淘宝网 Taobao.com

DTCC2013

秒杀场景下MySQL的低效
--原因和改进

@淘宝丁奇

1. 秒杀/热卖商品背景
2. 性能问题
3. 几种解决方案

秒杀/热卖商品背景

a)商品库存是有量的

b)买家下单以后库存要减掉

c)不能减成负数

秒杀/热卖商品背景

基本逻辑

- Start transaction
- Insert …
- Insert …
- Update set 库存 = 库存 – n where …
- Commit

秒杀/热卖商品背景

秒杀/热卖商品背景

多个用户
Start transaction
Insert …
Insert …
Update set 库存 where 商品id…（互相等待）
Commit

原因：InnoDB的行锁

性能问题

原因是：

行锁串行导致性能下降吗？

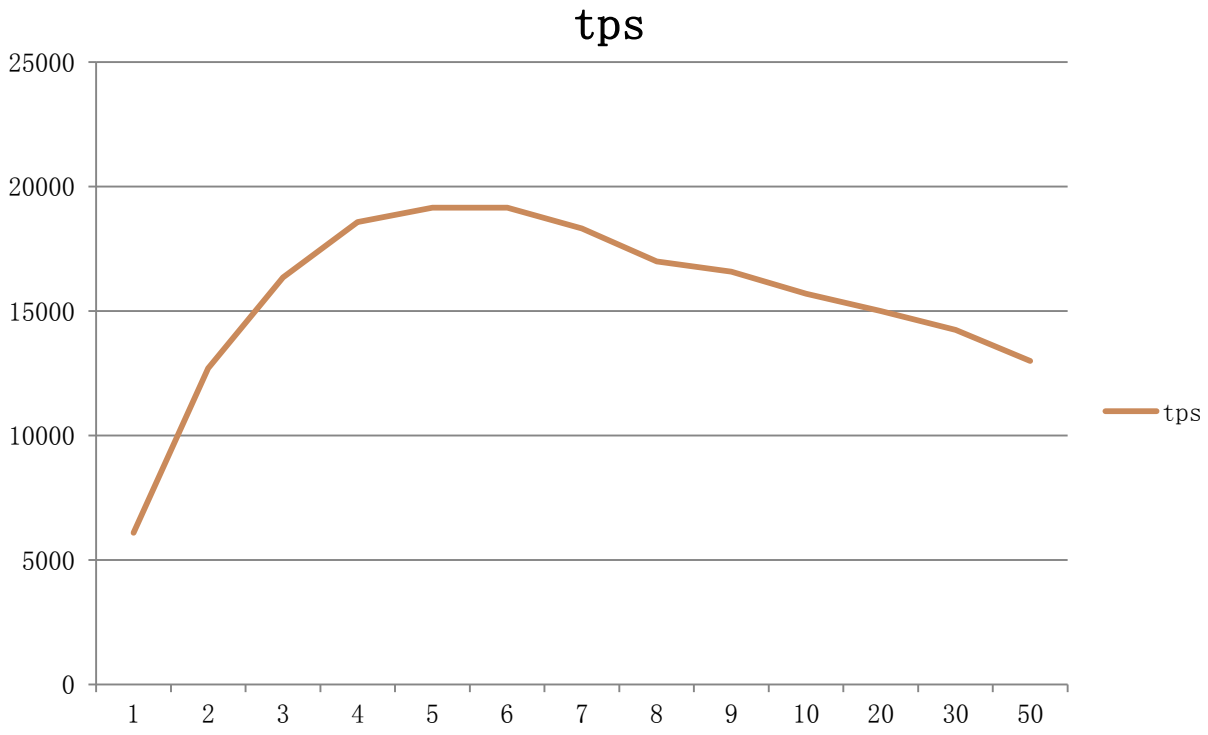
分析:

问题可以简化为，很多线程更新同一行

```
CREATE TABLE `t1` (
  `a` int(11) NOT NULL,
  `b` int(11) DEFAULT NULL,
  PRIMARY KEY (`a`)
) ENGINE=InnoDB;
Insert into t1 values($i, 2147483647);
```

插入50行数据

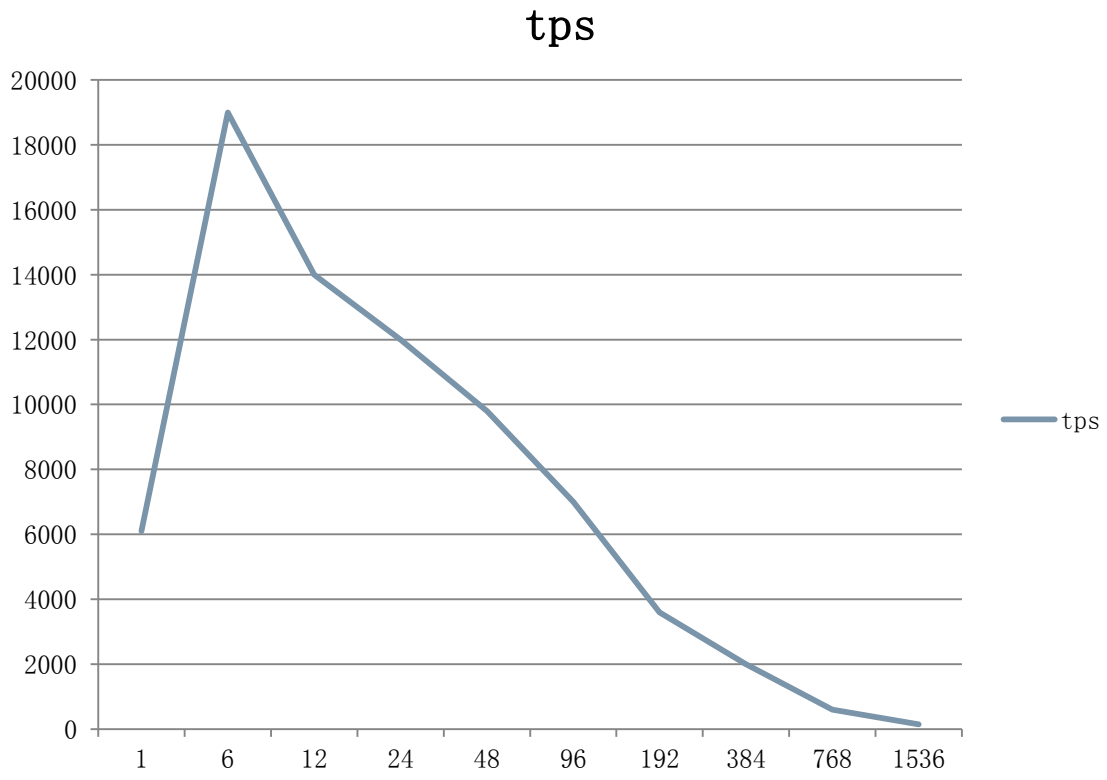
性能问题



线程数	TPS
1	6100
2	12700
3	16357
4	18570
5	19160
6	19160
7	18320
8	17000
9	16577
10	15700
20	15000
30	14250
50	13000

阶段结论： 6个线程并发各自更新一行性能最高

性能问题—现象



线程数	tps
1	6100
6	19160
12	14000
24	10000
48	9800
96	7000
192	3600
384	2000
768	600
1536	150

性能问题—分析

1、随着并发线程增加，tps急剧下降

2、每个商品有128个线程并发请求的时候，tps已经跌到600
不可接受

3、 坏消息是，秒杀的时候，一个商品何止128个人来抢？

性能问题—分析

当并发线程多时，MySQL在做什么

```
46.61%  mysqld                [.] lock_deadlock_recursive
 5.30%  mysqld                [.] lock_rec_convert_impl_to_expl
 2.65%  [kernel]              [k] find_busiest_group
 1.26%  libc-2.12.so          [.] vfprintf
 1.14%  [kernel]              [k] _spin_lock
 1.08%  libpthread-2.12.so    [.] pthread_cond_wait@@GLIBC_2.3.2
 0.92%  mysqld                [.] mutex_delay
 0.91%  [kernel]              [k] futex_wake
```

因此，串行并不是问题，问题是InnoDB内部存在太多线程

在那之前，先回顾这张图片

解决方案 1 -- 关掉死锁检测

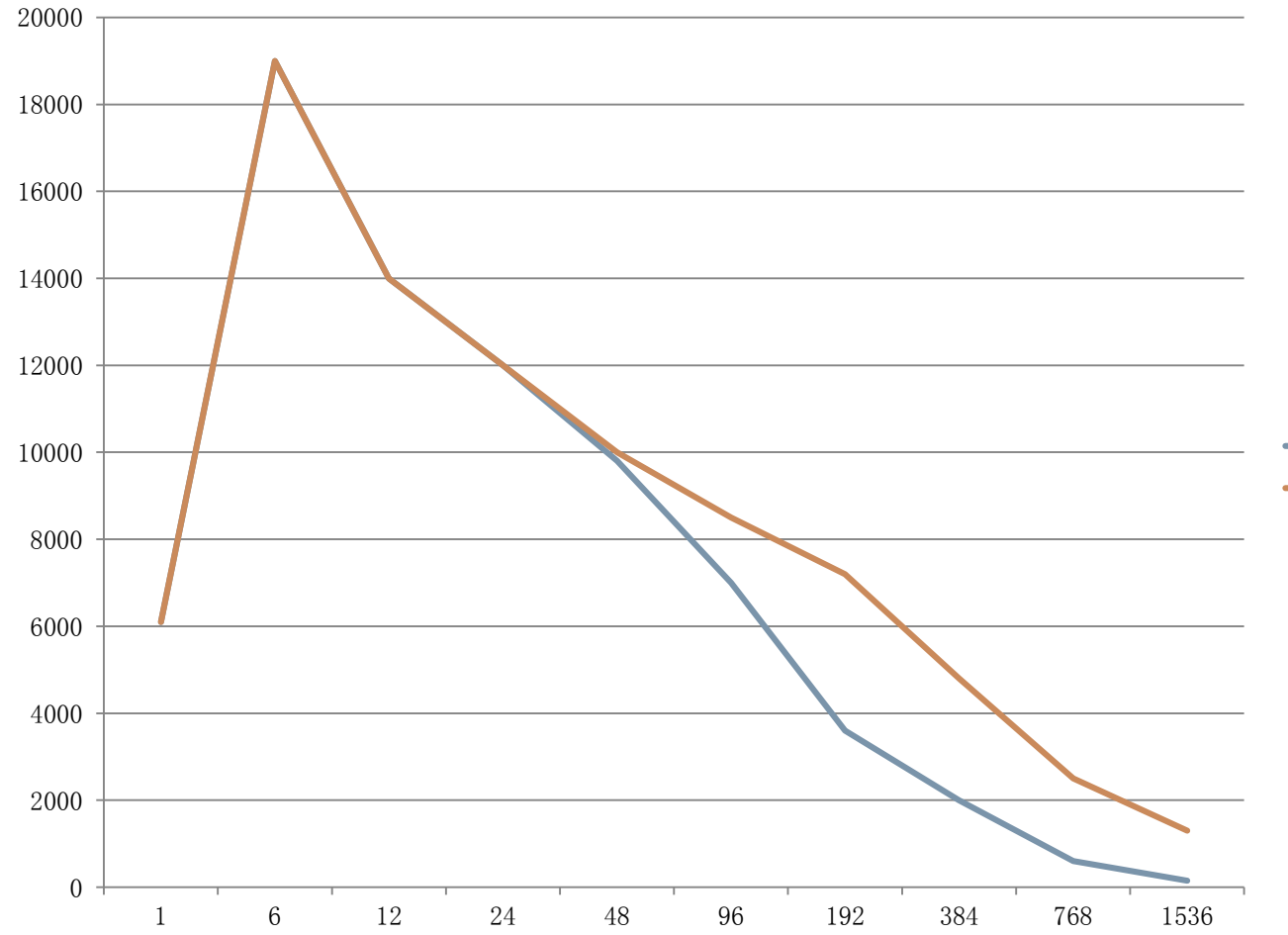
方案特点：

- 很直接
- 很暴力

  正常的业务死锁会变成超时
- 不治标

  除掉老大，还有老二，问题的症结没有解决

性能效果



线程数	TPS1	TPS2
1	6100	6100
6	19000	19000
12	14000	14000
24	12000	12000
48	9800	10000
96	7000	8500
192	3600	7200
384	2000	4800
768	600	2500
1536	150	1300

虽然啊在1536线程时性能是8倍，但依然不可接受

性能效果分析

并发线程

$o(n^2)$

$o(n^2)$

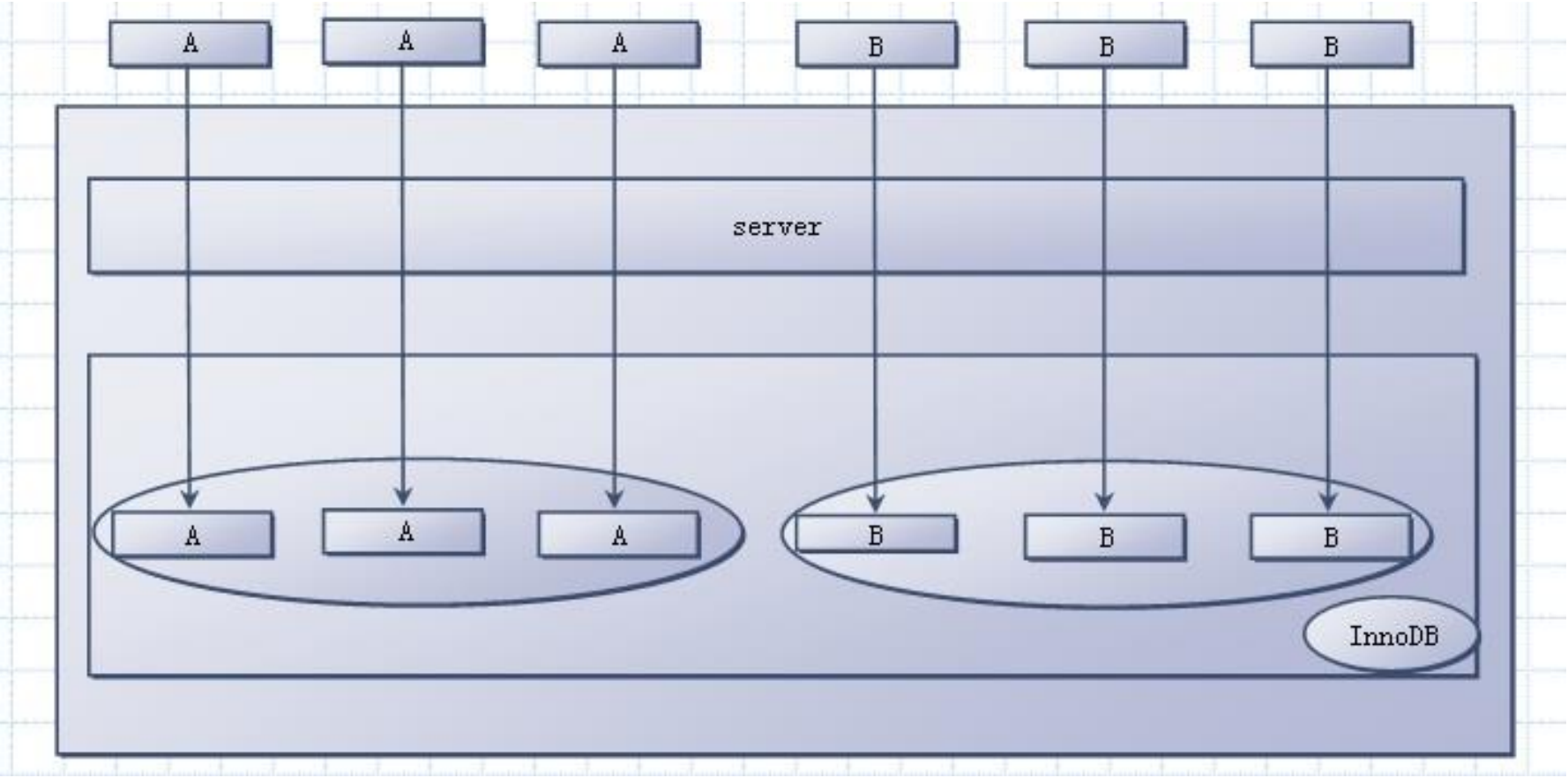
为什么平时我们没有这个问题？

固定车道的公路，最流畅的方式是什么

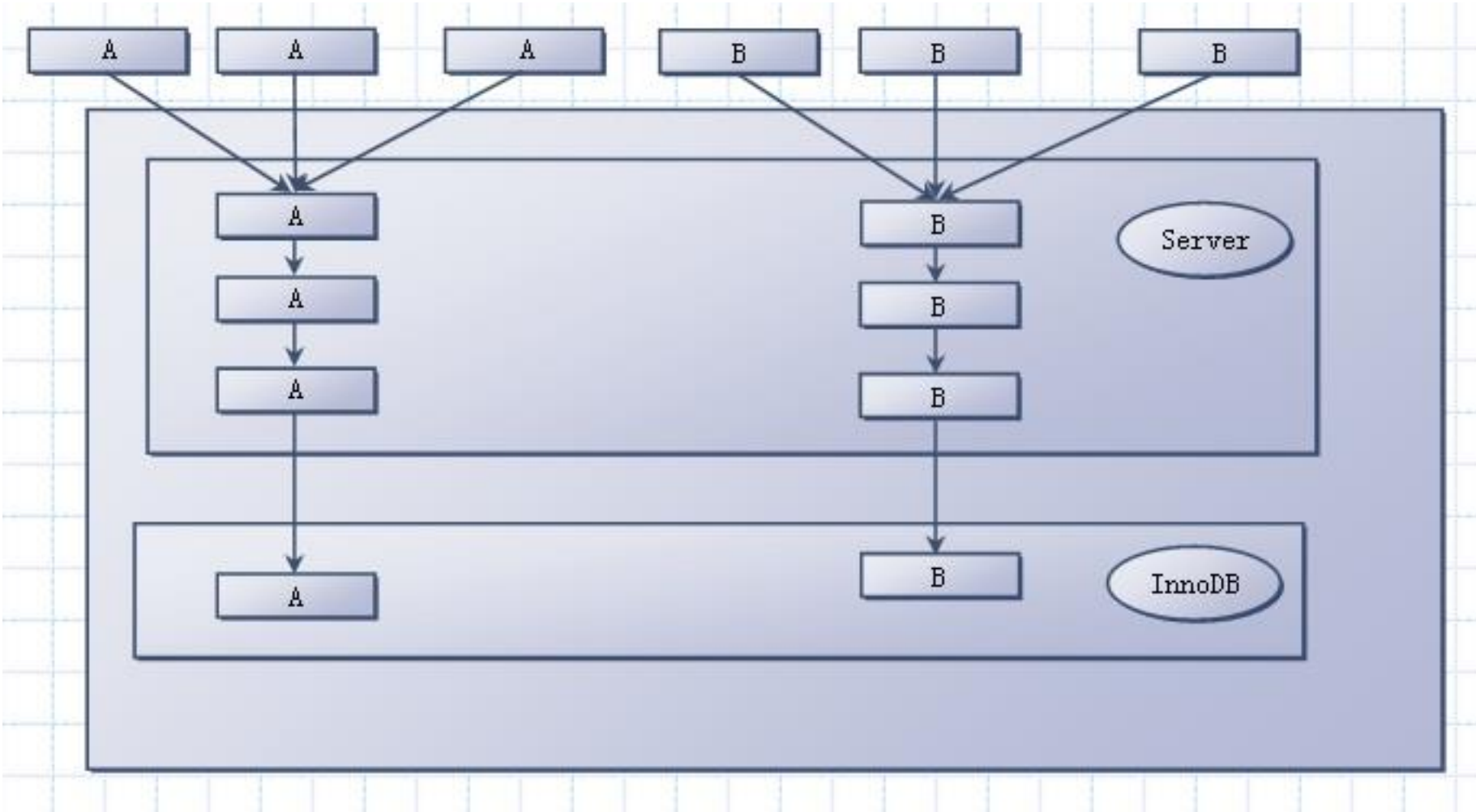
解决方案 2 – 系统最优值

- 在固定的硬件条件下、每个系统都有一个对应的状态最优值
- InnoDB的线程数
- 将排队队列提到进入引擎层前

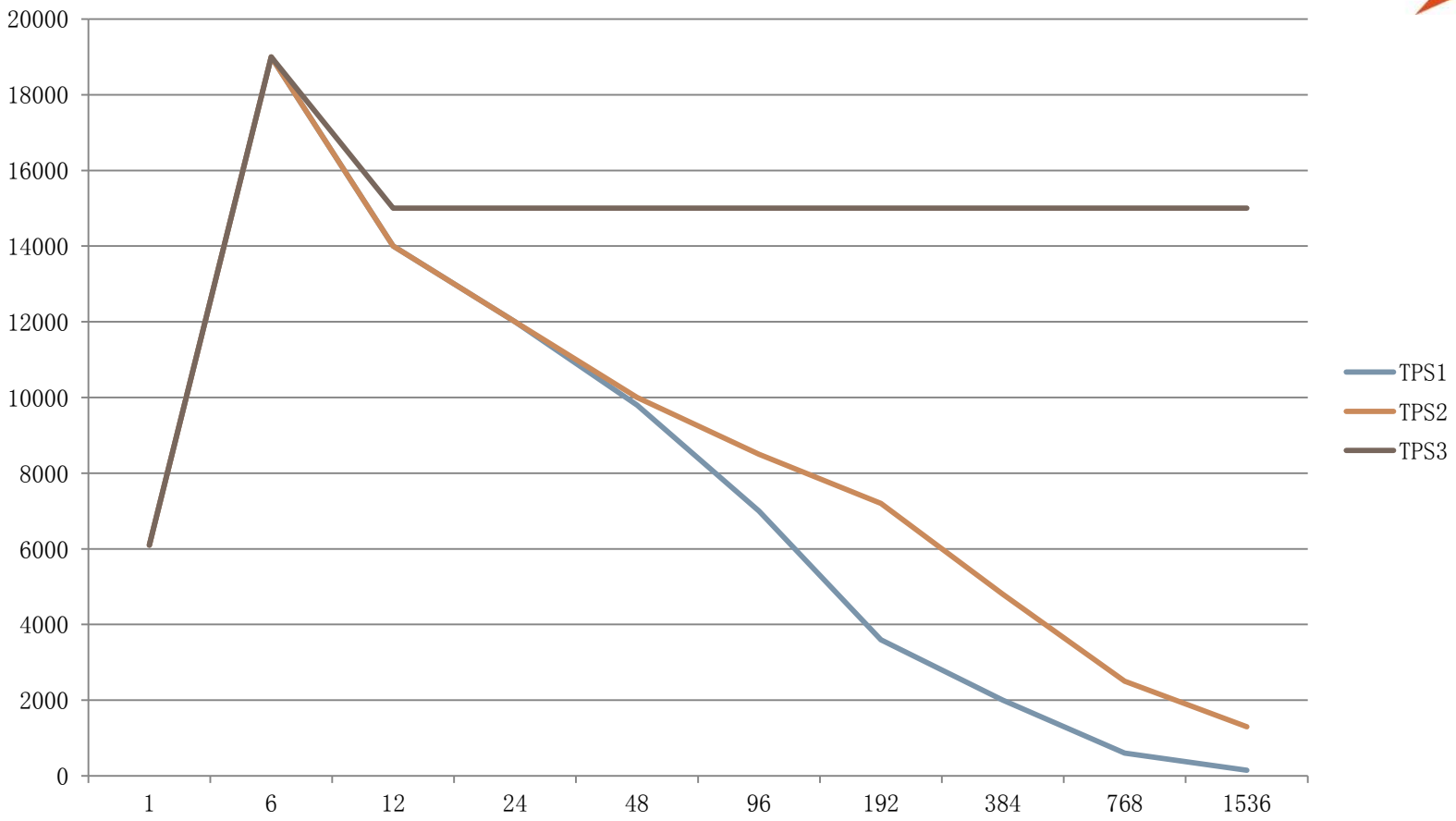
解决方案 2 – 排队



解决方案 2 – 排队



解决方案 2 – 性能



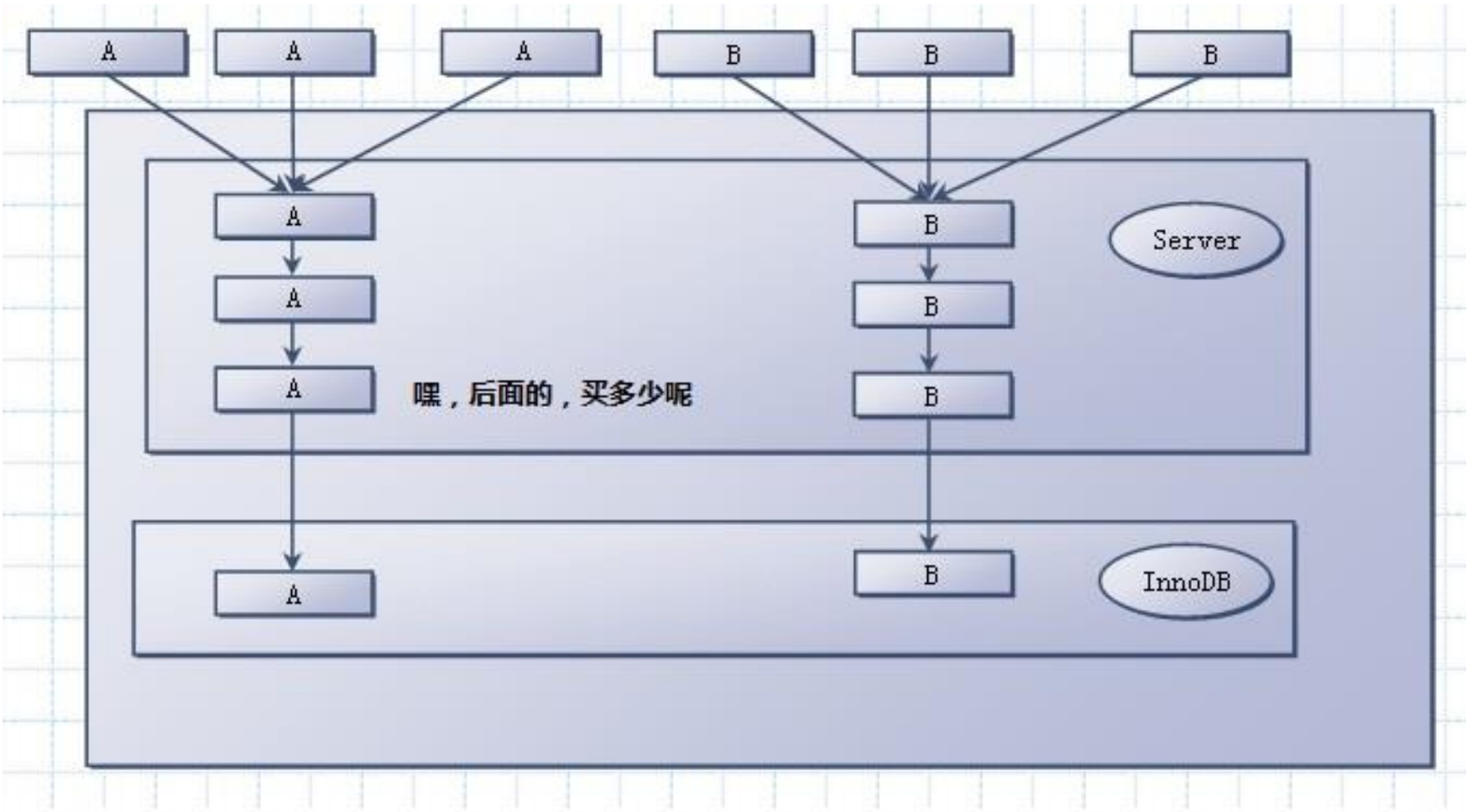
TPS不随着线程数增加而下跌，维持在1.5w

Update /*UPDATE_STOCK 1*/ t set b=b-1 where a=1;

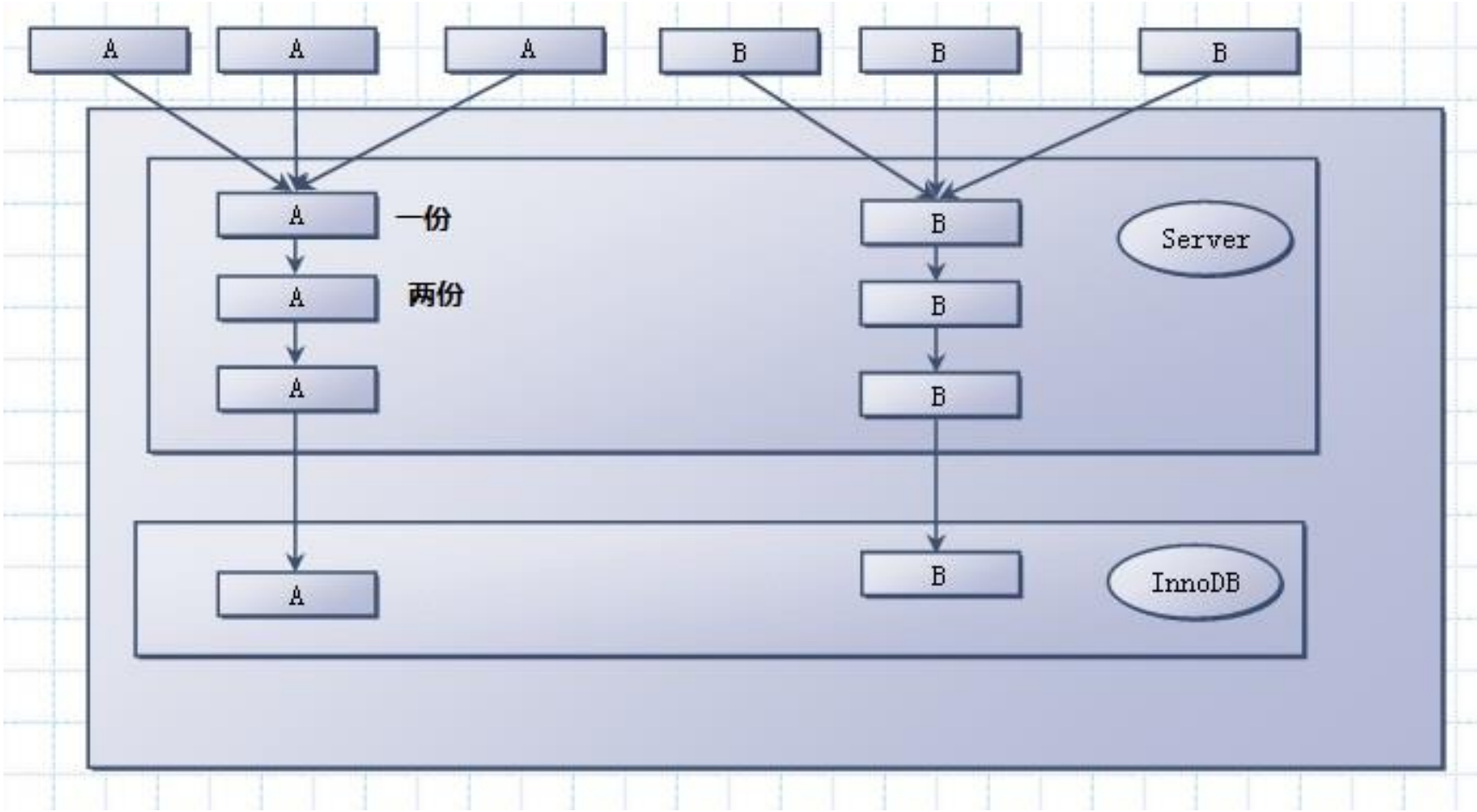
解决方案 3 – 排队优化

- 想想排队买票的事儿
- 排队优化，成组提交

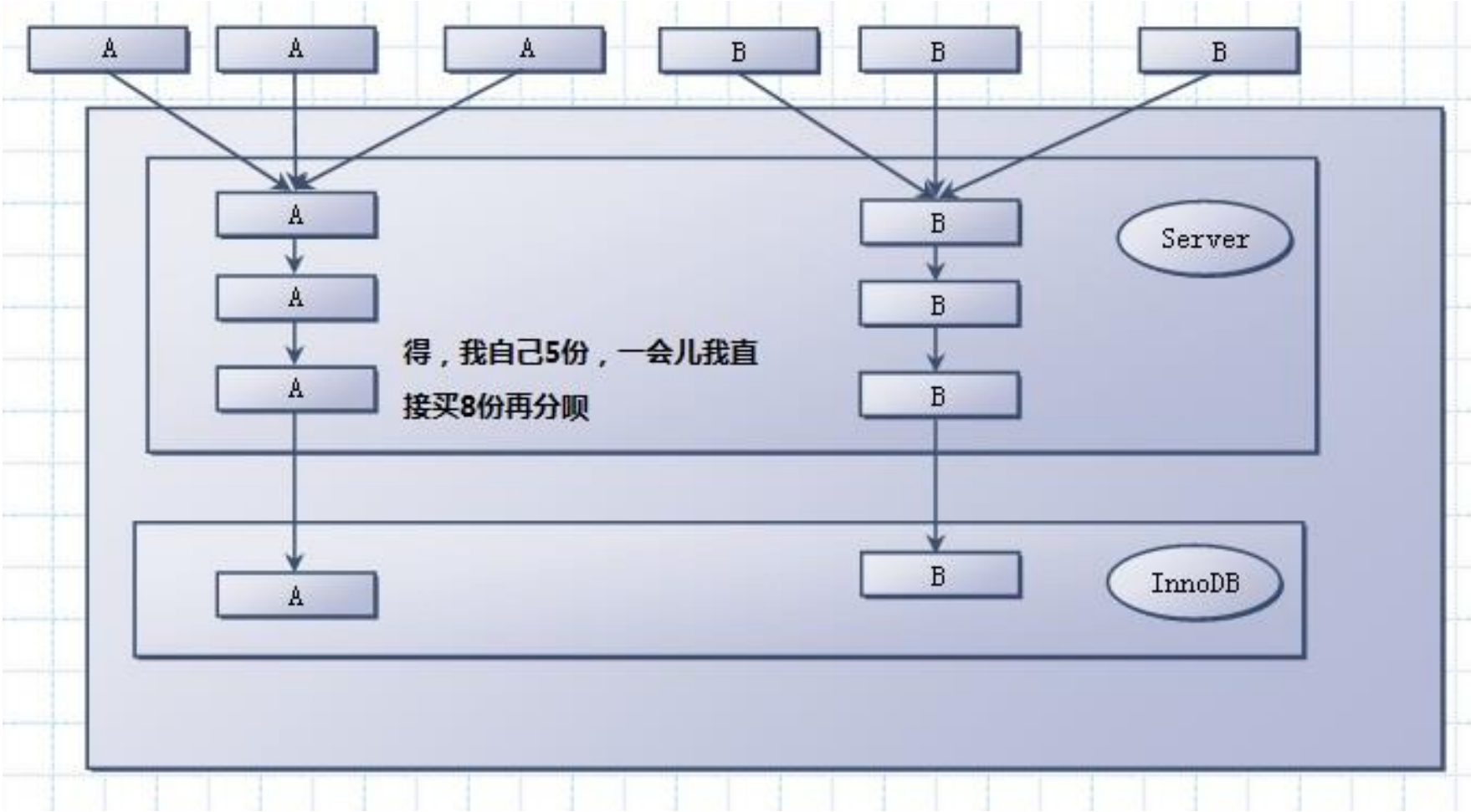
解决方案 2 – 排队优化



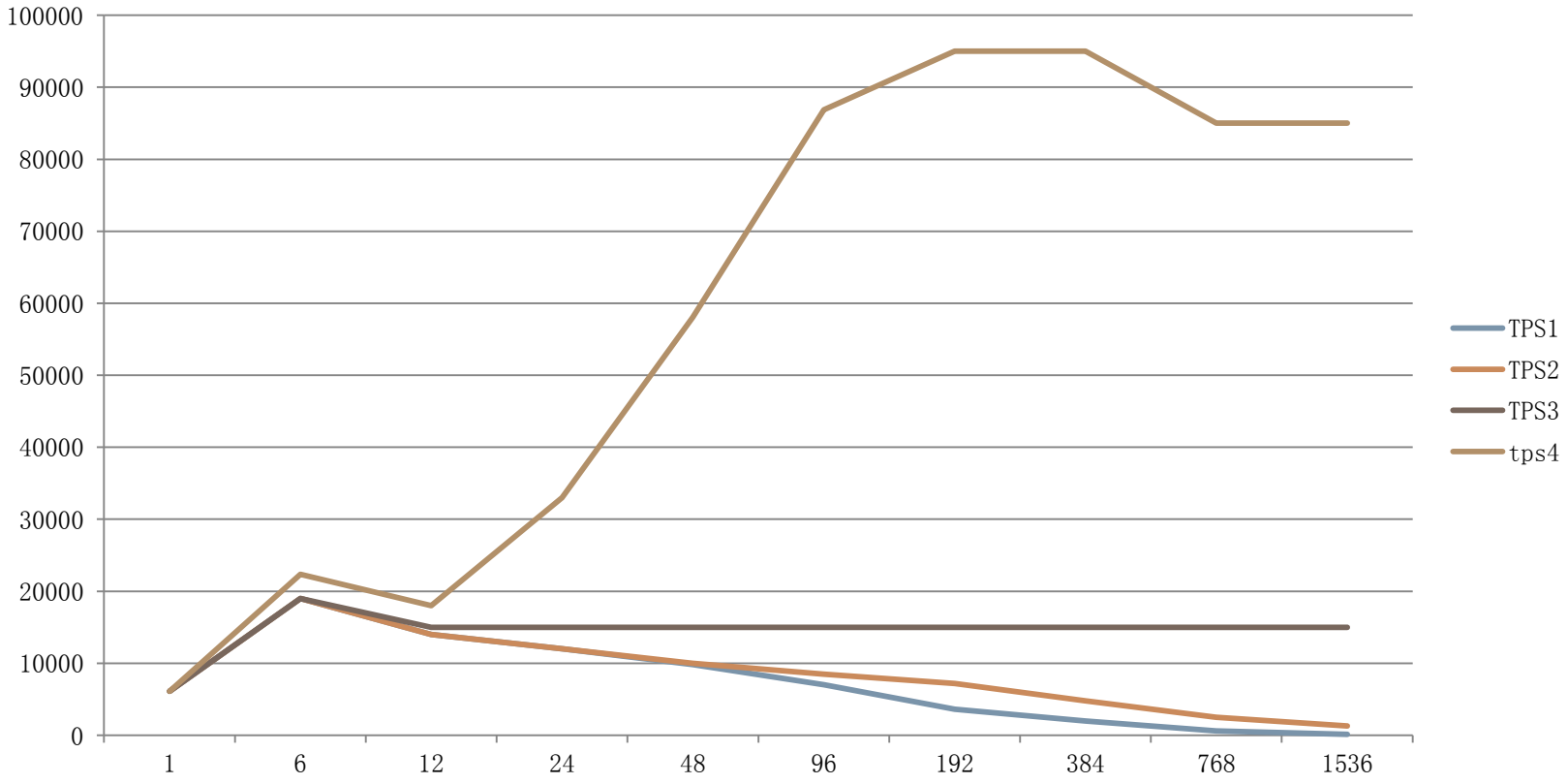
解决方案 2 – 排队优化



解决方案 2 – 排队优化



解决方案 3 – 性能对比



TPS 稳定在8.5w

解决方案 3—限制

语句：

```
update /*UPDATE_STOCK 6784 1 '1' 'H<=Fquantity‘*/
tbl_name
   set withholding_quantity = withholding_quantity + 1,
  gmt_modified = NOW()
   where auction_id = 6784
   and (withholding_quantity - 1) <= quantity;
```

“太定制”

1、跟业务逻辑绑定较紧密

2、使用常见必须符合“可组提交”的限制

```
| Com_deposit                   | 71048          |
| Com_deposit                   | 70943          |
| Com_deposit                   | 66216          |
| Com_deposit                   | 69877          |
```

总结

- 多线程并发下，InnoDB内部要做死锁检测等操作，对性能影响及其严重
- 明确的串行事务，则server层串行
- Group commit减少引擎执行次数

谢谢大家！