一步到位实现MySQL优化

----从硬件、系统、配置、设计等多角度优化MySQL

叶金荣

QQ群： 373900864 、125572178、272675472

weibo：@yejinrong，http://imysql.com， 微信公众号：MySQL中文网

2014.10

About Me

- 叶金荣，网络常用ID：yejr
- Oracle ACE(MySQL)
- 2006年创办国内首个MySQL专业技术网站 http://imysql.com
- 精通MySQL数据库，10年+MySQL相关工作经验，擅长MySQL优化、架构设计、故障处理

About Me

MySQL经历

- MySQl 3.32.48 ~ now(5.7)
- 2000 ~ now
- 2012, ORACLE ACE(MySQL)

Linux经历

- 蓝点Linux 2.0、TurboLinux、FreeBSD、Debian、Ubuntu、Fedora、CentOS、RHEL

About Me

联系方式

- http://imysql.com（国内首个MySQL技术博客站，From 2006）
- QQ群： 373900864 、125572178、272675472
- QQ：4700963
- Weibo: @叶金荣
- 微信公众号：MySQL中文网
- imysql@gmail.com

MySQL的特点

- Not ORACLE，Not SQL Server，Not PostgreSQL
- Not Excel, Not Access
- Not File Storage
- Not Calculator
- Not Seach Engin
- Not ...
- MySQL is MySQL

MySQL的特点

CPU的利用特点

- <5.1，多核心支持较弱
- 5.1，可利用4个核
- 5.5，可利用24个核
- 5.6，可利用64个核
- 每个连接对应一个线程，每个并发query只能使用到一个核

MySQL的特点

内存利用特点

- 类似ORACLE的SGA、PGA模式，注意PGA不宜分配过大
- 内存管理简单、有效。在高TPS、高并发环境下，可增加物理内存以减少物理IO，提高并发性能
- 官方分支锁并发竞争比较严重，MariaDB、Percona进行优化
- 有类似ORACLE library cache的query cache，但效果不佳，建议关闭
- 执行计划没有缓存（类似ORACLE的library cache）
- 通常内存建议按热点数据总量的15%-20%来规划，专用单实例则可以分配物理内存的50~70%左右
- 类似K-V简单数据，采用memcached、Redis等NOSQL来缓存

MySQL的特点

对磁盘的利用特点

- binlog、redo log、undo log主要顺序IO
- datafile是随机IO和顺序IO都有
- OLTP业务以随机IO为主，建议加大内存，尽量合并随机IO为顺序IO
- OLAP业务以顺序IO为主，极大内存的同时增加硬盘数量提高顺序IO性能
- MyISAM是堆组织表（HOT），InnoDB是索引组织表（IOT）
- InnoDB相比MyISAM更消耗磁盘空间

优化思路

- 确认问题
- 确认瓶颈
- 制定方案
- 测试方案
- 实施方案
- 回顾反馈

确认瓶颈

- top
- vmstat
- sar
- iotop
- dstat
- oprofile

确认瓶颈

- slow log
- show global status
- show processlist
- show engine innodb status
- pt-ioprofile

硬件优化 – BIOS设置优化

- System Profile（系统配置）选择Performance Per Watt Optimized(DAPC)，发挥最大功耗性能
- Memory Frequency（内存频率）选择Maximum Performance（最佳性能）
- C1E，允许在处理器处于闲置状态时启用或禁用处理器切换至最低性能状态，建议关闭（默认启用）
- C States（C状态），允许启用或禁用处理器在所有可用电源状态下运行，建议关闭（默认启用）

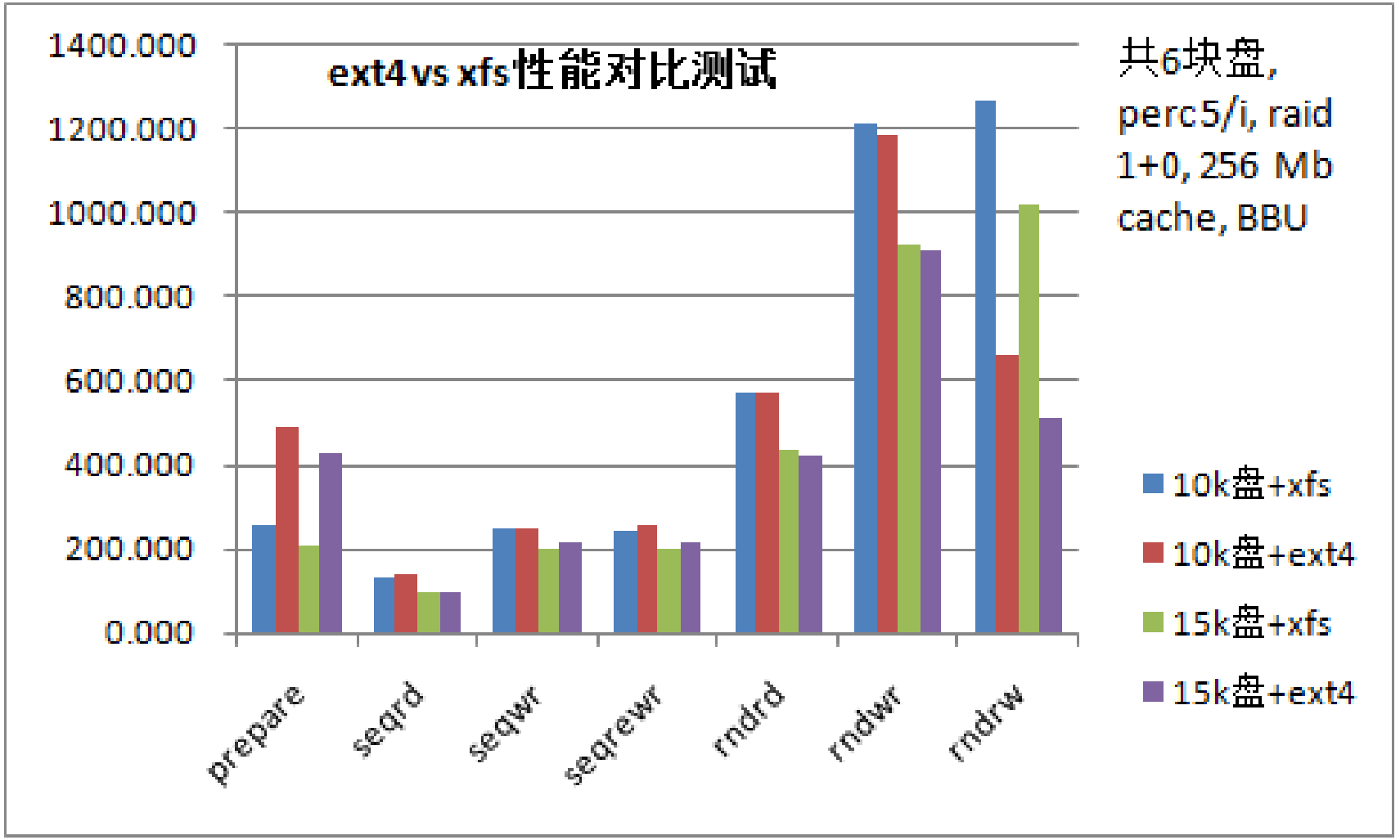
硬件优化 – IO子系统优化

- 阵列卡配备CACHE及BBU模块，提高IOPS
- 设置写策略为WB，或者FORCE WB，禁用WT策略
- 关闭预读，没必要预读，那点宝贵的CACHE用来做写缓存
- 阵列级别使用RAID 1+0，而不是RAID 5
- 关闭物理磁盘cache策略，防止丢数据
- 使用高转速硬盘，不使用低转速盘
- 使用SSD或者PCIe-SSD盘

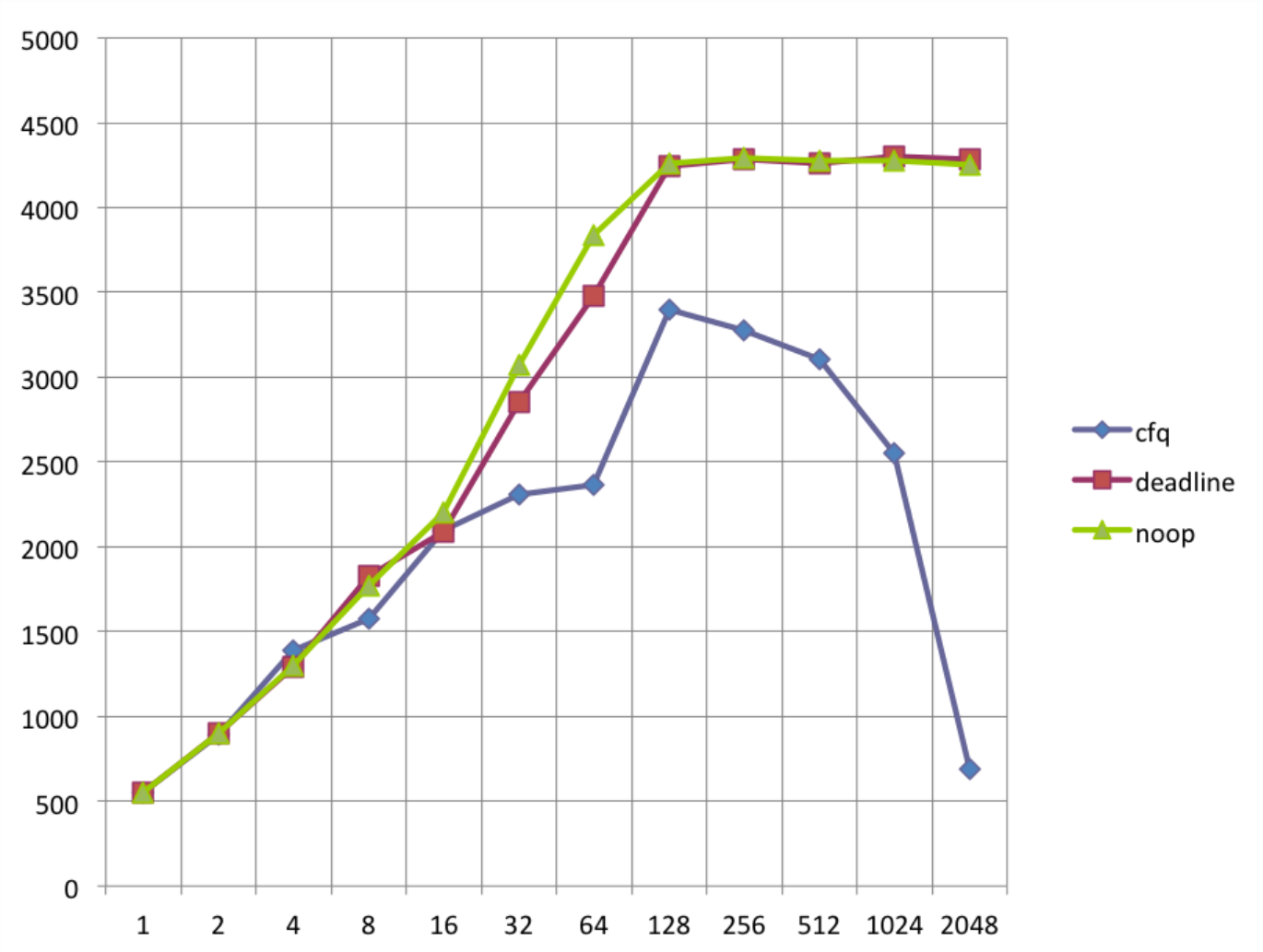
系统优化

- vm.swappiness
- /sys/block/sdX/queue/scheduler
- 文件系统首选xfs，其次ext4，zfs也很不错，但在linux下不是那么可靠

系统优化


ext4 vs xfs性能对比测试
1400.000
1200.000
1000.000
800.000
600.000
400.000
200.000
0.000
prepare
seqrd
seqwr
seqrewr
rndrd
rndwr
rndrw
共6块盘，perc5/i, raid 1+0, 256 Mb cache, BBU
10k盘+xfs
10k盘+ext4
15k盘+xfs
15k盘+ext4

系统优化

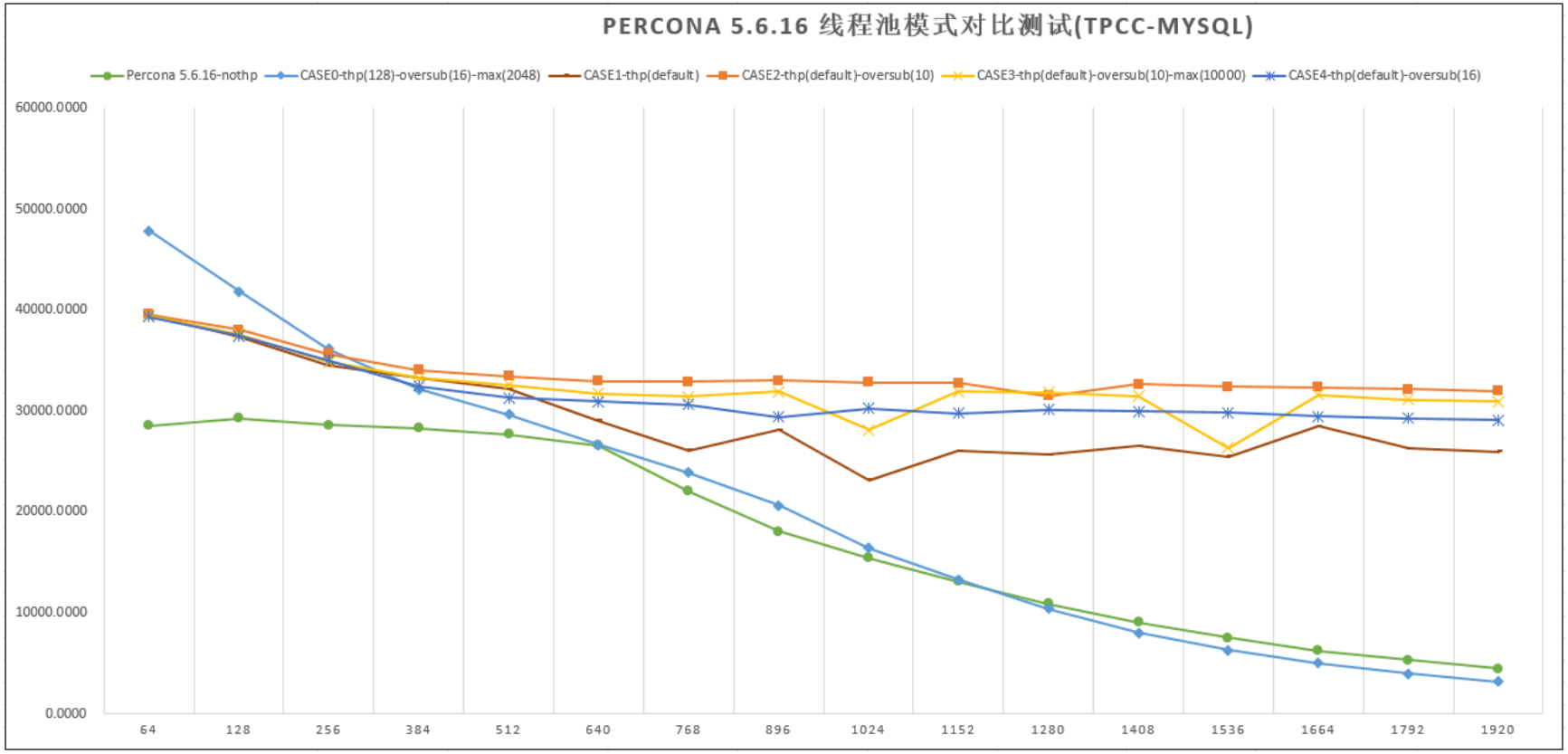

5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1
2
4
8
16
32
64
128
256
512
1024
2048
cfq
deadline
noop

配置优化 – 全局参数

- interactive_timeout、wait_timeout
- open_files_limit
- max_connections
- thread_pool

配置优化 – 全局参数

- thread_pool



配置优化 – 内存相关

内存相关参数

mysql使用总内存 = global_buffers + thread_buffers

global buffer(全局内存分配总和) =

innodb_buffer_pool_size

+innodb_additional_mem_pool_size

+innodb_log_buffer_size

+key_buffer_size

+query_cache_size

+table_open_cahce

+table_definition_cache

+thread_cache_size

All thread buffer(会话/线程级内存分配总和) =

max_threads * (

read_buffer_size

+read_rnd_buffer_size

+sort_buffer_size

+join_buffer_size

+binlog_cache_size

+tmp_table_size

+thread_stack

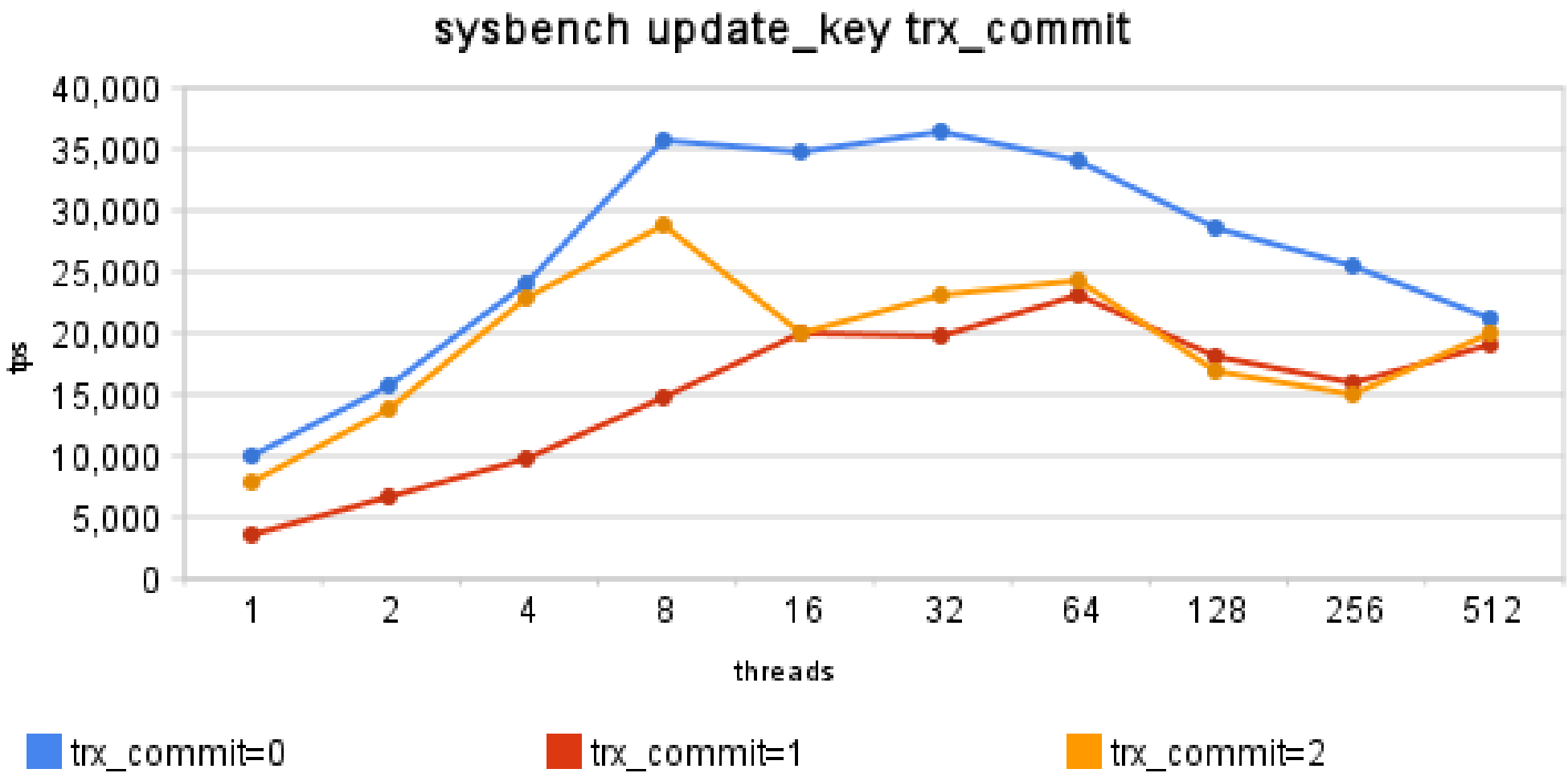
+net_buffer_length

+ bulk_insert_buffer_size)

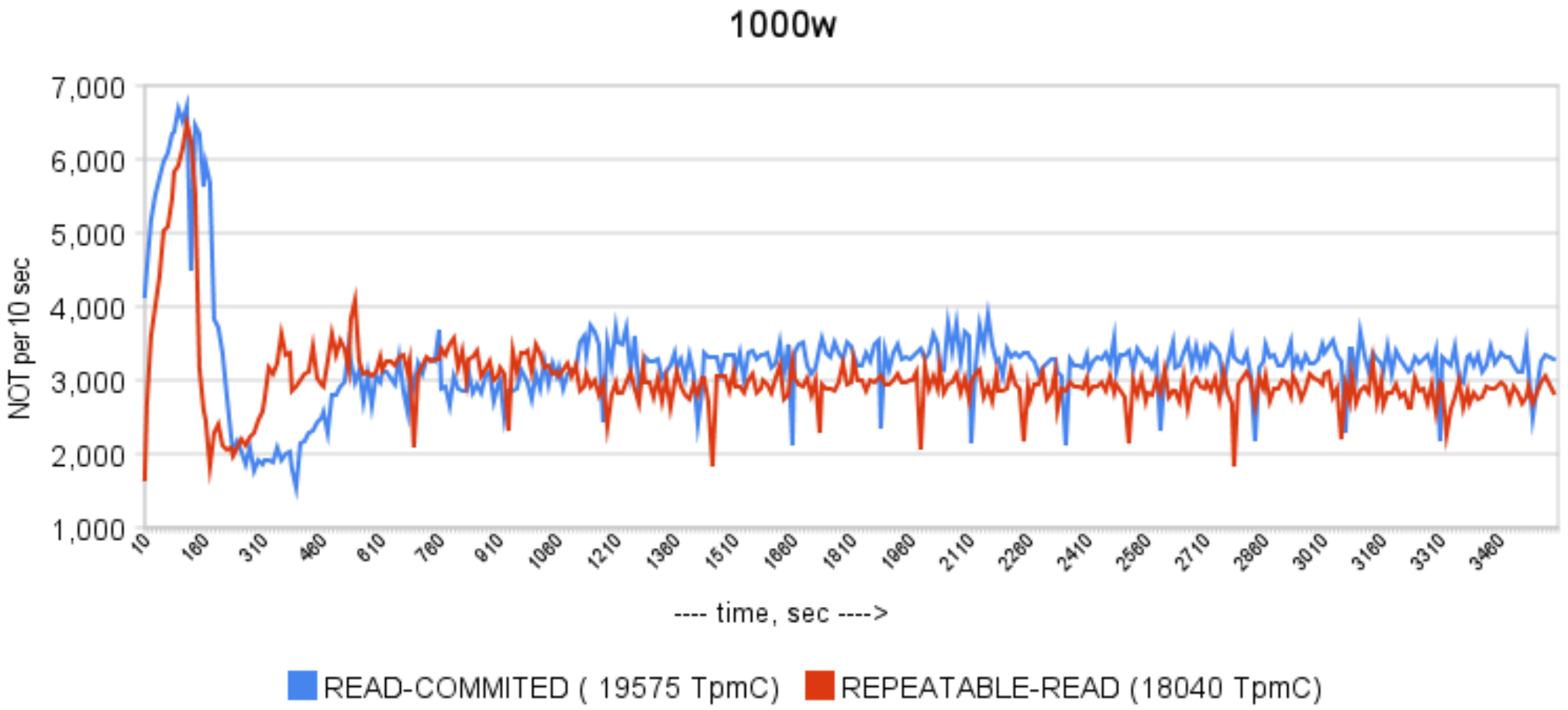
配置优化 – InnoDB相关

- innodb_buffer_pool_size
- innodb_data_file_path
- innodb_flush_log_at_trx_commit
- innodb_log_buffer_size & innodb_log_file_size
- transaction_isolation

配置优化 – InnoDB相关



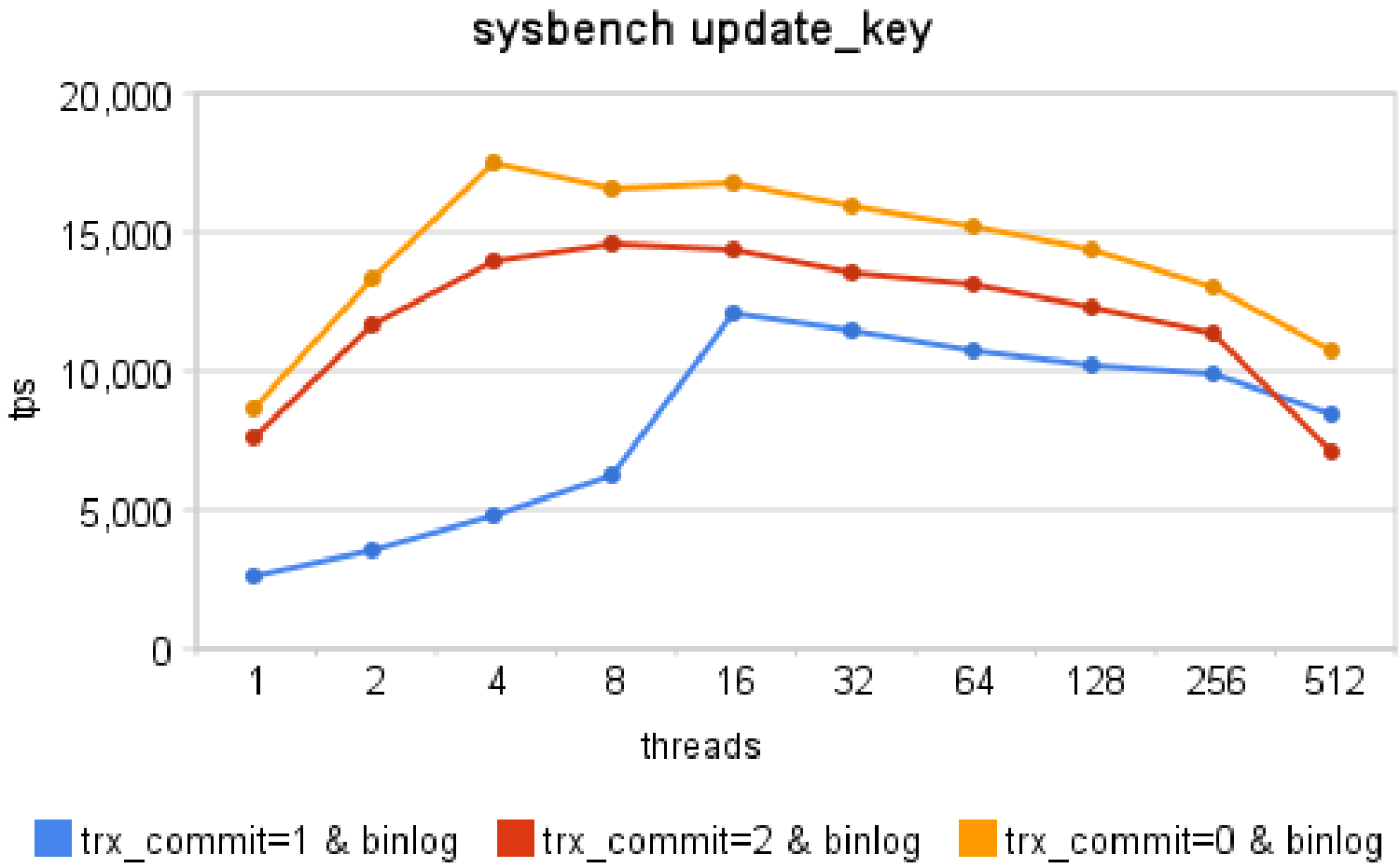
配置优化 – InnoDB相关



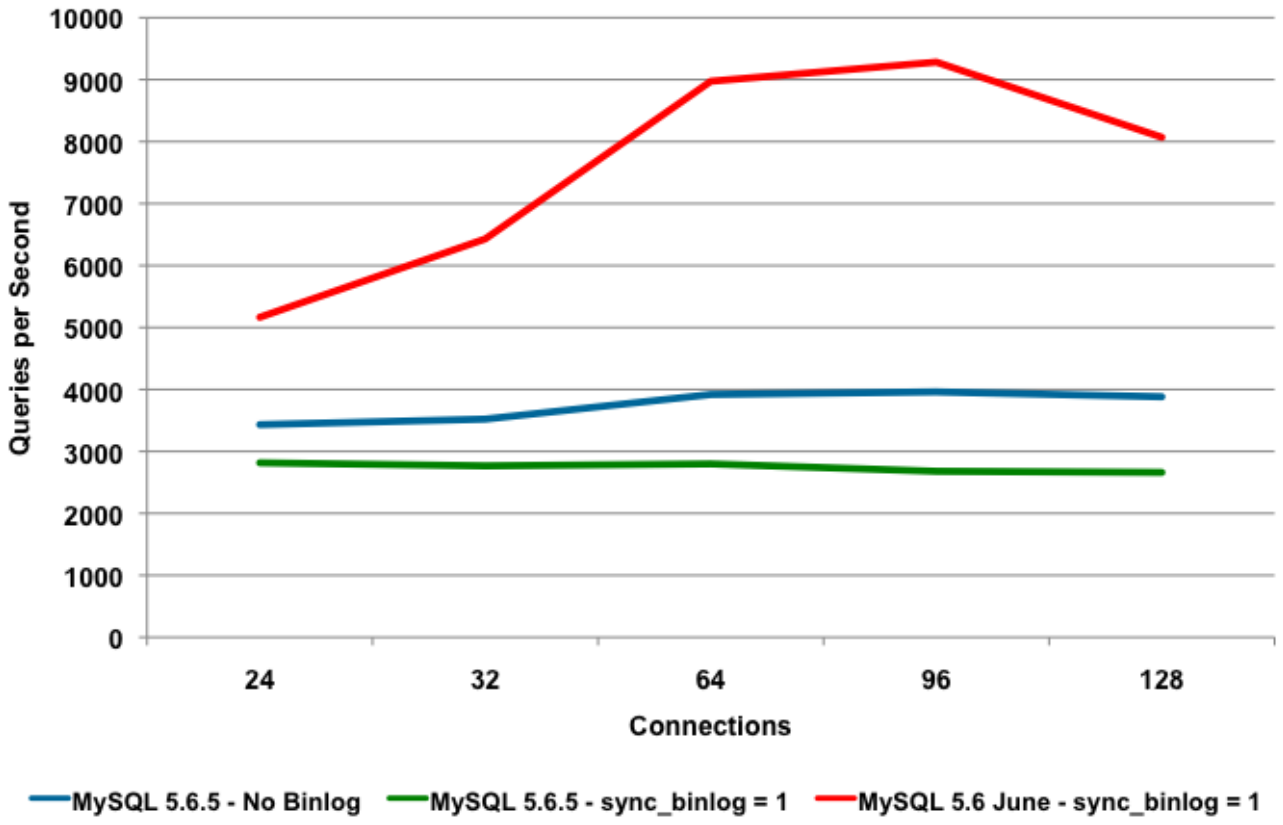
配置优化 – 其他

- general_log
- log_bin
- sync_binlog
- long_query_time
- log_slow_query

配置优化 – InnoDB相关



配置优化 – InnoDB相关



设计优化 – 先入为主

默认使用InnoDB引擎，可适用99%以上业务场景

- **并发** – 没人愿意所有的请求都被串行的执行完成
- **数据一致性** – 交易类业务要求数据高一致性，确保数据完整性
- **Crash Recovery** – 故障自动修复，修复相比MyISAM速度更快
- **更高存取效率** – 行锁减低锁粒度，更高内存利用率提高数据、索引存取效率

设计优化 – Schema设计

- 不管InnoDB与否，都设计自增列主键
- 日期时间、IPV4适用INT UNSIGNED存储
- 性别、是否等枚举类型，使用ENUM/TINYINT，而非CHAR/VARCHAR
- 杜绝TEXT/BLOB，可以做垂直拆分，或者转成MyISAM表
- USERNAME：~~VARCHAR(255)~~ VS VARCHAR(30) vs ~~CHAR(30)~~
- 所有字段显式定义NOT NULL

设计优化 – Schema设计

- 基数（ Cardinality ）很低的字段不创建索引（MySQL还不支持 bitmap 索引）
- 采用第三方系统实现text/blob全文检索
- 常用排序（ORDER BY）、分组（GROUP BY）、取唯一（DISTINCT）字段上创建索引
- 索引数量不要太多，有负作用
- 多使用联合索引，少用单独索引
- 字符型列需要索引时，创建前缀索引

设计优化 – 无法使用索引的场景

- 通过索引扫描的记录数超过30%，变成全表扫描
- 联合索引中，第一个索引列使用范围查询 -- 只能用到部分索引
- 联合索引中，第一个查询条件不是最左索引列
- 模糊查询条件列最左以通配符 % 开始
- 内存表(HEAP 表)使用HASH索引时，使用范围检索或者ORDER BY
- 两个独立索引，其中一个用于检索，一个用于排序 -- 只能用到其中一个索引，5.6以上有ICP特性
- 表关联字段类型不一样（也包括长度不一样）
- 索引字段条件上使用函数

设计优化 – 常见杀手级SQL

- SELECT * vs SELECT col1, col2
- ORDER BY RAND()
- LIMIT huge_num, offset
- SELECT COUNT(*) on InnoDB table
- WHERE func(key_col) = ? -- 无法使用索引
- WHERE key_part2 =? AND key_part3 =? -- 无法使用索引
- WHERE key_part1 > ? AND key_part2 =? -- 只能用到部分索引
- SELECT … WHERE key_col + ? = ? -- 无法使用索引

设计优化 – 常见杀手级SQL

SELECT a.x ...

FROM a

ORDER BY a.y LIMIT 11910298, 20;

采用子查询进行优化 =>

SELECT a.x ...

FROM a

WHERE a.pkid > (SELECT pkid FROM a WHERE pkid >= 11910298 ORDER BY a.y) LIMIT 20;

设计优化 - 架构设计

- 减少物理I/O，让MySQL闲下来
- 转变随机I/O为顺序I/O
- 减小活跃数据
- 分库分表
- 读写分离
- OLTP、OLAP分离

优化工具

- pt-ioprofile
- mysqldumpslow
- pt-query-digest + Box Anemometer/Query-Digest-UI

常见优化误区

- 分配内存越多越好，可能导致OS Swap
- session级内存分配过大，导致OOM
- 索引越多越好，可能导致更多IO
- Qcache设置过大，实际效果差
- 认为MyISAM的只读效率远高于InnoDB
- 人云亦云，不自己动手实践
- 过度优化，反而带来成本的上升

谢谢关注！

若本次分享对您有帮助，请转发扩散，谢谢！

在 http://imysql.com 中获取更多内容