MySQL8.0基准性能测试参考

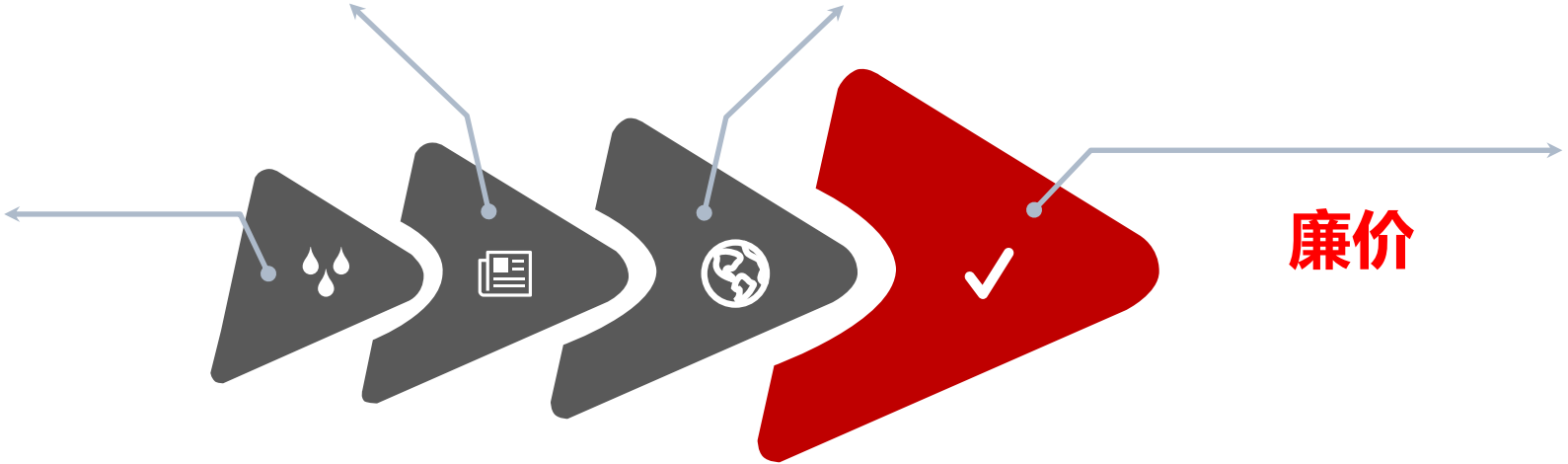
京东商城-王伟

JD.COM 京东

稳定

高效

安全



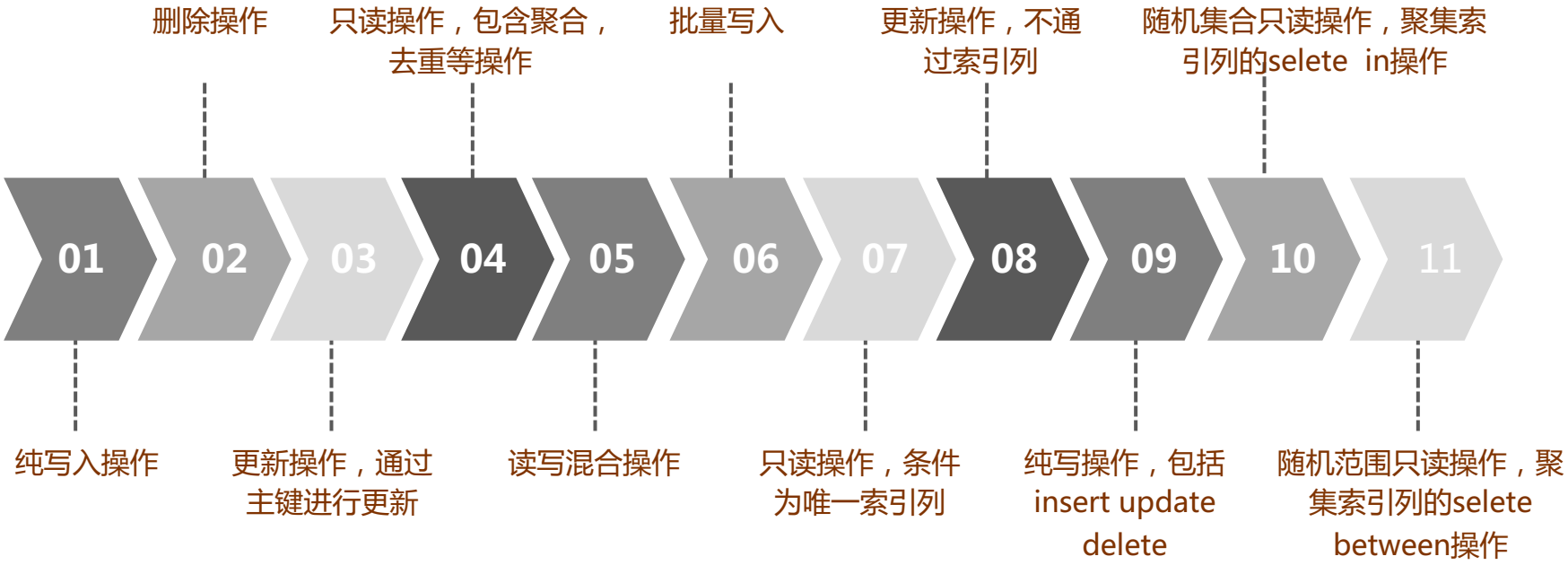
廉价

测试注意事项

测试注意事项

- 1、物理环境相同，服务器硬件参数设置和OS内核参数优化相同，OS版本相同。
- 2、sysbench版本统一，数据库的参数统一，数据需要一致，测试数据完整性检查
- 3、每次的数据测试需要在不同的版本上校验，确定数据是一致的。
- 4、涉及到数据写操作的测试，每个threads重新使用初始化数据，并重启数据库，清理缓存；记录测试前后的数据物理文件大小和测试后的数据条数。
- 5、只读操作可以使用初始化数据完成多个threads的测试,每个threads轮转前直接重启数据库，清理缓存。
- 6、尽可能详尽的监控。
- 7、测试场景要全，覆盖线上业务多种场景。

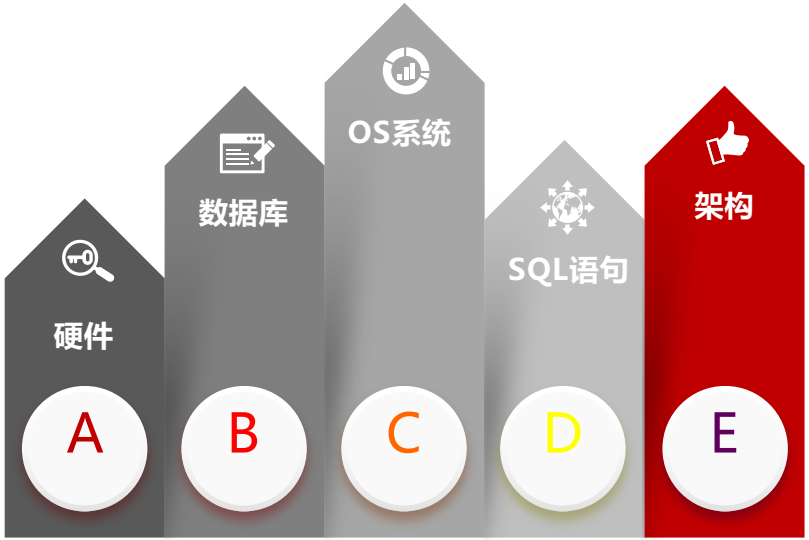
采用sysbench进行MySQL的全场景测试，包含如下11个场景



测试关注的指标

1. 每秒处理的事务平均个数(TPS)
2. 每秒操作数据库的平均个数(QPS)
3. 数据库平均响应时间Latency(ms)
4. 每秒钟接收的数据包(rxpck/s)
5. 用户使用CPU的百分比(%user)
6. 硬盘IO的使用率（util%)

数据库优化



- 硬件设备参数优化
- 数据库参数的优化
- OS系统参数优化
- SQL语句的优化
- 架构的优化

OS系统参数的优化

- **优化操作：**
  - 1、推荐使用Centos7
  - 2、设置系统打开文件数ulimit
  - 3、使用deadline/noop这两种I/O调度器,推荐使用noop
  - 4、推荐使用xfs文件系统
  - 5、增加socket缓存区的内存大小
  - 6、使用socket快速回收，特别是短连接较多的Server
  - 7、编译安装MySQL

架构的优化

- 1、业务拆分：搜索业务，像like %xxxxx%，建议使用ES等数据库产品
- 2、数据库前端必须加缓存，例如：redis，用户登录，商品查询，订单查询等
- 3、某些业务应用使用nosql持久化缓存，例如：redis，memcache等，例如：用户、商品信息等
- 4、动态的数据静态化，整个文件静态化，页面片段静态化，避免大字段。比如：用户发布一个商品，商品生成html，推到前端(大网站必须要做的)
- 5、数据库集群与读写分离：一主多从，双主多从，通过程序或dbproxy进行集群读写分离。
- 6、超级大表，拆库拆表。
- 7、组合使用不同的使用数据库

• SQL语句的优化

- 1、索引的优化
- 2、慢查询分析
- 3、不使用大字段
- 4、禁止使用外键、存储过程、视图、触发器等
- 5、规范的命名规则
- 6、使用简单的查询避免复杂查询
- 7、其他等等

硬件设备参数优化

01 选择PerformancePerWattOptimized(DAPC)

02 MemoryFrequency（内存频率）选择最佳性能

03 关闭NUMA

04 使用高速SSD设备

05 购置阵列卡同时配备CACHE及BBU模块，RAID卡使用RAID10，阵列写策略为WB

06 使用多网卡做Bond

InnoDB事务调度算法

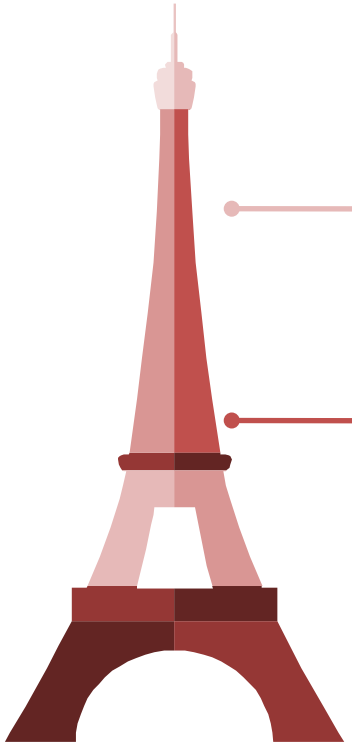
8.0.3开始默认调度算法

CATS机制：能够感知到事务竞争关系来实现全局最小开销的锁调度算法，当一个事务已经持有了多个对象的锁，当该事务请求一个新的锁的时候，该事务应该被优先分配。从另一个方面讲，解锁这样的事务有助于解锁更多的事务。因为该事务优先被分配锁能更快的结束事务，释放另外已经获取到的对象的锁。通过这种方式可以使数据库获得更高的吞吐和更低的延迟。

MySQL老的默认事务调度算法：

FIFO机制：将锁分配给最先请求该锁的事务（即该事务在等待队列的最前面，除非它们与当前锁赋予的锁不兼容）。

CATS算法带来的影响



复杂场景

MySQL是全球第一个使用这种最先进的CATS事务调度算法的数据库，这个算法解决了数据库在遇到高压力情况下性能急剧下降的问题，CATS算法是针对当事务并发超过32的情况，这个数值没有参数配置。

性能提升

Oracle的Dimitri Kravtchuk通过Sysbench 的OLTP脚本测试这种新的算法。通过结果显示，在并发情况下，CATS算法比FIFO算法在TPS，平均延迟，95%延迟等指标方面都有显著的性能提升。值得注意的是，即使在没有并发的情况下，CATS算法的性能和FIFO算法性能是一样的。那是因为在没有并发的时候，没有事务需要进行调度，所以也就没有性能的差异。换而言之，使用CATS算法替换FIFO算法，没有任何损失，反而在数据库繁忙的时候，有很大的性能提升。

简单来说：新的算法在复杂的业务场景中带来很大的性能提升。

线上业务测试表现

数据库版本	并发用户数	TPS	最大TPS	最大响应时间	最小响应时间	平均响应时间	TP50	TP90	TP99	TP999
5.7	200	8385	9048	1000	0	24	23	30	39	71
	500	6480	10790	1081	0	77	61	132	292	529
8.0	200	8688	9238	1117	0	23	22	27	38	63
	500	10809	11788	984	0	46	43	64	94	148

线上业务测试表现

- 5.6版本

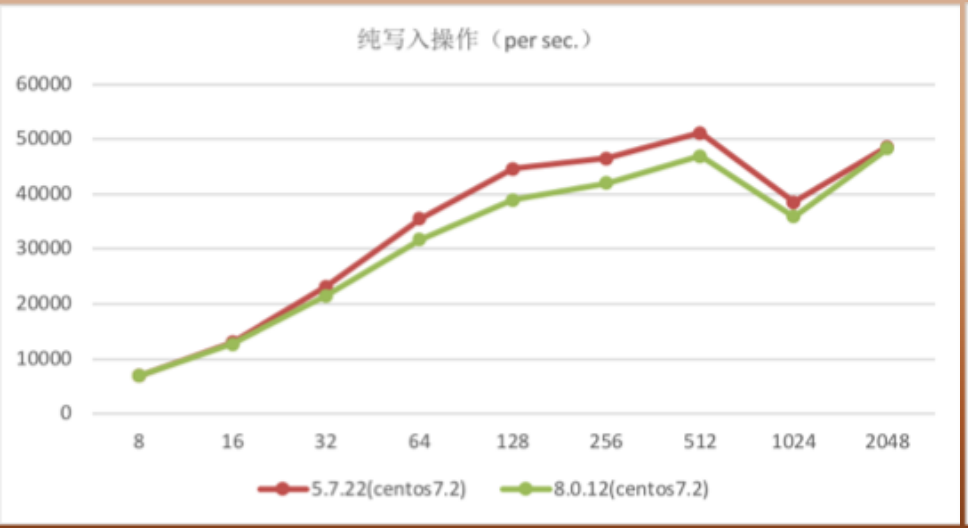
- 5.7版本

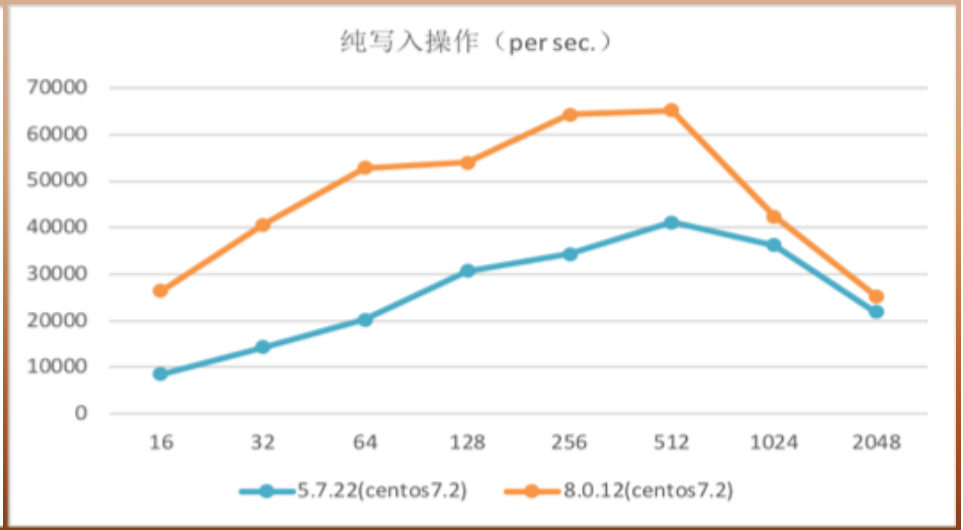
- 8.0版本

CPU:32 内存：128 硬盘：STAT-SSD 8*800G

log_bin
innodb_thread_concurrency = 0
innodb_log_file_size = 1G
innodb_log_files_in_group = 16
innodb_log_buffer_size = 64M
thread_cache_size = 2048
innodb_buffer_pool_size = 80G
sync_binlog = 1
innodb_flush_log_at_trx_commit = 1
innodb_buffer_pool_instances =16
innodb_write_io_threads = 16
innodb_read_io_threads = 16
innodb_io_capacity = 3000-->10000
innodb_io_capacity_max = 16000 -->20000

参数优化-Insert对比

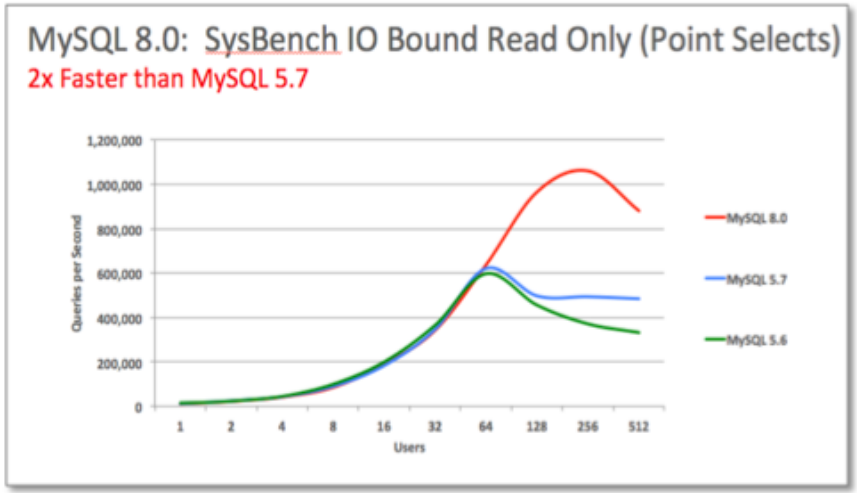


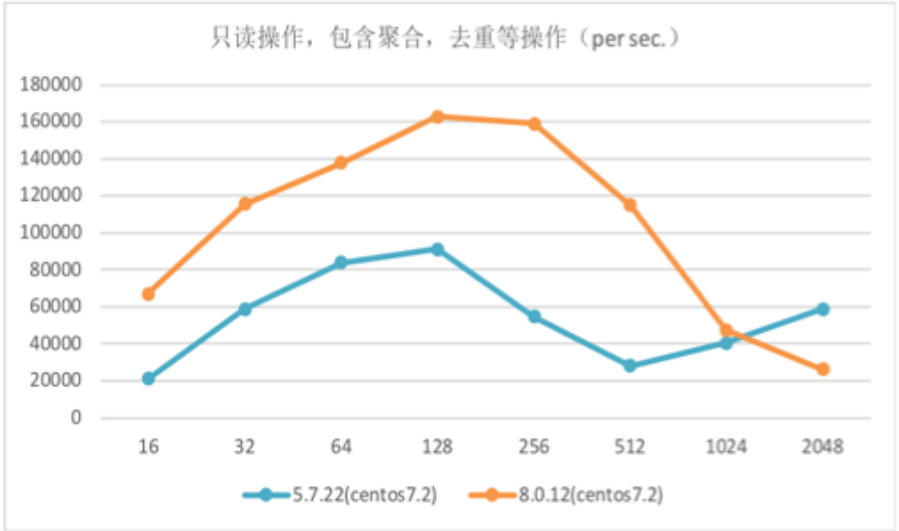


纯写入insert场景，经过参数优化后效果还是比较明显。

Read-only场景

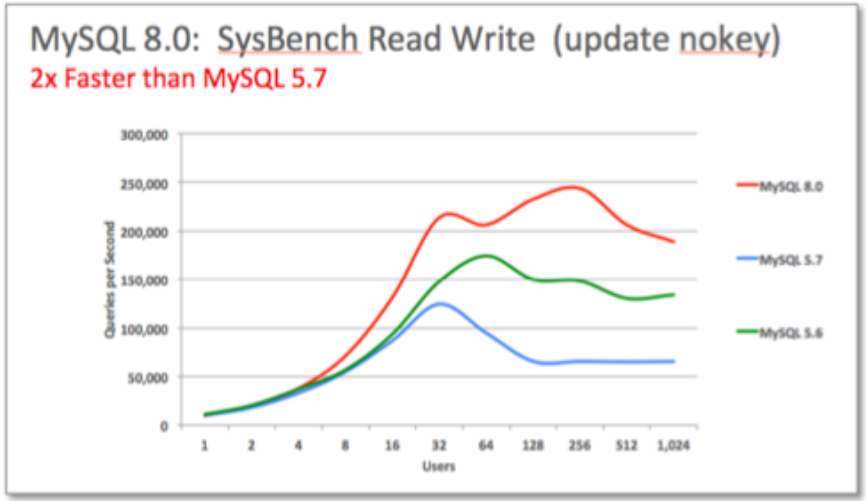
2x Faster than MySQL 5.7: SysBench Read Only (Point Selects)

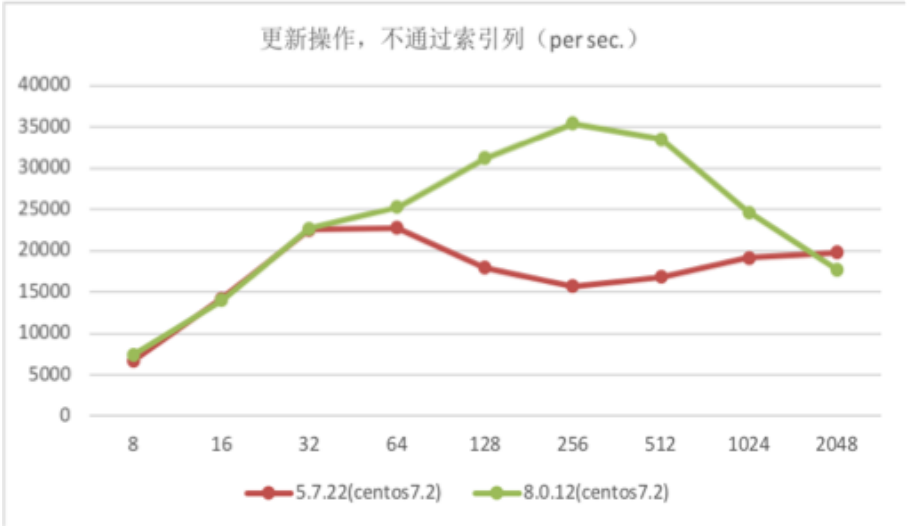




Update nokey场景

2x Faster than MySQL 5.7: SysBench Read Write

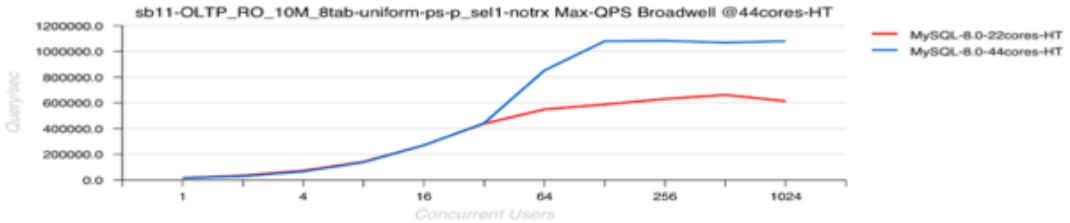




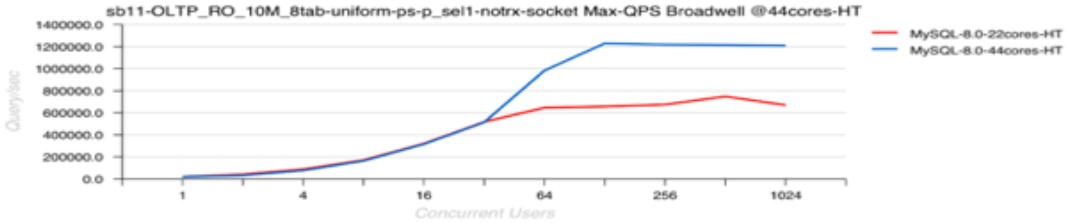
Dimitrik多核对比测试

Broadwell 2S 44cores-HT

- IP port :

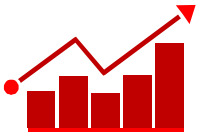


- UNIX socket :



Comments :

- wow, up to 18% difference !
- and you can clearly see MySQL 8.0 out-passing 1.2M QPS with UNIX socket, and staying under 1.1M QPS with IP port..



- 1、在测试的过程中，发现MySQL 8.0.12的版本存在一个严重的BUG，redolog在checkpoint时发生崩溃，导致更新的时候会出现数据库假死，BUG的链接地址为：https://bugs.mysql.com/bug.php?id=90993，这个BUG在MySQL8.0.13的版本中修复。
- 2、MySQL 8.0的默认用户密码验证发生了变化，在数据库升级的时候需要设置老的密码验证策略；授权语句也发生了变化，建议升级过程中数据和权限分开导出导入，使用最新的pt-show-grants工具导出支持MySQL 8.0的授权语句，然后导入新的数据库中；在JDBC驱动方面，需要使用最新的驱动；相关的运维系统需要做调整，做好兼容。
- 3、重构了不少代码，去掉了不少参数同时新增了很多参数，需要多做功能测试。
- 4、持续关注，早做准备。

JD.COM 京东

谢谢观赏