MySQL中间件性能测试

I

目录

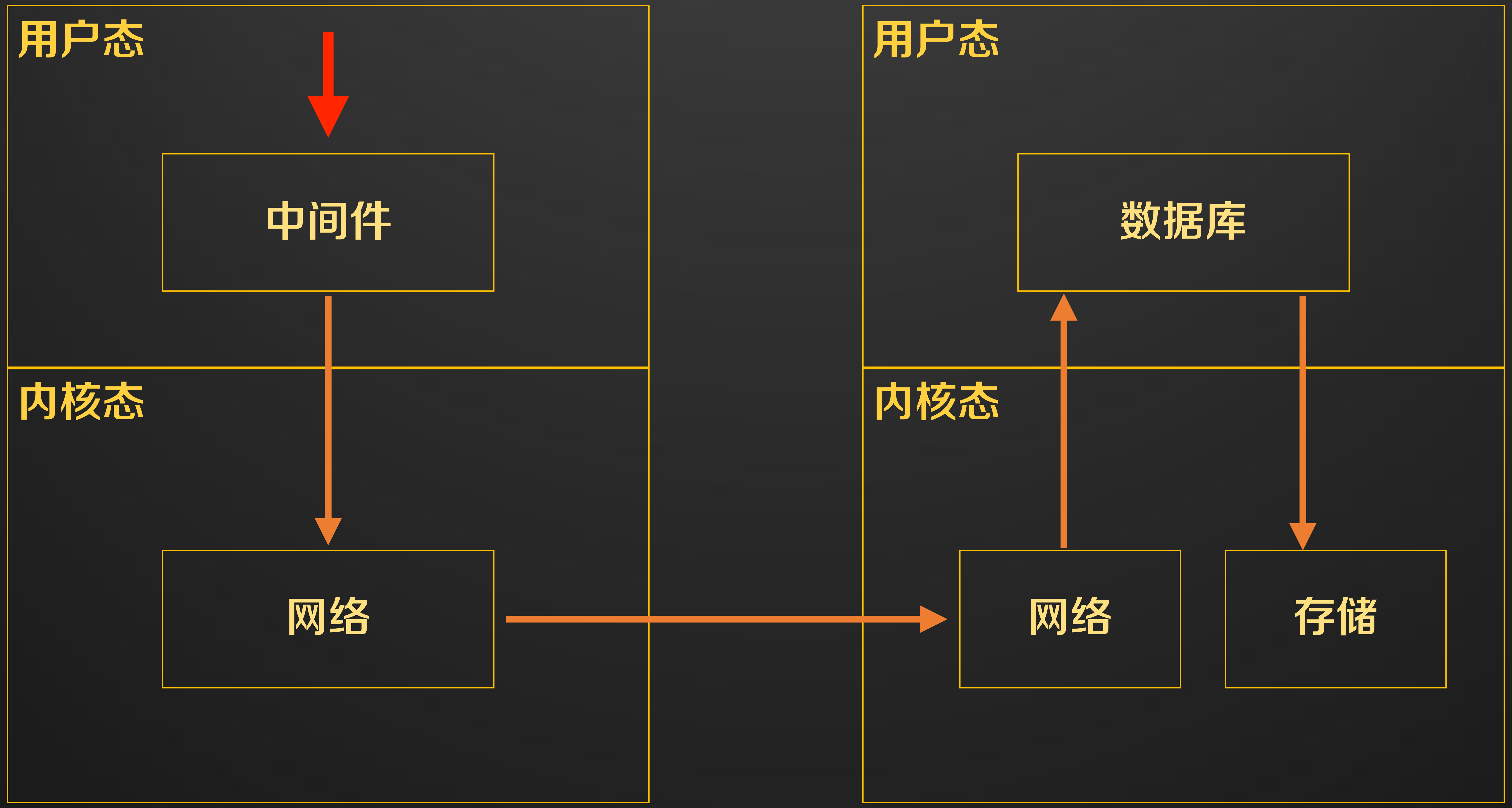
性能测试的常见(错误)方法

- 测试中间件性能的观测对象是中间件 ??
- 性能测试指标选取: 吞吐 or 延迟 ??
- 性能测试报告的结论 是要得到绝对数值 ??

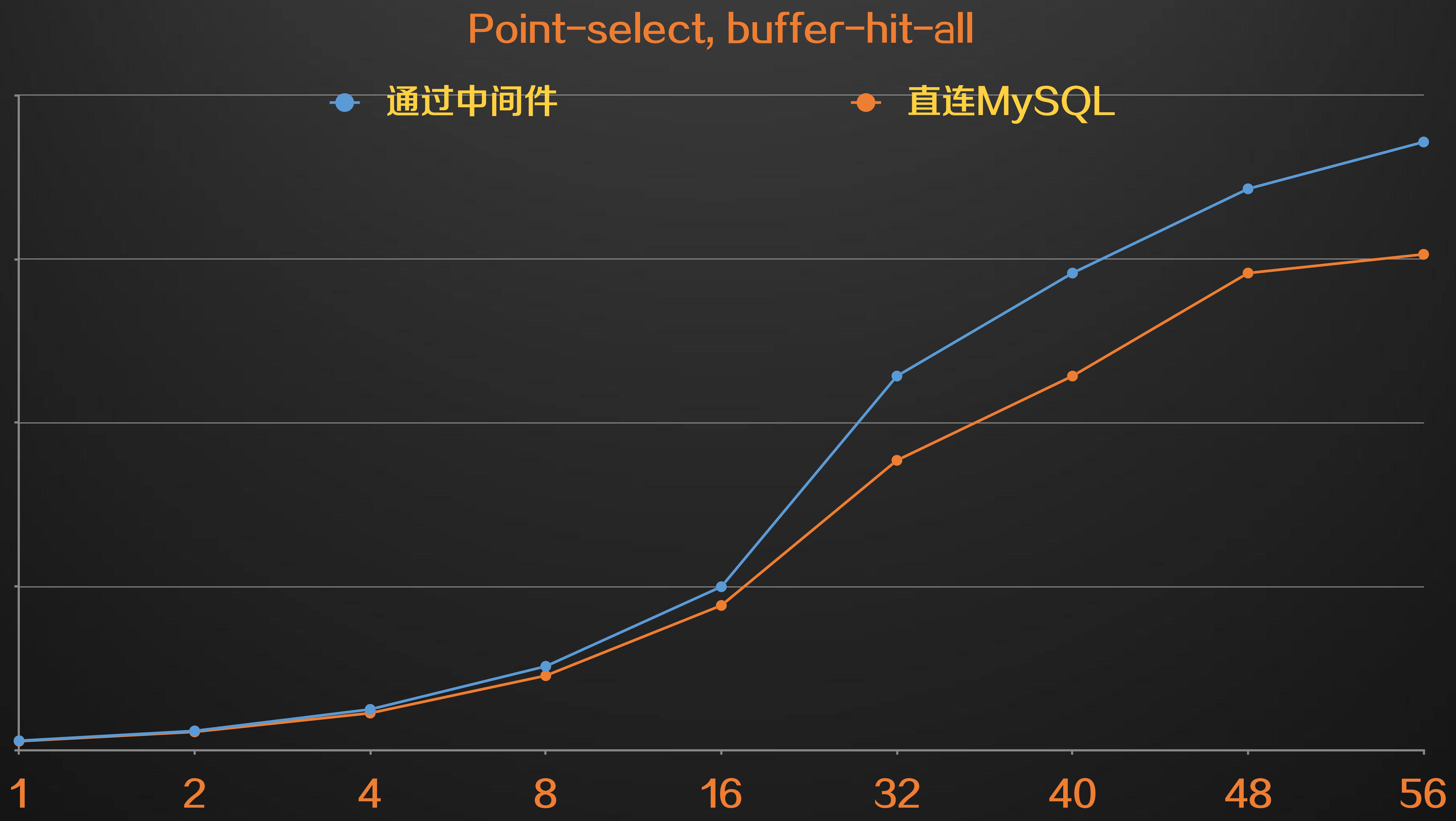
性能测试的常见(错误)方法 – 1

测试中间件性能的观测对象是中间件 ??

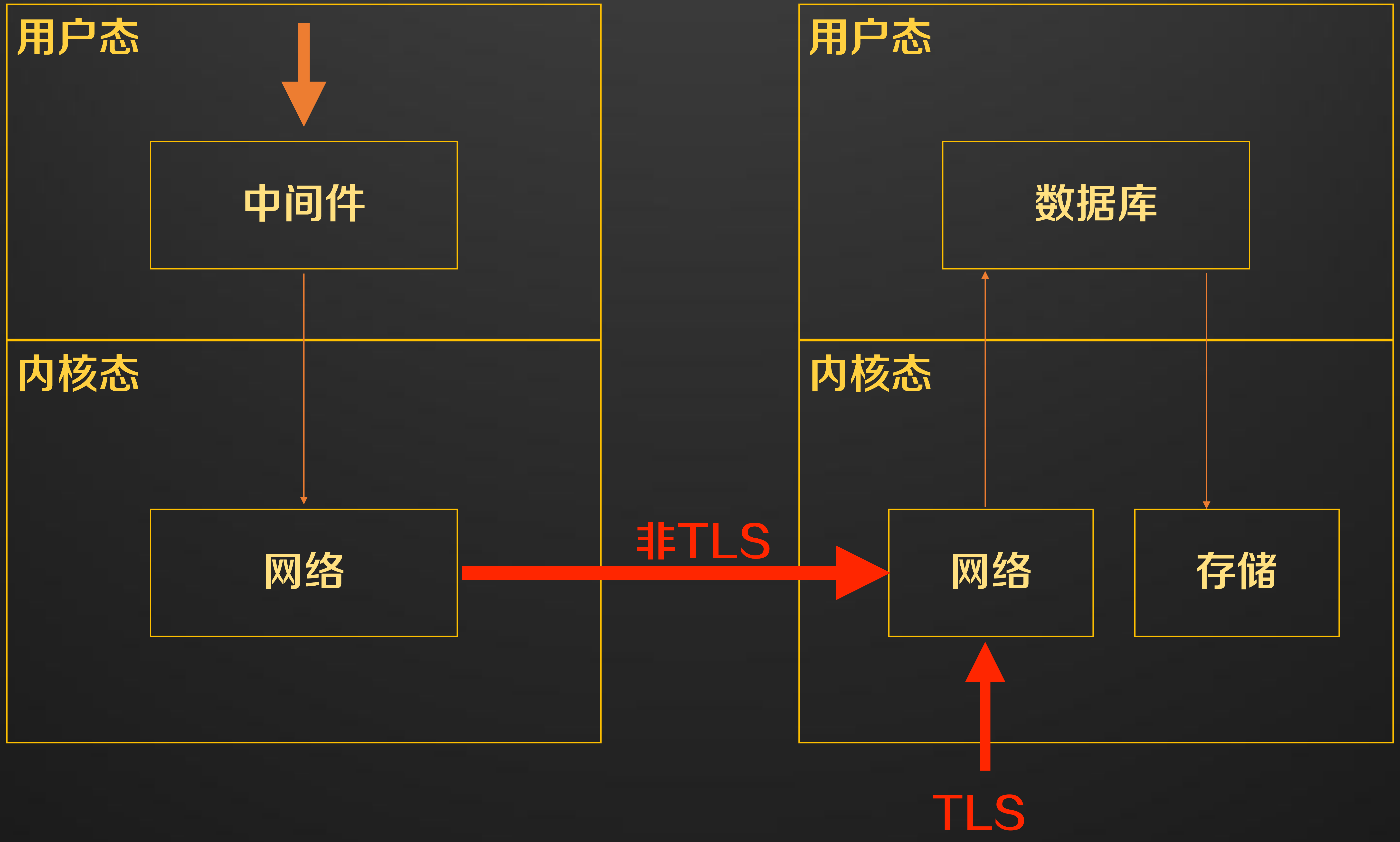
性能测试的常见(错误)方法 – 1.1



性能测试的常见(错误)方法 – 1.1



性能测试的常见(错误)方法 – 1.1



性能测试的常见(错误)方法 – 1.1

测试中间件的观察对象是

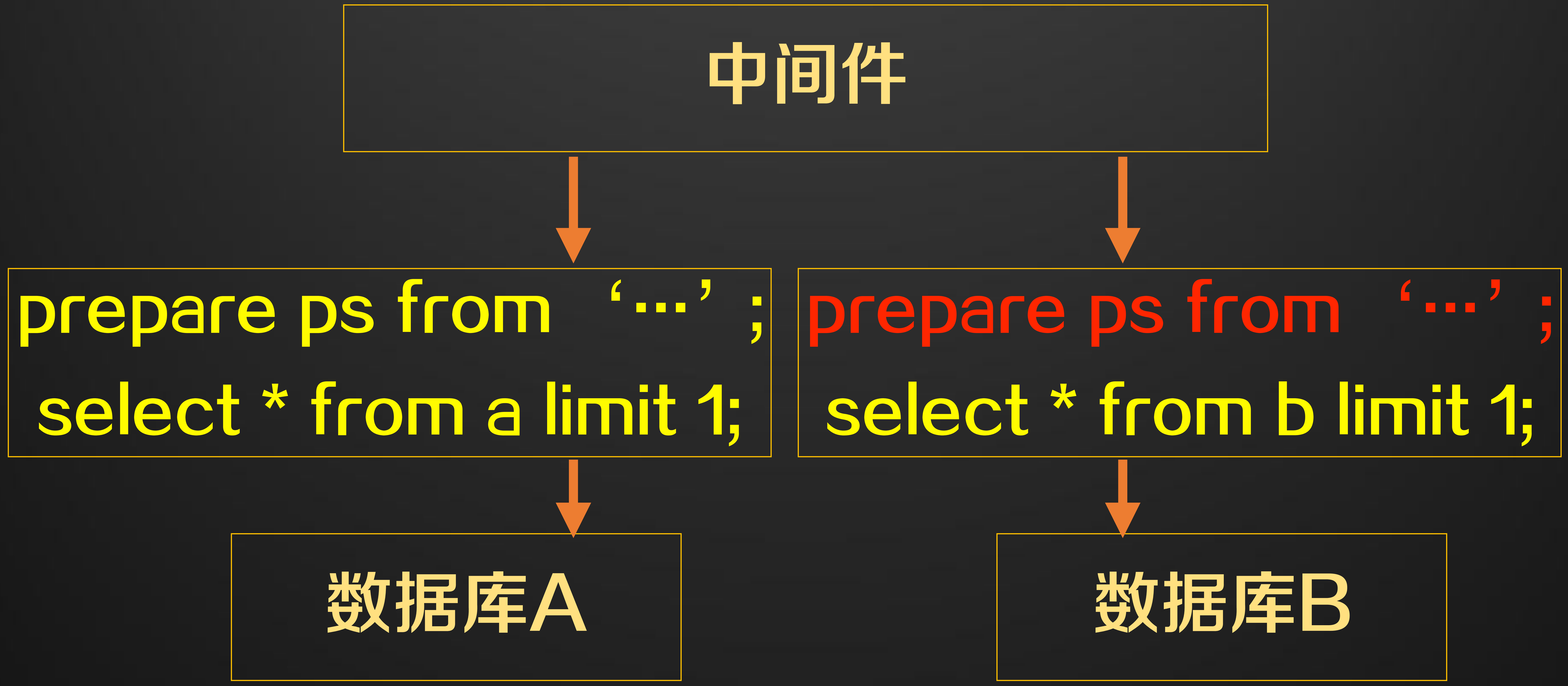
中间件+连接属性+??

性能测试的常见(错误)方法 – 1.2

考虑如下SQL:

```
prepare ps from '…' ;
select * from a limit 1;
select * from b limit 1;
```

性能测试的常见(错误)方法 – 1.2

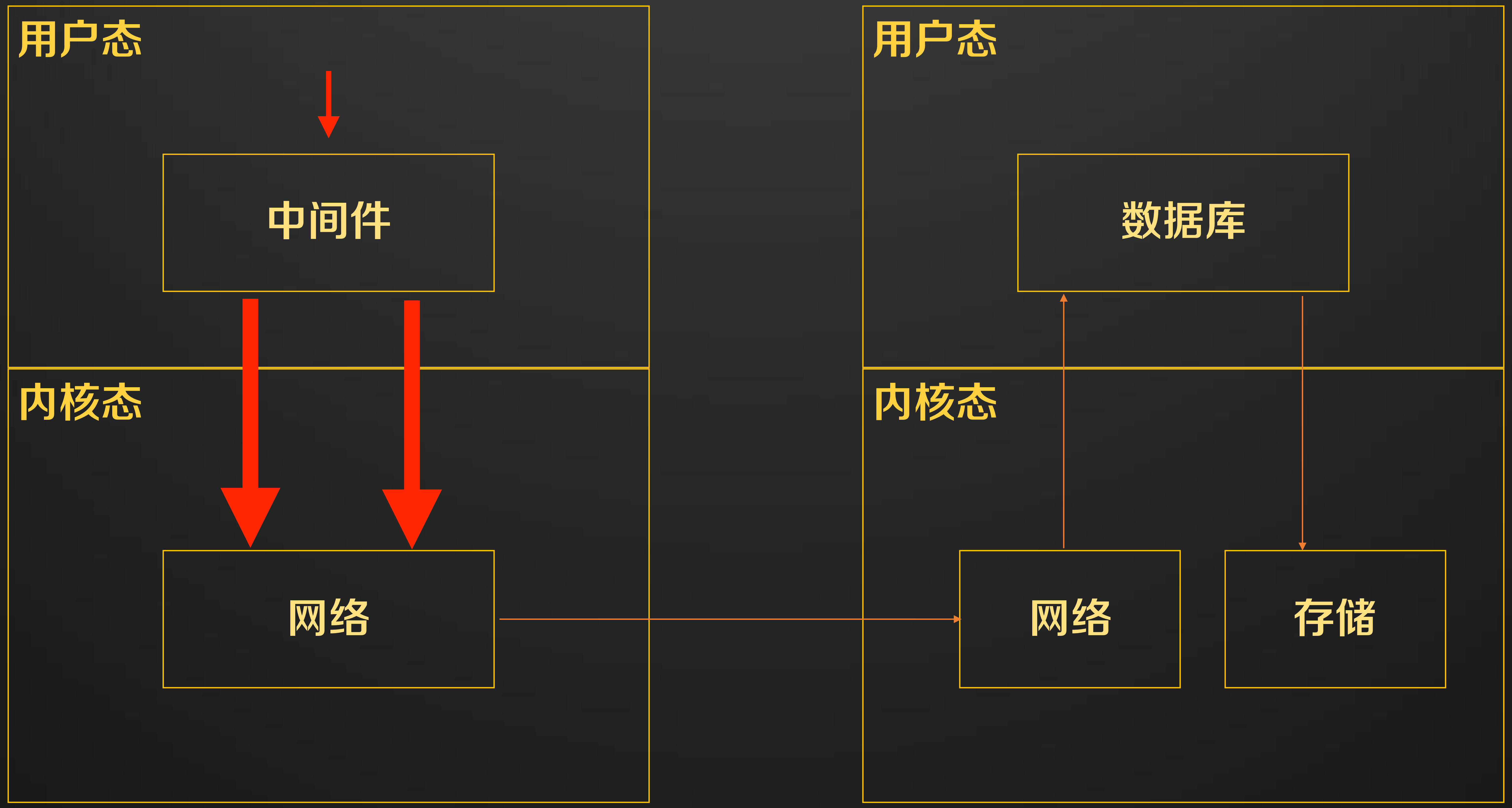


中间件

```
prepare ps from '…' ;
select * from a limit 1;
```

```
prepare ps from '…' ;
select * from b limit 1;
```

性能测试的常见(错误)方法 – 1.2



ACTION TECHNOLOGY 爱可生

性能测试的常见(错误)方法 – 1.2

中间件的上下文转移

- 事务级别 (同一事务一定发到同一节点)
- 会话级别 (上下文迁移)
  - 系统变量
  - Default schema
  - …

性能测试的常见(错误)方法 – 1.2

- 会话级别 (续)
  - Prepare Statement
  - 临时表
  - 用户变量
  - 与会话相关的特殊函数

    LAST_INSERT_ID/ROW_COUNT

性能测试的常见(错误)方法 – 1.2

测试中间件的观察对象是

中间件+连接属性+

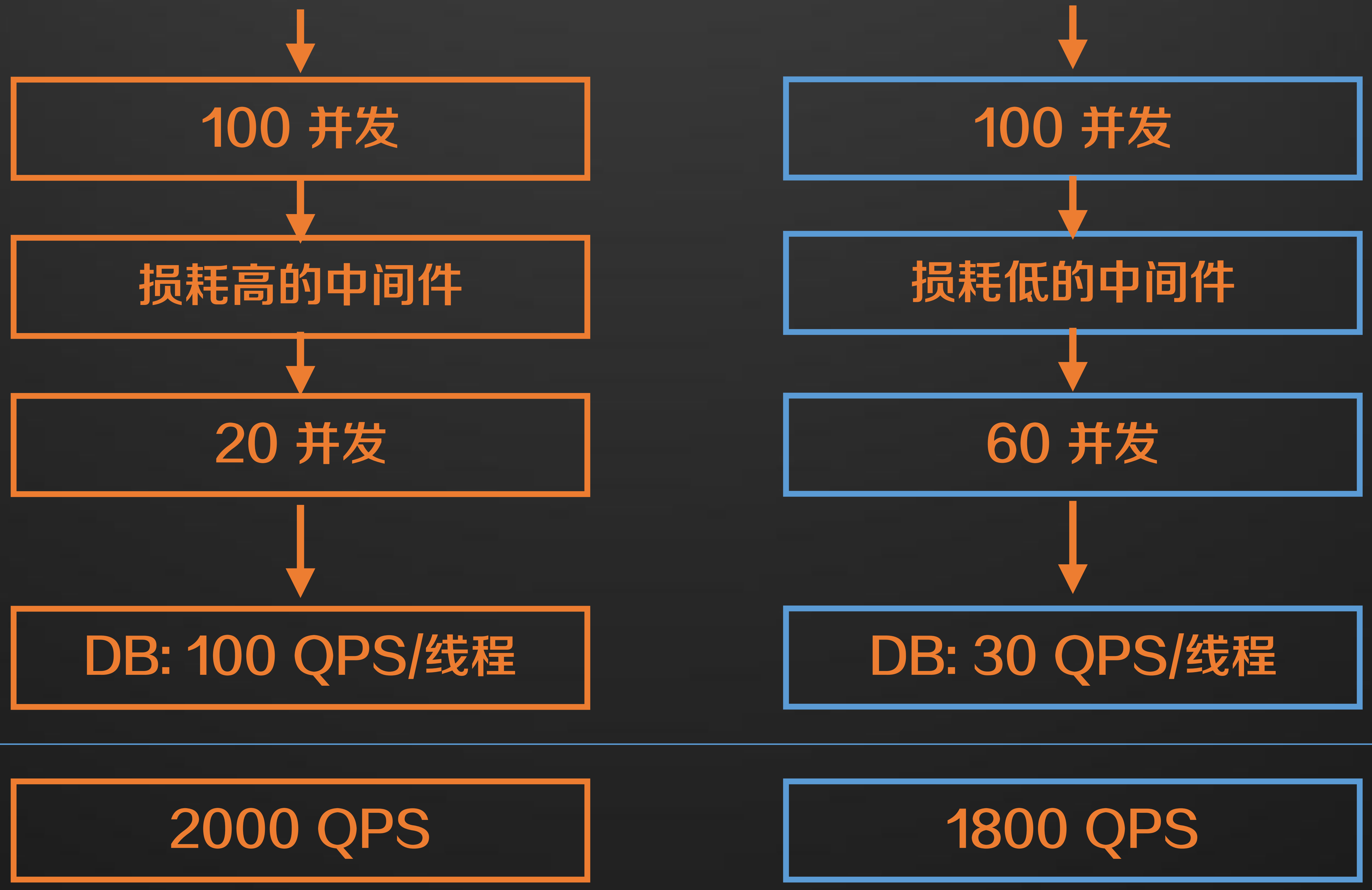
实际下发的SQL+??

性能测试的常见(错误)方法 – 1.3

同一环境下, 中间件损耗越小

是否QPS一定越高 ??

性能测试的常见(错误)方法 – 1.3



性能测试的常见(错误)方法 – 1.3

$$\text{穿透率} = \frac{\text{中间件下发的压力}}{\text{发到中间件的压力}}$$

性能测试的常见(错误)方法 – 1.3

在比较性能时,
保持对数据库的压力表现相同:
连接属性/SQL/平均延迟/…

性能测试的常见(错误)方法 – 1.3

测试中间件的观察对象是

中间件+向数据库的实际压力

TECHNOLOGY ACTION 爱可生

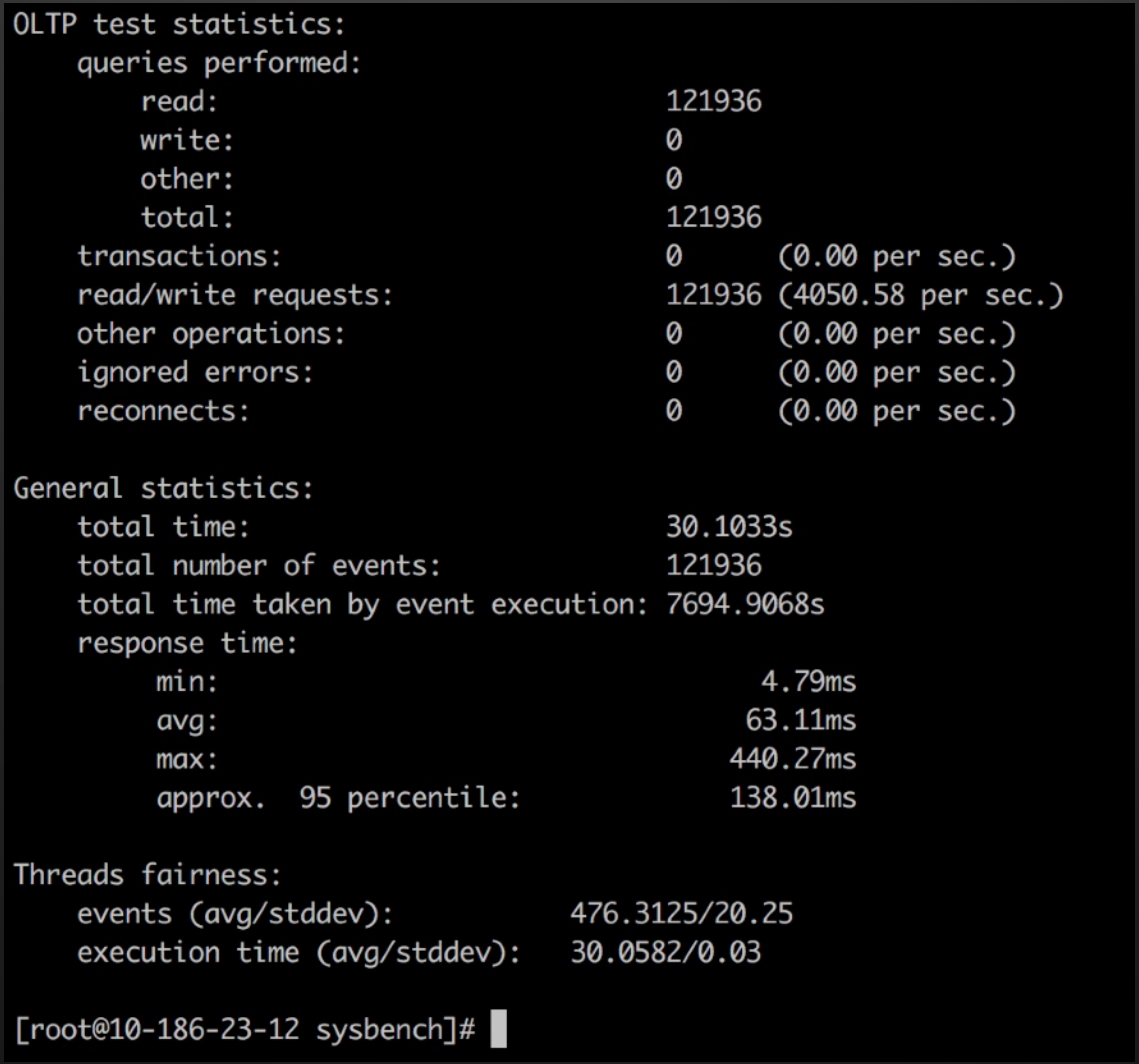
性能测试的常见(错误)方法 – 1

- 测试中间件的观察对象是

  中间件+向数据库的实际压力

- 性能测试指标选取: 吞吐 or 延迟 ??

- 性能测试报告 的结论是得到绝对数值??

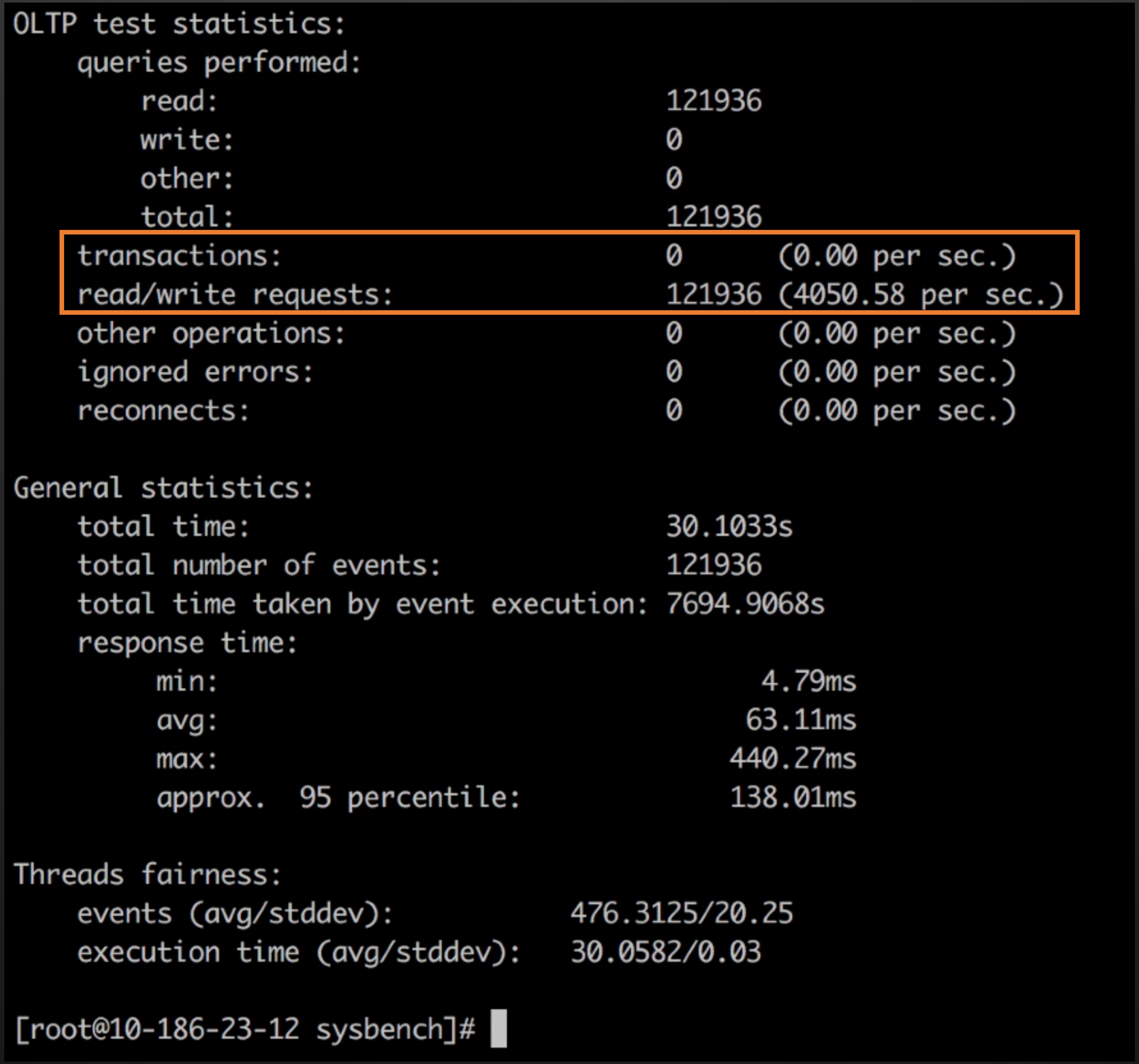


```
OLTP test statistics:
    queries performed:
        read:                            121936
        write:                           0
        other:                           0
        total:                           121936
    transactions:                        0      (0.00 per sec.)
    read/write requests:                 121936 (4050.58 per sec.)
    other operations:                    0      (0.00 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          30.1033s
    total number of events:              121936
    total time taken by event execution: 7694.9068s
    response time:
         min:                                  4.79ms
         avg:                                 63.11ms
         max:                                440.27ms
         approx.  95 percentile:             138.01ms

Threads fairness:
    events (avg/stddev):           476.3125/20.25
    execution time (avg/stddev):   30.0582/0.03

[root@10-186-23-12 sysbench]#
```

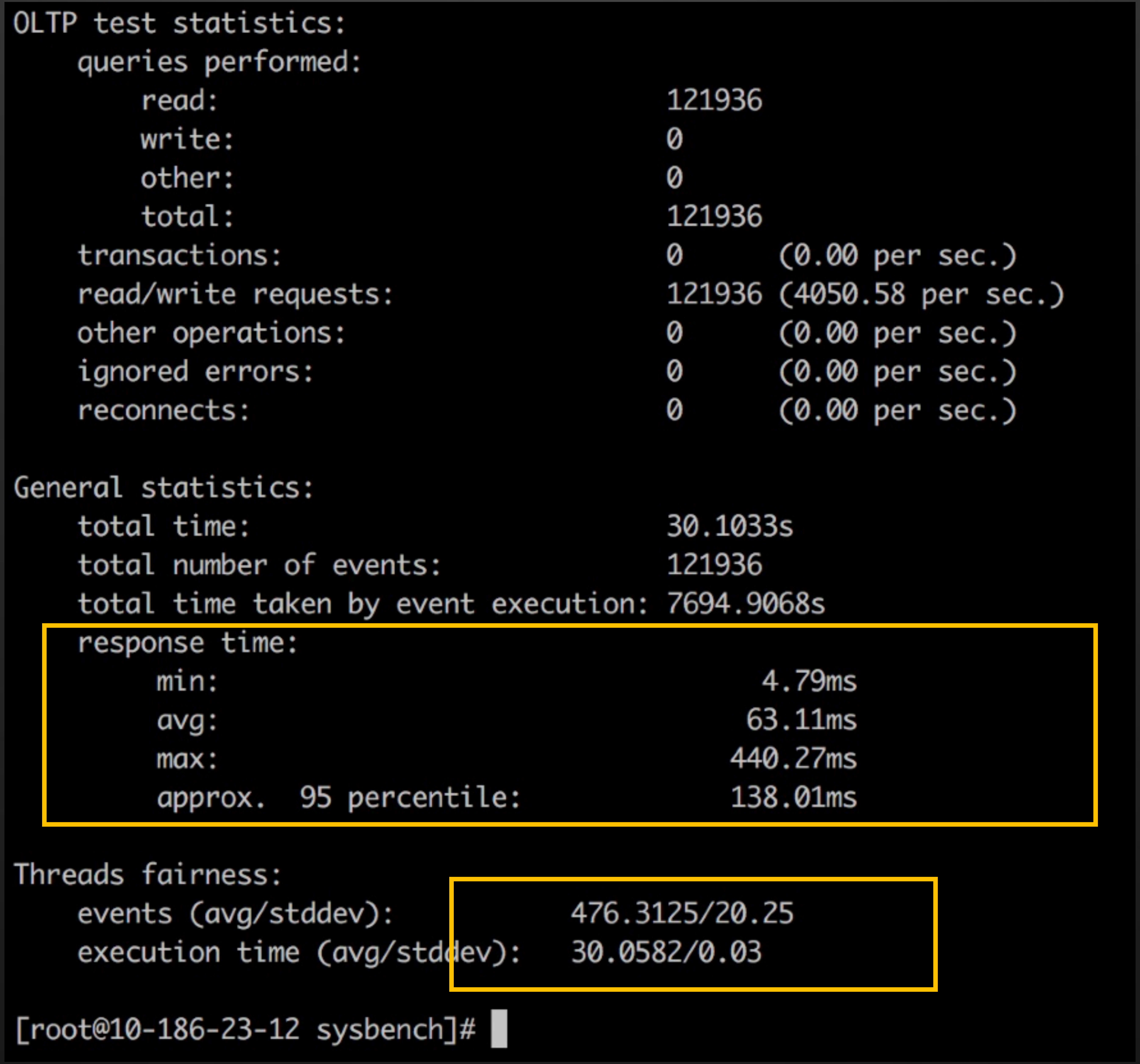


```
OLTP test statistics:
    queries performed:
        read:                            121936
        write:                           0
        other:                           0
        total:                           121936
    transactions:                        0      (0.00 per sec.)
    read/write requests:                 121936 (4050.58 per sec.)
    other operations:                    0      (0.00 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          30.1033s
    total number of events:              121936
    total time taken by event execution: 7694.9068s
    response time:
         min:                                  4.79ms
         avg:                                 63.11ms
         max:                                440.27ms
         approx.  95 percentile:             138.01ms

Threads fairness:
    events (avg/stddev):           476.3125/20.25
    execution time (avg/stddev):   30.0582/0.03

[root@10-186-23-12 sysbench]# 
```



```
OLTP test statistics:
    queries performed:
        read:                            121936
        write:                           0
        other:                           0
        total:                           121936
    transactions:                        0      (0.00 per sec.)
    read/write requests:                 121936 (4050.58 per sec.)
    other operations:                    0      (0.00 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          30.1033s
    total number of events:              121936
    total time taken by event execution: 7694.9068s
    response time:
         min:                                  4.79ms
         avg:                                 63.11ms
         max:                                440.27ms
         approx.  95 percentile:             138.01ms

Threads fairness:
    events (avg/stddev):           476.3125/20.25
    execution time (avg/stddev):   30.0582/0.03

[root@10-186-23-12 sysbench]# 
```

性能测试的常见(错误)方法 – 2

并发	吞吐	平均延迟
500 并发	4500 TPS	100 ms
100 并发	1000 TPS	20 ms
50 并发	500 TPS	5 ms

性能测试的常见(错误)方法 – 2

高压力下, 高吞吐

低压力下, 低延迟

性能测试的常见(错误)方法 – 2

- 测试中间件的观察对象是

  中间件+向数据库的实际压力

- 性能测试指标选取:

  在不同压力下, 选择不同指标

- 性能测试报告 的结论是得到绝对数值 ??

性能测试的常见(错误)方法 – 3

Sysbench OLTP_RO

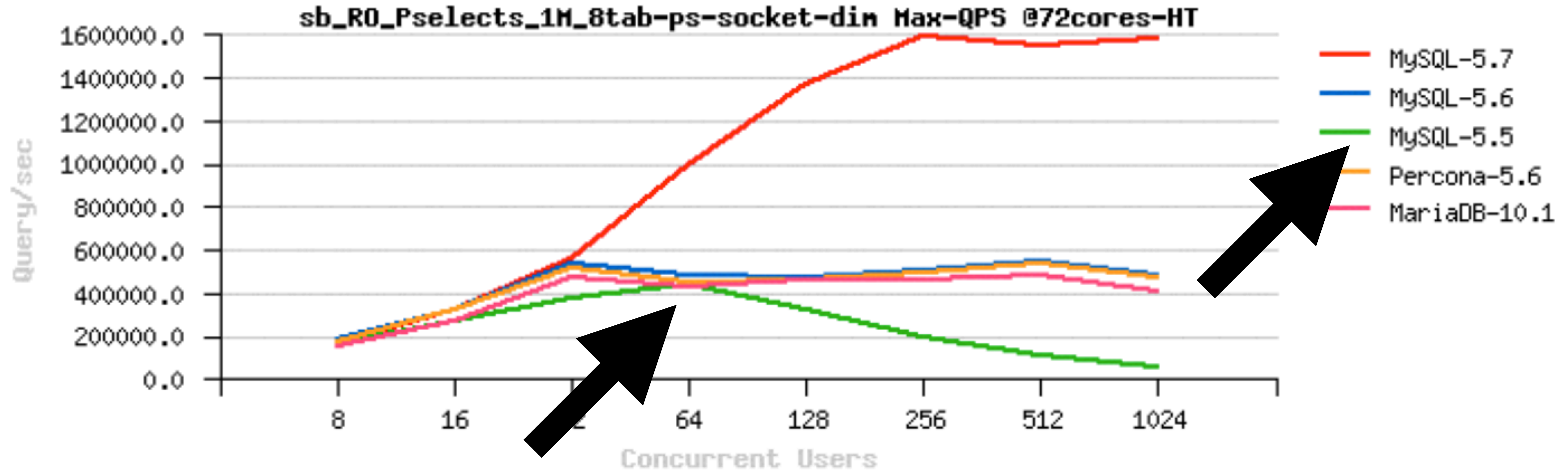
Point-Select, 1Mx8-tables

socket, 64-threads, 72core-HT

400000+ QPS

RO Point-Selects @MySQL 5.7 (Oct.2015)

- **1.6M (!!) QPS** Sysbench Point-Selects 8-tab :
  - 72cores-HT



性能测试的常见(错误)方法 – 3

MySQL 5.7 Performance:
Scalability & Benchmarks

Dimitri KRAVTCHUK
MySQL Performance Architect @Oracle

性能测试的常见(错误)方法 – 3

Numbers are just reflecting what is behind

性能测试的常见(错误)方法 – 3

实践经验:

以找到瓶颈为目的, 直到瓶颈无法解决

更容易找到 可重现的 正确的 性能值

性能测试的常见(错误)方法 – 结论

- 测试中间件的观察对象是

  中间件+向数据库的实际压力

- 性能测试指标选取:

  在不同并发下, 选择不同指标

- 性能测试报告 的结论应当是:

  同等条件下的性能比较 和 性能瓶颈分析

性能测试的一些(我们用的)方法

-观测, 观测, 观测

- -eBPF/Systemtap
- -中间件自身提供观测
- -USE

-测试工具校准

性能测试的一些(我们用的)方法 – 1

- eBPF
  (PHPCON/洪斌/MySQL性能诊断与实践)
  - MySQL Query 延迟分布
  - VFS 延迟分布
  - Ext4 延迟分布
  - 块设备 延迟分布

性能测试的一些(我们用的)方法 – 1

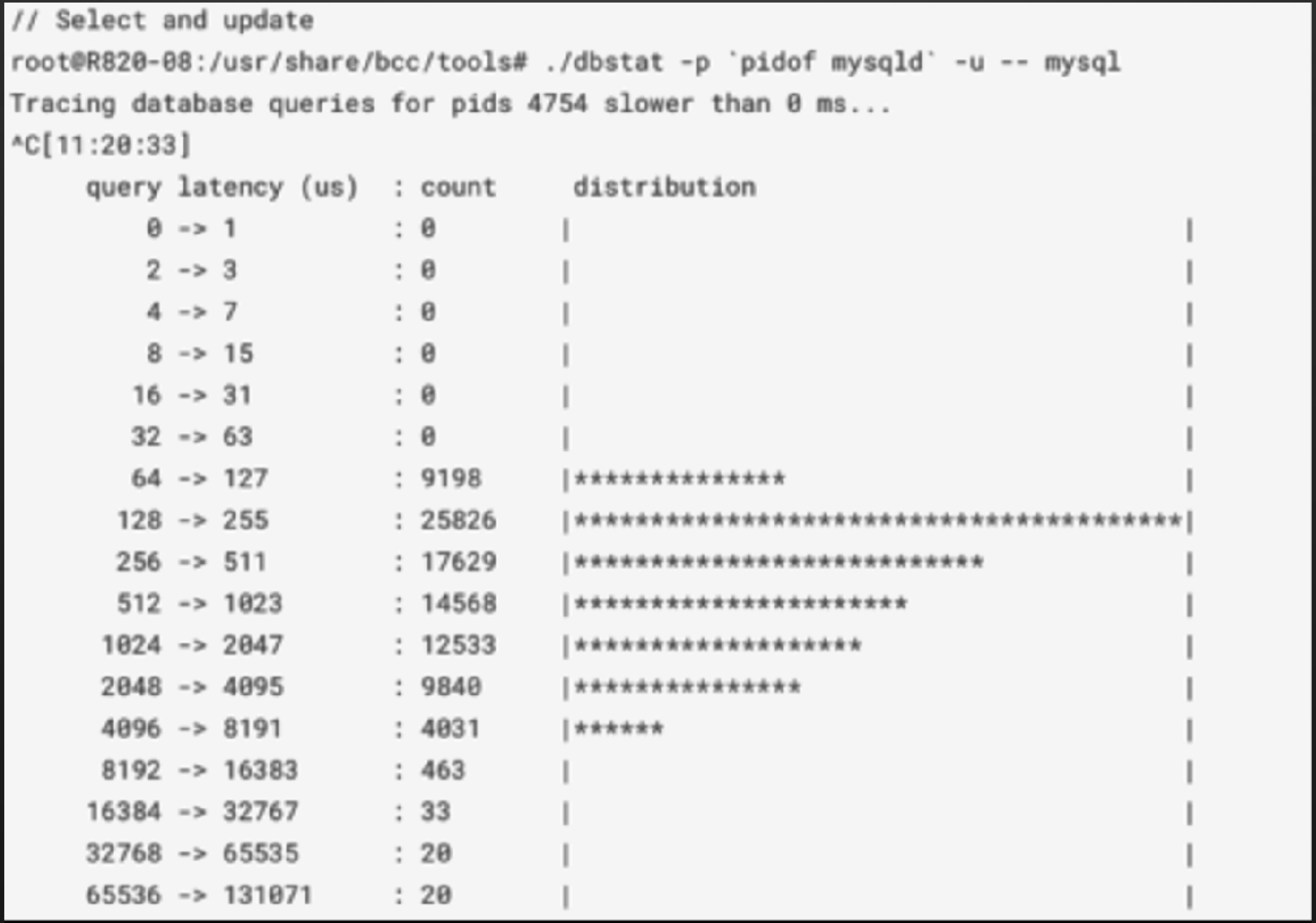
- eBPF (续)

  (PHPCON/洪斌/MySQL性能诊断与实践)

  - MySQL 文件IO压力分析
  - 临时表文件生命周期观测
  - 短连接分析
  - …

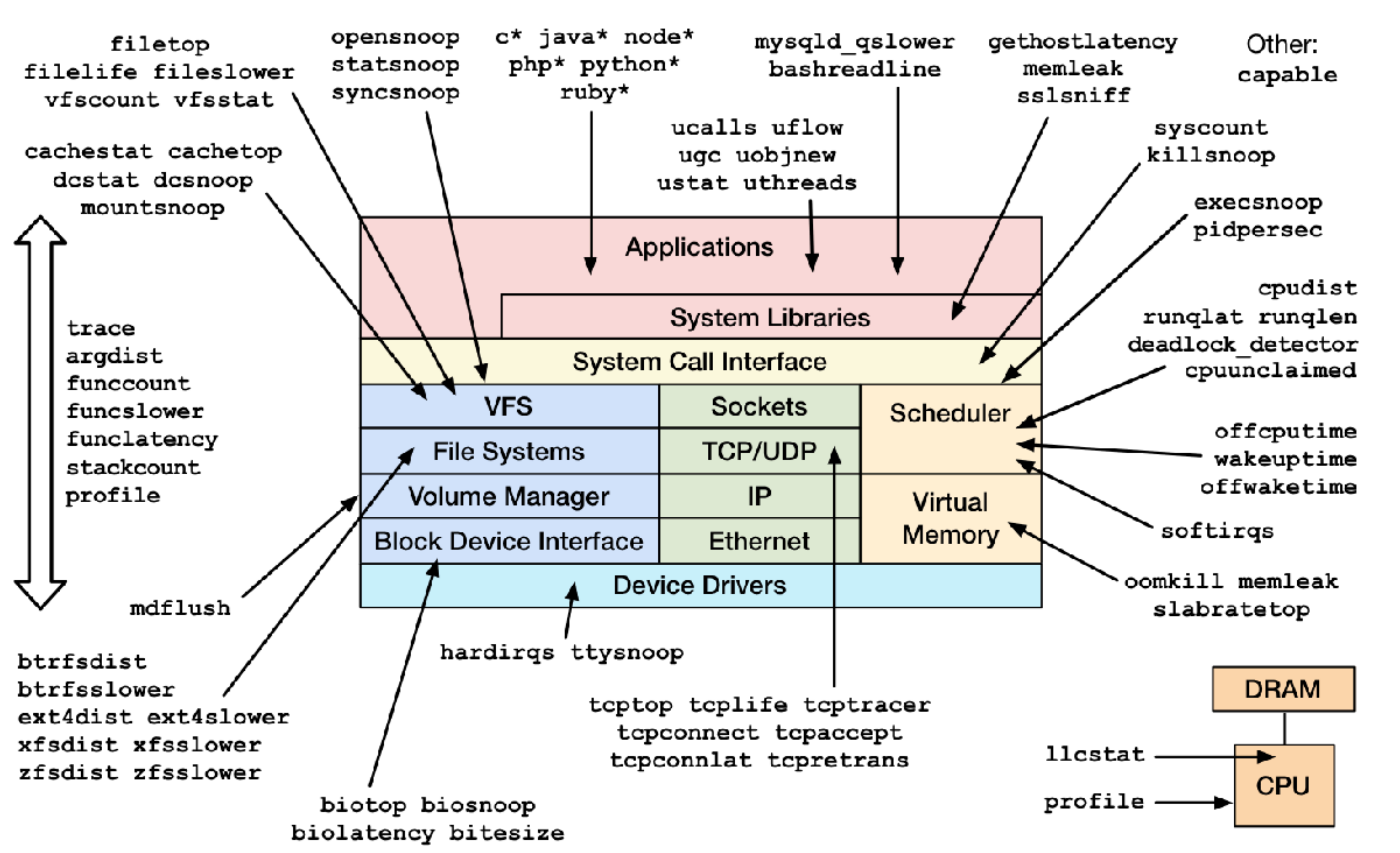
性能测试的一些(我们用的)方法 – 1

github.com/actiontech/slides



```
// Select and update
root@R820-08:/usr/share/bcc/tools# ./dbstat -p `pidof mysqld` -u -- mysql
Tracing database queries for pids 4754 slower than 0 ms...
^C[11:20:33]
     query latency (us)  : count     distribution
         0 -> 1          : 0        |                                        |
         2 -> 3          : 0        |                                        |
         4 -> 7          : 0        |                                        |
         8 -> 15         : 0        |                                        |
        16 -> 31         : 0        |                                        |
        32 -> 63         : 0        |                                        |
        64 -> 127        : 9198     |**************                          |
       128 -> 255        : 25826    |****************************************|
       256 -> 511        : 17629    |***************************             |
       512 -> 1023       : 14568    |**********************                  |
      1024 -> 2047       : 12533    |*******************                     |
      2048 -> 4095       : 9840     |***************                         |
      4096 -> 8191       : 4031     |******                                  |
      8192 -> 16383      : 463      |                                        |
     16384 -> 32767      : 33       |                                        |
     32768 -> 65535      : 20       |                                        |
     65536 -> 131071     : 20       |                                        |
```

Linux bcc/BPF Tracing Tools

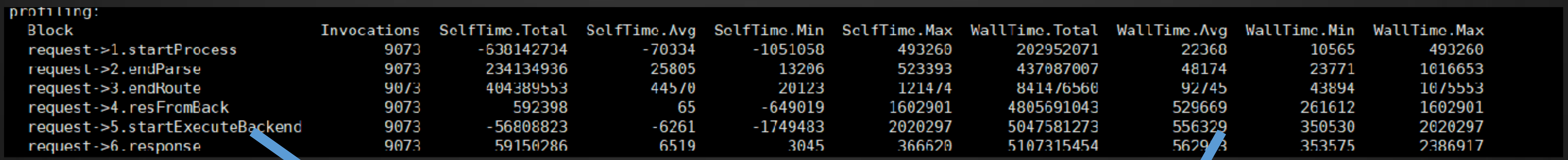


https://github.com/iovisor/bcc#tools 2017

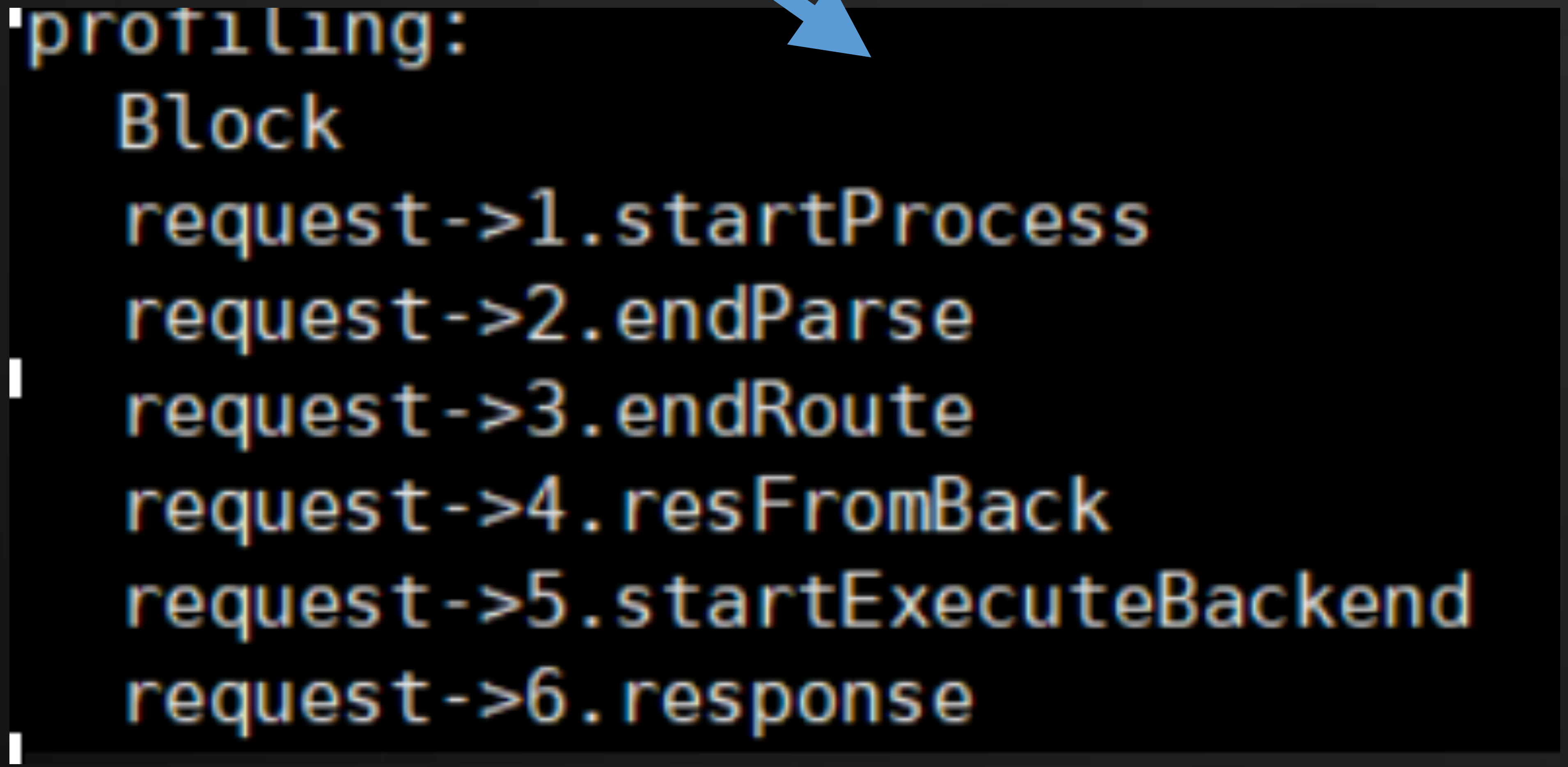
ACTION TECHNOLOGY 爱可生

性能测试的一些(我们用的)方法 – 1

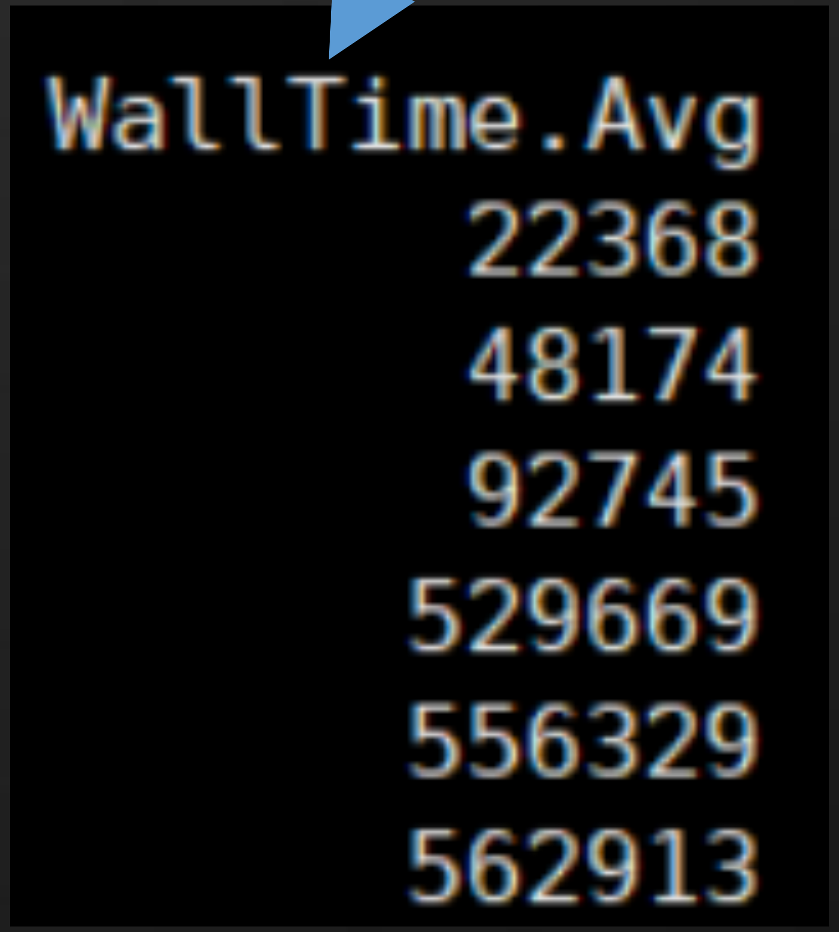
- 中间件自身提供观测 (DBLE)



```
profiling:
  Block                              Invocations  SelfTime.Total  SelfTime.Avg  SelfTime.Min  SelfTime.Max  WallTime.Total  WallTime.Avg  WallTime.Min  WallTime.Max
  request->1.startProcess                   9073      -638142734        -70334      -1051058        493260       202952071         22368         10565        493260
  request->2.endParse                       9073       234134936         25805         13206        523393       437087007         48174         23771       1016653
  request->3.endRoute                       9073       404389553         44570         20123        121474       841476560         92745         43894       1075553
  request->4.resFromBack                    9073          592398            65       -649019       1602901      4805691043        529669        261612       1602901
  request->5.startExecuteBackend            9073       -56808823         -6261      -1749483       2020297      5047581273        556329        350530       2020297
  request->6.response                       9073        59150286          6519          3045        366620      5107315454        562913        353575       2386917
```

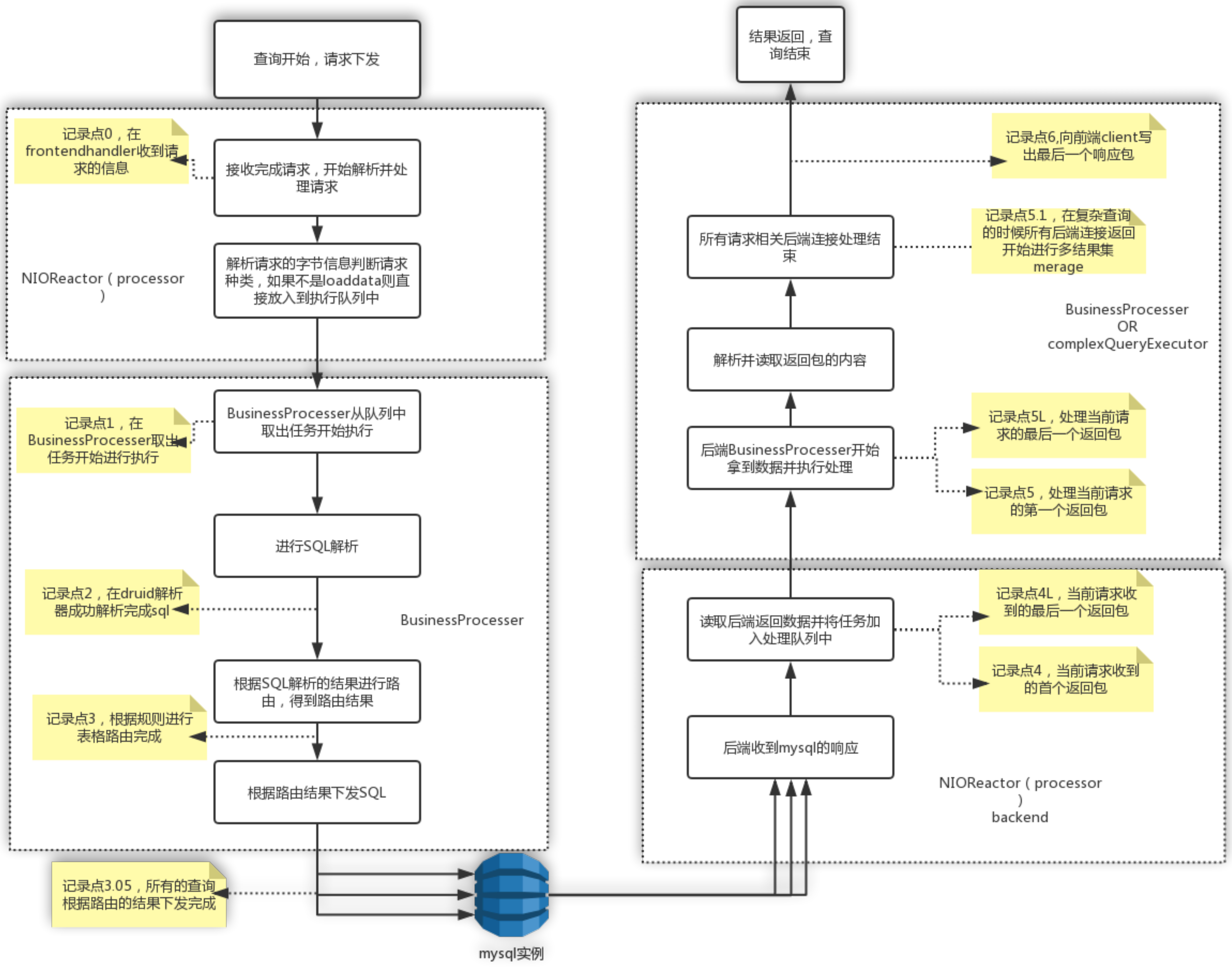


```
profiling:
  Block
  request->1.startProcess
  request->2.endParse
  request->3.endRoute
  request->4.resFromBack
  request->5.startExecuteBackend
  request->6.response
```



```
WallTime.Avg
       22368
       48174
       92745
      529669
      556329
      562913
```

ACTION 爱可生 TECHNOLOGY


查询开始，请求下发
记录点0，在frontendhandler收到请求的信息
接收完成请求，开始解析并处理请求
NIOReactor（processor）
解析请求的字节信息判断请求种类，如果不是loaddata则直接放入到执行队列中
记录点1，在BusinessProcesser取出任务开始进行执行
BusinessProcesser从队列中取出任务开始执行
进行SQL解析
记录点2，在druid解析器成功解析完成sql
BusinessProcesser
根据SQL解析的结果进行路由，得到路由结果
记录点3，根据规则进行表格路由完成
根据路由结果下发SQL
记录点3.05，所有的查询根据路由的结果下发完成
mysql实例
结果返回，查询结束
记录点6,向前端client写出最后一个响应包
所有请求相关后端连接处理结束
记录点5.1，在复杂查询的时候所有后端连接返回开始进行多结果集merage
BusinessProcesser OR complexQueryExecutor
解析并读取返回包的内容
后端BusinessProcesser开始拿到数据并执行处理
记录点5L，处理当前请求的最后一个返回包
记录点5，处理当前请求的第一个返回包
读取后端返回数据并将任务加入处理队列中
记录点4L，当前请求收到的最后一个返回包
记录点4，当前请求收到的首个返回包
后端收到mysql的响应
NIOReactor（processor）backend

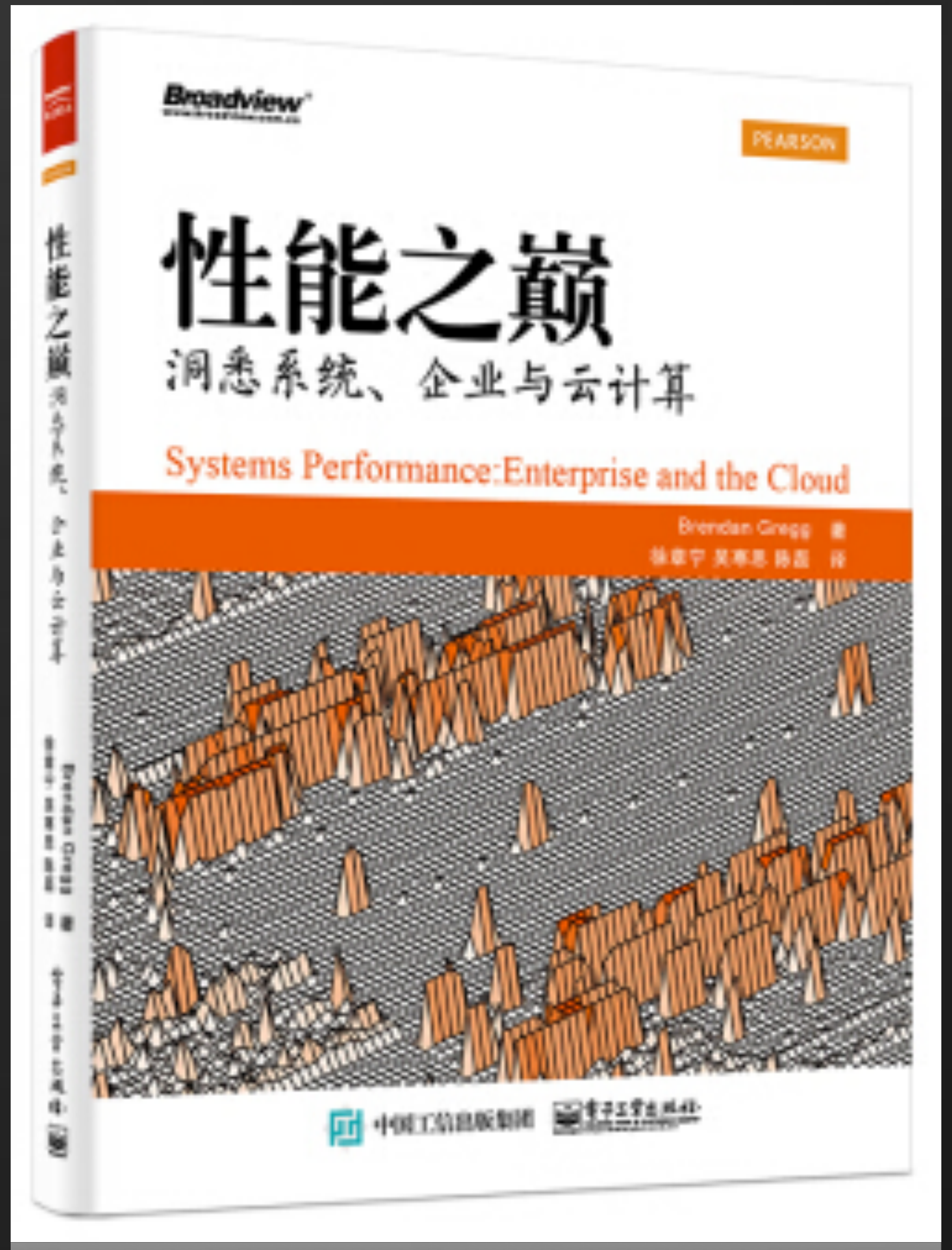
性能测试的一些(我们用的)方法 – 1

- dble.cloud

ACTION 爱可生

性能测试的一些(我们用的)方法 – 1

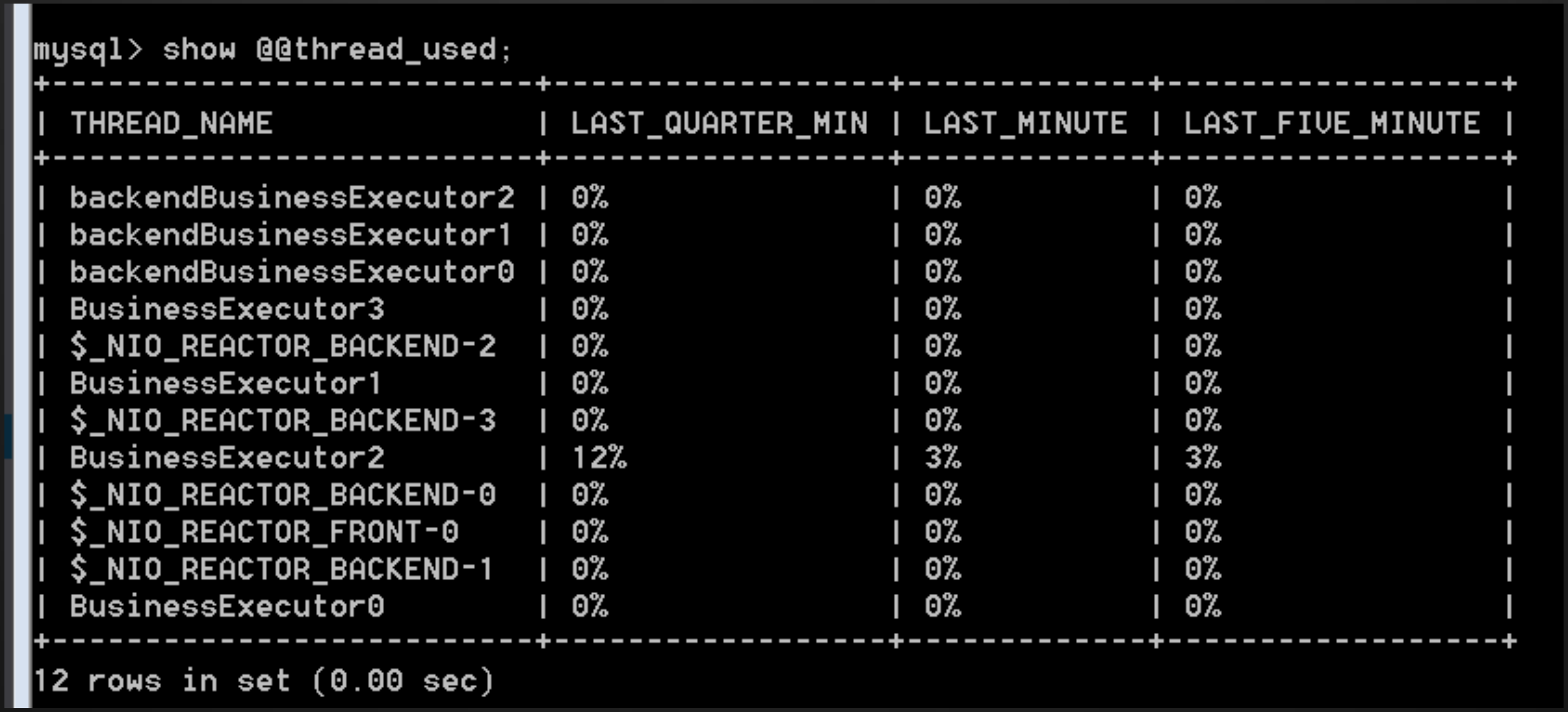
- USE
  - Usability
  - Saturation
  - Error



ACTION 爱可生

性能测试的一些(我们用的)方法 – 1

-USE (DBLE)



```
mysql> show @@thread_used;
+---------------------------+-------------------+---------------+-------------------+
| THREAD_NAME               | LAST_QUARTER_MIN  | LAST_MINUTE   | LAST_FIVE_MINUTE  |
+---------------------------+-------------------+---------------+-------------------+
| backendBusinessExecutor2  | 0%                | 0%            | 0%                |
| backendBusinessExecutor1  | 0%                | 0%            | 0%                |
| backendBusinessExecutor0  | 0%                | 0%            | 0%                |
| BusinessExecutor3         | 0%                | 0%            | 0%                |
| $_NIO_REACTOR_BACKEND-2   | 0%                | 0%            | 0%                |
| BusinessExecutor1         | 0%                | 0%            | 0%                |
| $_NIO_REACTOR_BACKEND-3   | 0%                | 0%            | 0%                |
| BusinessExecutor2         | 12%               | 3%            | 3%                |
| $_NIO_REACTOR_BACKEND-0   | 0%                | 0%            | 0%                |
| $_NIO_REACTOR_FRONT-0     | 0%                | 0%            | 0%                |
| $_NIO_REACTOR_BACKEND-1   | 0%                | 0%            | 0%                |
| BusinessExecutor0         | 0%                | 0%            | 0%                |
+---------------------------+-------------------+---------------+-------------------+
12 rows in set (0.00 sec)
```

性能测试的一些(我们用的)方法 – 2

BenchmarkSQL

– for (!deleted) {delete}

– 如下压力顺序, 在RR隔离级别下, 会导致死循环:

```
1> set auto_commit = 0
1> select * from A where id = 1
                                        2> delete from A where id = 1
1> delete from A where id = 1
1< Query OK, 0 rows affected
```

分布式事务相关

- “如何证明分布式事务的实现是有效的”
- “你可以随便拔电源100次”

分布式事务相关

- 事务性
  - ACID相关的数据异常
    - 脏读/不可重复读/幻读/脏写/更新丢失/写偏序/读偏序/…
  - 针对锁机制的弱点: S2PL/SS2PL
  - …

分布式事务相关

- 可靠性和性能
  - CPU
  - 内存 (perf – NUMA)
  - 磁盘 (systemtap – 延迟/错误)
  - 网络 (tc – 延迟/乱序/篡改/丢失)
  - 进程 (kill / hang / 线程乱序执行)
  - …

分布式事务相关

怀着敬畏之心

怀疑每一行代码都会

– 出错

或者

– 不返回结果

TECHNOLOGY
ACTION
爱可生