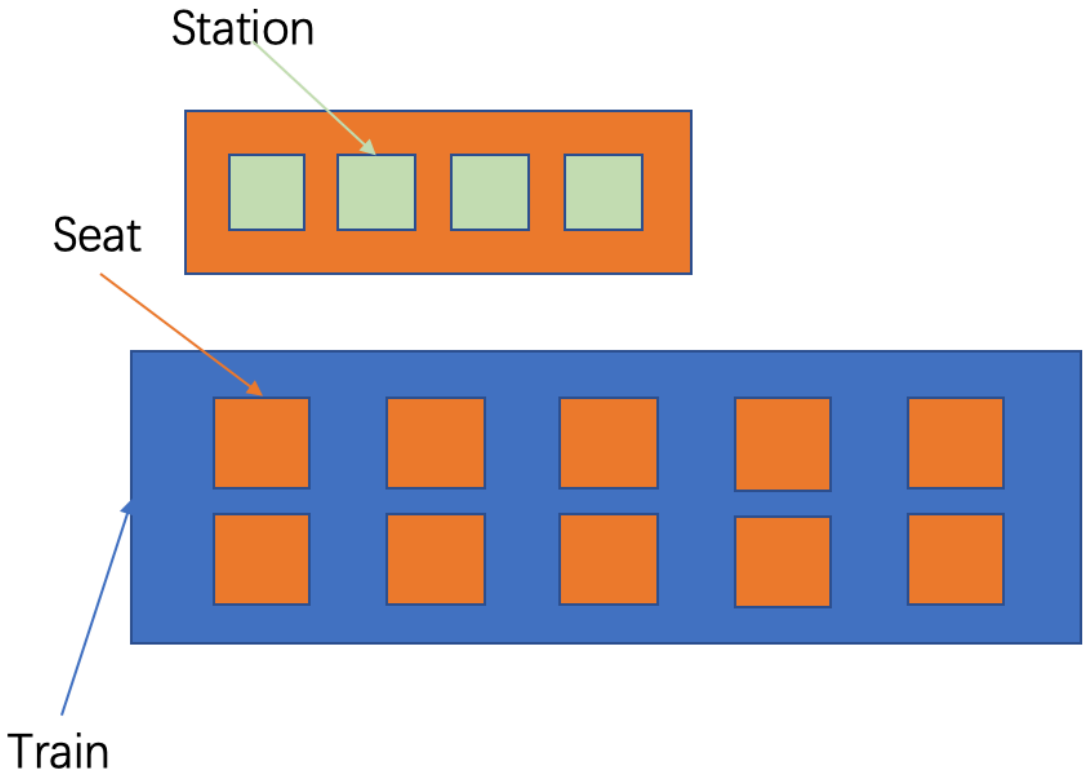
邓泽一 2020E8018682102 网安 2005

1 并发数据结构

抽象模型



图一

如图一所示，蓝色矩形代表火车，其上橙色的方块代表火车上的每个座位（因为查询、购买、退票的三个方法都没有关于车厢的参数，所以我们在数据抽象时可以不用考虑火车的车厢数，而将整个火车视为一个车厢），橙色方块上的第 i 个青色方块代表该座位在站点 i(i<stationNum)和站点 i+1 站点之间的路程被占用的情况。

数据结构

最重要的数据结构其实是 Seat 类。

```
class Seat {
    private Lock lock;//可重入锁
    private BitMap bitmap;//图一中的青色方块
}
```

Seat 类上的 BitMap 类。

```
class BitMap {
    private AtomicLongArray map;//记录座位在哪些区间被占用
}
```

map 对象用来记录座位在哪些区间被占用。

查询

查询时遍历所有座位，检查其中的 AtomicLongArray 对象 map 来判断该座位是否满足满足要求。

查询是对座位的 atomic 对象 map 进行查询，该方法是无等待的。

购买

购票时系统在阈值次数内采用随机分配的方式产生车票，并对该车票进行检查，即查询该车票是否可购买，如果可购买，则对该座位进行加锁（lock），之后再重复一次检查，若依旧可购买，则对该 Seat 上的 bitmap 进行更改，之后解锁，其中的任何一步失败则重新随机产生车票。如果随机购票次数超出了设定的阈值，则遍历本车所有的座位，直到购票成功，或者由于无票而返回。

购票时用了锁，该方法是无死锁但可能饥饿。

退票

退票时先检查想要退的车票是否合法。若合法，则修改该 Seat 上的 AtomicLongArray 对象 map。

退票是对座位的 atomic 对象 map 进行操作，该方法是无等待的。

2 多线程测试程序

按照查询，购票，退票的比例，随机产生请求并统计每一次请求所需的平均时间和总的吞吐量。

3 系统的正确性和性能

正确性

单线程测试使用老师提供的工具，测试结果为通过。

多线程测试使用“计算所胡起”同学的多线程测试程序。结果如图二所示，通过。

```
[INFO] Results:
[INFO]
[INFO] Tests run: 74, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  02:18 min
[INFO] Finished at: 2020-12-21T20:24:29+08:00
[INFO] ------------------------------------------------------------------------
```

图二

性能

ThreadNum: 4 BuyAvgTime(ns): 44115 RefundAvgTime(ns): 3283 InquiryAvgTime(ns): 46688 ThroughOut(t/s): 85000

ThreadNum: 8 BuyAvgTime(ns): 72681 RefundAvgTime(ns): 2747 InquiryAvgTime(ns): 54700 ThroughOut(t/s): 125000

ThreadNum: 16 BuyAvgTime(ns): 85077 RefundAvgTime(ns): 2168 InquiryAvgTime(ns): 64833 ThroughOut(t/s): 231000

ThreadNum: 32 BuyAvgTime(ns): 75964 RefundAvgTime(ns): 3525 InquiryAvgTime(ns): 46755 ThroughOut(t/s): 558000

ThreadNum: 64 BuyAvgTime(ns): 158049 RefundAvgTime(ns): 7977 InquiryAvgTime(ns): 83231 ThroughOut(t/s): 529000

图三

测试设备课程所用服务器。