(表锁，行锁，排他锁，共享锁)混为一谈。
一、悲观锁
顾名思义，就是对于数据的处理持悲观态度，总认为会发生并发冲突，获取和修改数据时，别人会修改数据。所以在整个数据处理过程中，需要将数据锁定。
悲观锁的实现，通常依靠数据库提供的锁机制实现，比如mysql的排他锁，select ... for update来实现悲观锁。
例子：商品秒杀过程中，库存数量的减少，避免出现超卖的情况。

代码块

```
CREATE TABLE `tb_goods_stock` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT 'ID',
  `goods_id` bigint(20) unsigned DEFAULT '0' COMMENT '商品ID',
  `nums` int(11) unsigned DEFAULT '0' COMMENT '商品库存数量',
  `create_time` datetime DEFAULT NULL COMMENT '创建时间',
  `modify_time` datetime DEFAULT NULL COMMENT '更新时间',
  PRIMARY KEY (`id`),
  UNIQUE KEY `goods_id` (`goods_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='商品库存表';
```

将商品库存数量nums字段类型设为unsigned，保证在数据库层面不会发生负数的情况。
注意，使用悲观锁，需要关闭mysql的自动提交功能，将 set autocommit = 0;
注意，mysql中的行级锁是基于索引的，如果sql没有走索引，那将使用表级锁把整张表锁住。
1、开启事务，查询要卖的商品，并对该记录加锁。

意见反馈 收藏教程 标记书签

优质IT资源微信x923713

代码块

```
begin;
select nums from tb_goods_stock where goods_id = {$goods_id} for update
```

2、判断商品数量是否大于购买数量。如果不满足，就回滚事务。
3、如果满足条件，则减少库存，并提交事务。

代码块

```
update tb_goods_stock set nums = nums - {$num} where goods_id = {$goods
commit;
```

事务提交时会释放事务过程中的锁。
悲观锁在并发控制上采取的是先上锁然后再处理数据的保守策略，虽然保证了数据处理的安全性，但也降低了效率。

gorm如何实现for update
https://gorm.io/zh_CN/docs/advanced_query.html

2. 乐观锁

顾名思义，就是对数据的处理持乐观态度，乐观的认为数据一般情况下不会发生冲突，只有提交数据更新时，才会对数据是否冲突进行检测。
如果发现冲突了，则返回错误信息给用户，让用户自己决定如何操作。
乐观锁的实现不依靠数据库提供的锁机制，需要我们自己实现，实现方式一般是记录数据版本，一种是通过版本号，一种是通过时间戳。
给表加一个版本号或时间戳的字段，读取数据时，将版本号一同读出，数据更新时，将版本号加1。
当我们提交数据更新时，判断当前的版本号与第一次读取出来的版本号是否相等。如果相等，则予以更新，否则认为数据过期，拒绝更新，让用户重新操作。

代码块

```
CREATE TABLE `tb_goods_stock` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT 'ID',
  `goods_id` bigint(20) unsigned DEFAULT '0' COMMENT '商品ID',
  `nums` int(11) unsigned DEFAULT '0' COMMENT '商品库存数量',
  `create_time` datetime DEFAULT NULL COMMENT '创建时间',
  `modify_time` datetime DEFAULT NULL COMMENT '更新时间',
  `version` bigint(20) unsigned DEFAULT '0' COMMENT '版本号',
  PRIMARY KEY (`id`),
  UNIQUE KEY `goods_id` (`goods_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COMMENT='商品库
```

1、查询要卖的商品，并获取版本号。

代码块

```
begin;
select nums, version from tb_goods_stock where goods_id = {$goods_id};
```

2、判断商品数量是否大于购买数量。如果不满足，就回滚事务。
3、如果满足条件，则减少库存。(更新时判断当前version与第1步中获取的version是否相同)

代码块

```
ion + 1
```

优质IT资源微信x923713

4、判断更新操作是否成功执行，如果成功，则提交，否则就回滚。
乐观锁是基于程序实现的，所以不存在死锁的情况，适用于读多的应用场景。如果经常发生冲突，上层应用不断的让用户进行重新操作，这反而降低了性能，这种情况下悲观锁就比较适用。

3. 优缺点

优点：

1. 简单
2. 不需要额外的组件 - 维护，mysql的维护比较简单 - 最合适的才是最好的。 系统的可用性

缺点：
性能

---

2. 基于redis的分布式锁

分布式锁需要解决的问题：
互斥性：任意时刻只能有一个客户端拥有锁，不能同时多个客户端获取
安全性：锁只能被持有该锁的用户删除，而不能被其他用户删除
死锁：获取锁的客户端因为某些原因而宕机，而未能释放锁，其他客户端无法获取此锁，需要有机制来避免该类问题的发生

1. 代码异常，导致无法运行到release
2. 你的当前服务器网络出问题 - 脑裂
3. 断电

容错：当部分节点宕机，客户端仍能获取锁或者释放锁

如何解决上述问题的发生 - 设置过期时间
过期设置会产生新的问题：

1. 当前的线程如果在一段时间后没有执行完， 当前的程序没有执行完，然后key过期了
    1. 不安全
    2. 另一个线程进来以后会将当前的key给删除掉， 另一个线程删除掉了本该属于我设置的值
    3. 如果当前的线程没有执行完，那我的这个线程还应该在适当的时候去续租，将过期时间重新设置
        1. 应该在什么时候去设置过期 - 15的2/3的时候去续租，也就是运行10s以后去将过期时间重新设置为15s
        2. 如果去定时的完成这个续租的过程 - 启动一个线程去完成

6. redlock详解 上一节 下一节 1. 表结构设计和proto文件定义

我要提出意见反馈
意见反馈 收藏教程 标记书签

优质IT资源微信x923713

企业服务 网站地图 网站首页 关于我们 联系我们 讲师招募 帮助中心 意见反馈 代码托管

Copyright © 2022 imooc.com All Rights Reserved | 京ICP备 12003892号-11 京公网安备11010802030151号

意见反馈 收藏教程 标记书签