

MySQL新技术在淘宝的使用

应元



追風堂



Solid-State Drives

FUSION-io

HandlerSocket





- MySQL数据库的用途？
- MySQL总体架构
- 常见的Tair+MySQL(InnoDB)应用架构
- 常见的MySQL服务器硬件架构
- 核心数据库 MySQL集群概况
- 新出现的硬件技术(Flash:SSD/FusionIO)
- HandlerSocket-基于MySQL实现的NoSQL插件
- Percona VS MySQL
- 讨论时间
- 课后思考



MySQL数据库的用途



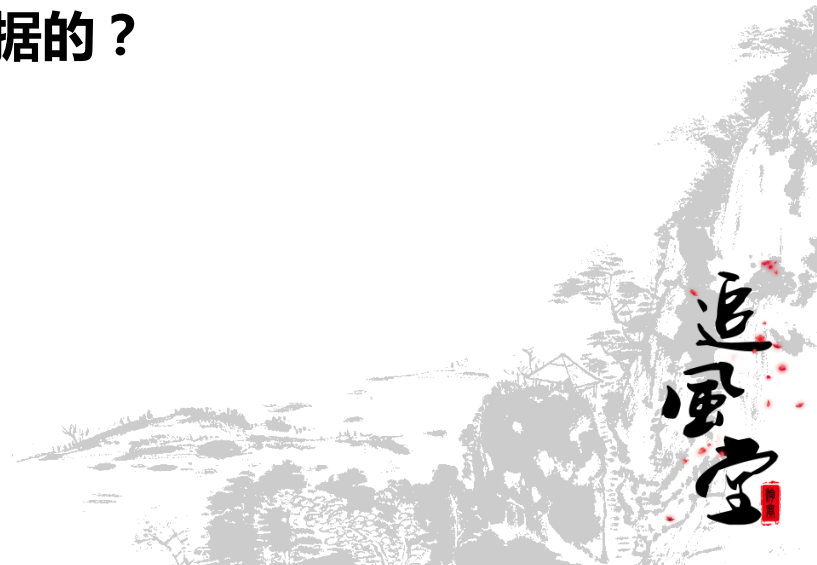
- 讨论大家平常都用MySQL来干些什么事情
- ?



MySQL数据库的用途

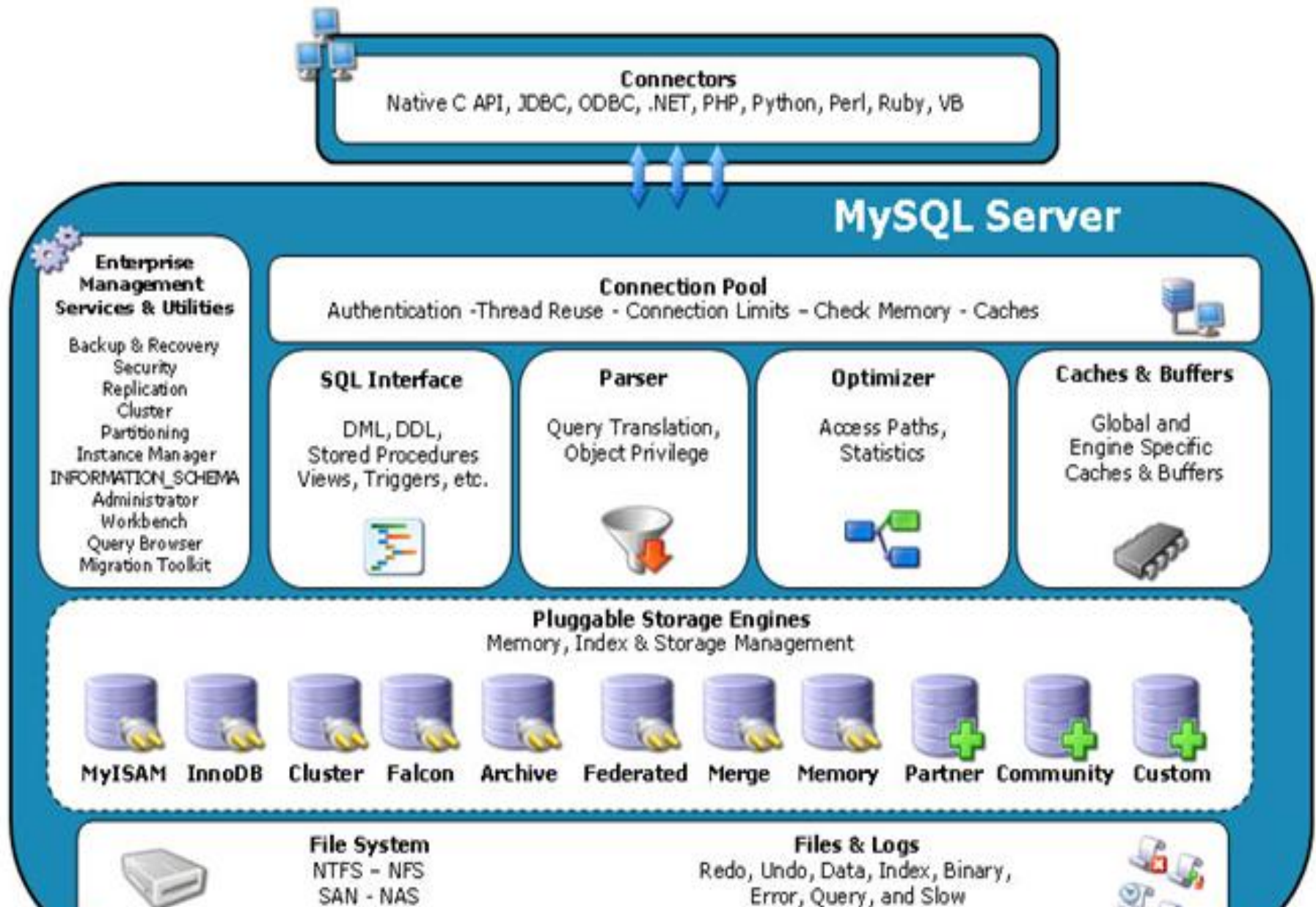


- 写配置,记录用户信息,记录交易信息,记录商品信息...
- 读配置,读用户信息,读交易信息,读商品信息
- 所有的行为都可以归结为 写数据,读数据
- MySQL是如何为我们进行读数据和写数据的？



追風堂

MySQL的总体架构



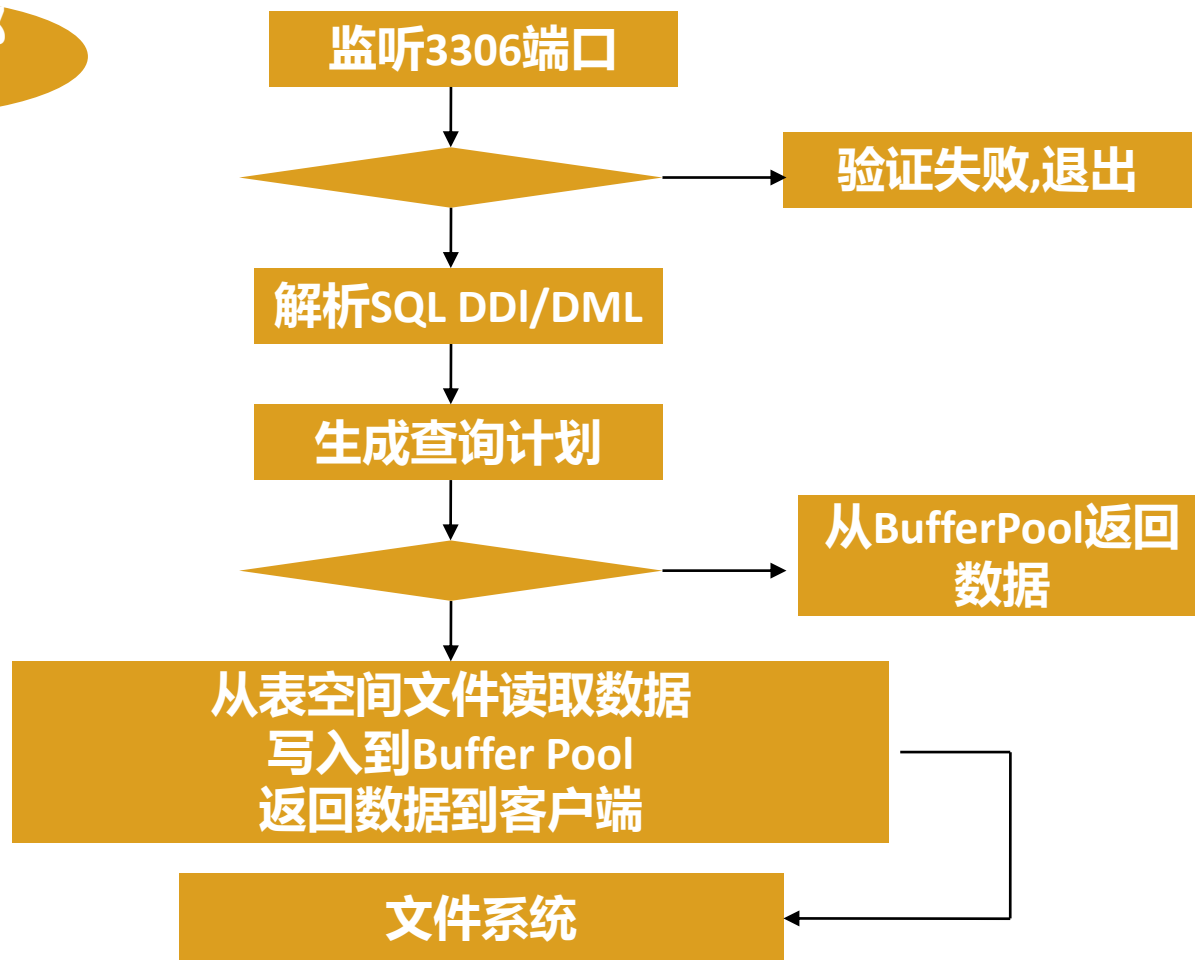
One Story of Query In MySQL(InnoDB)

- MySQL服务器监听3306端口
- 验证用户
- 创建线程解析SQL
- 查询优化
- 打开表
- 检查Buffer Pool是否有对应的缓存记录
- 到磁盘捞数据
- 写入到缓存
- 返回数据给客户端
- 关闭表
- 关闭线程
- 关闭连接



One Story of Select In MySQL(InnoDB)

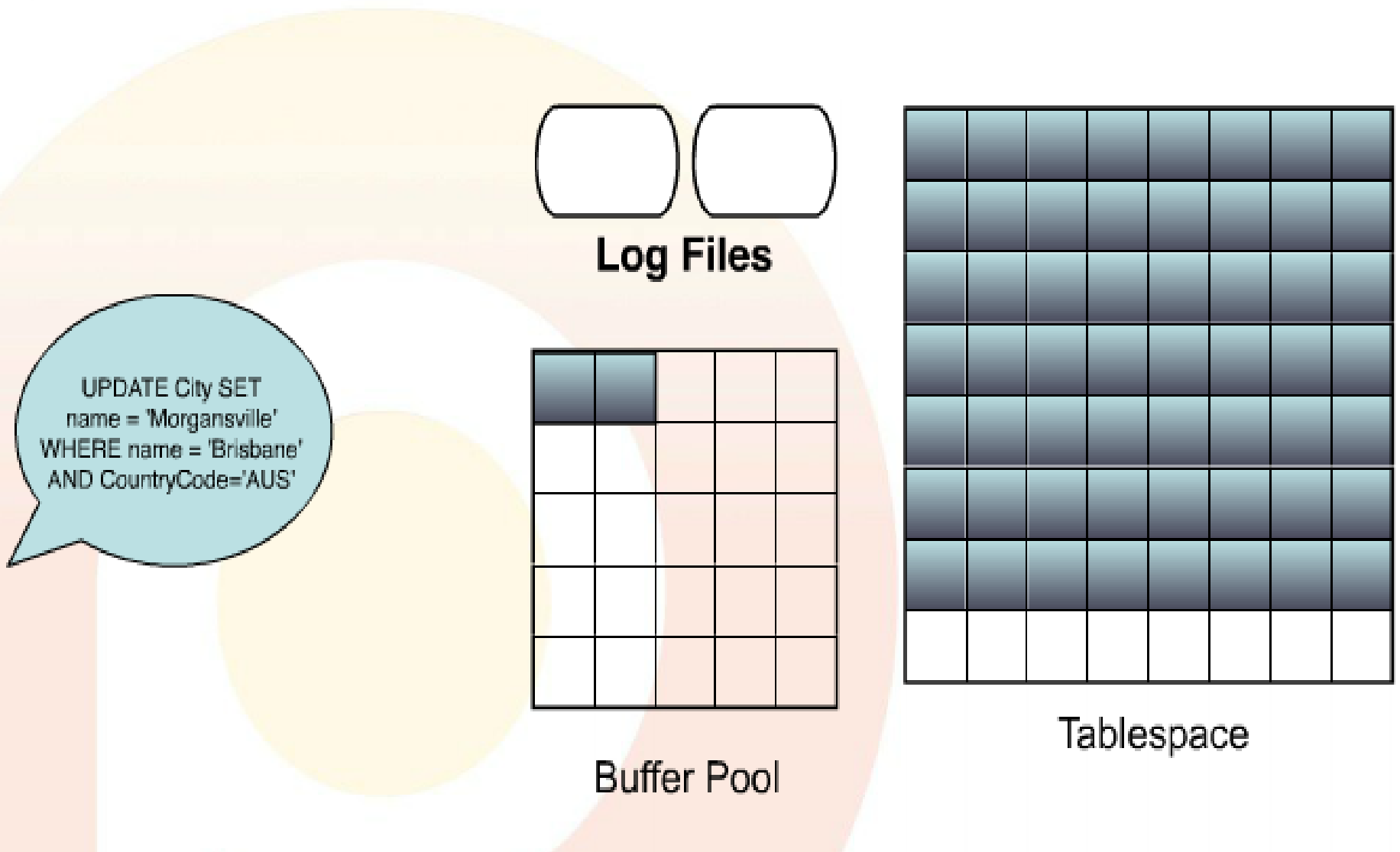
MySQL内部 流程



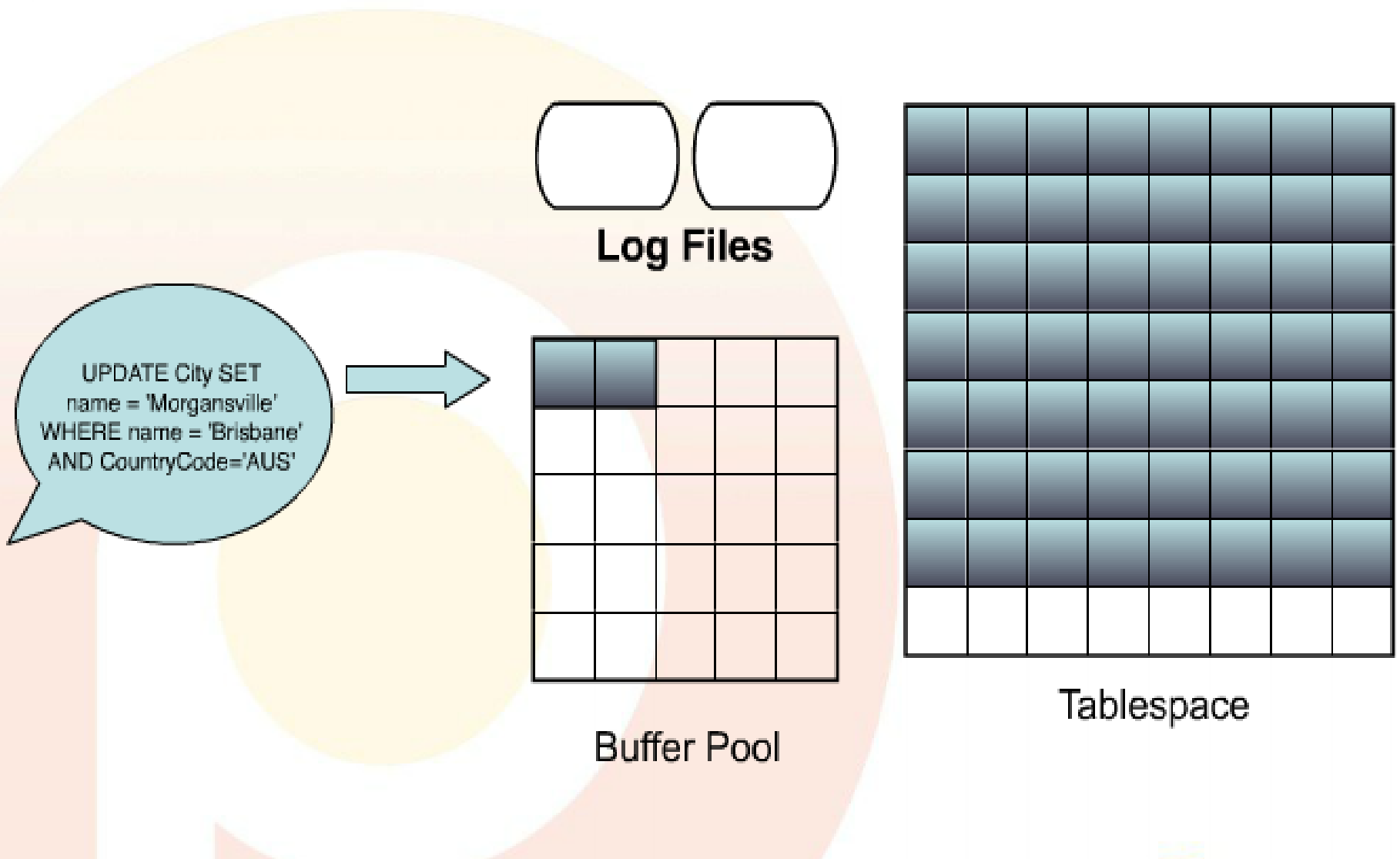
硬件



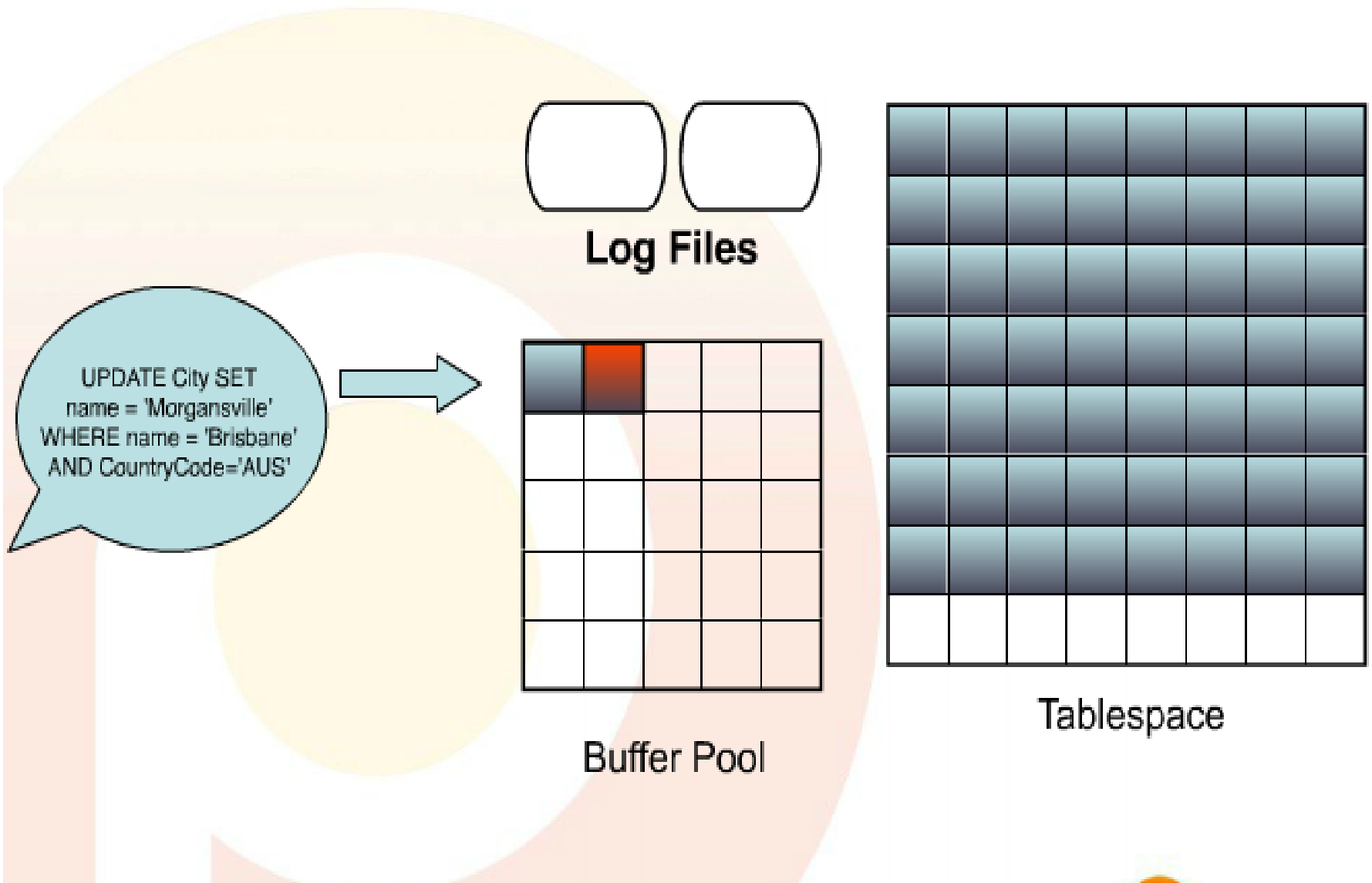
One Story of TPS In MySQL(InnoDB)



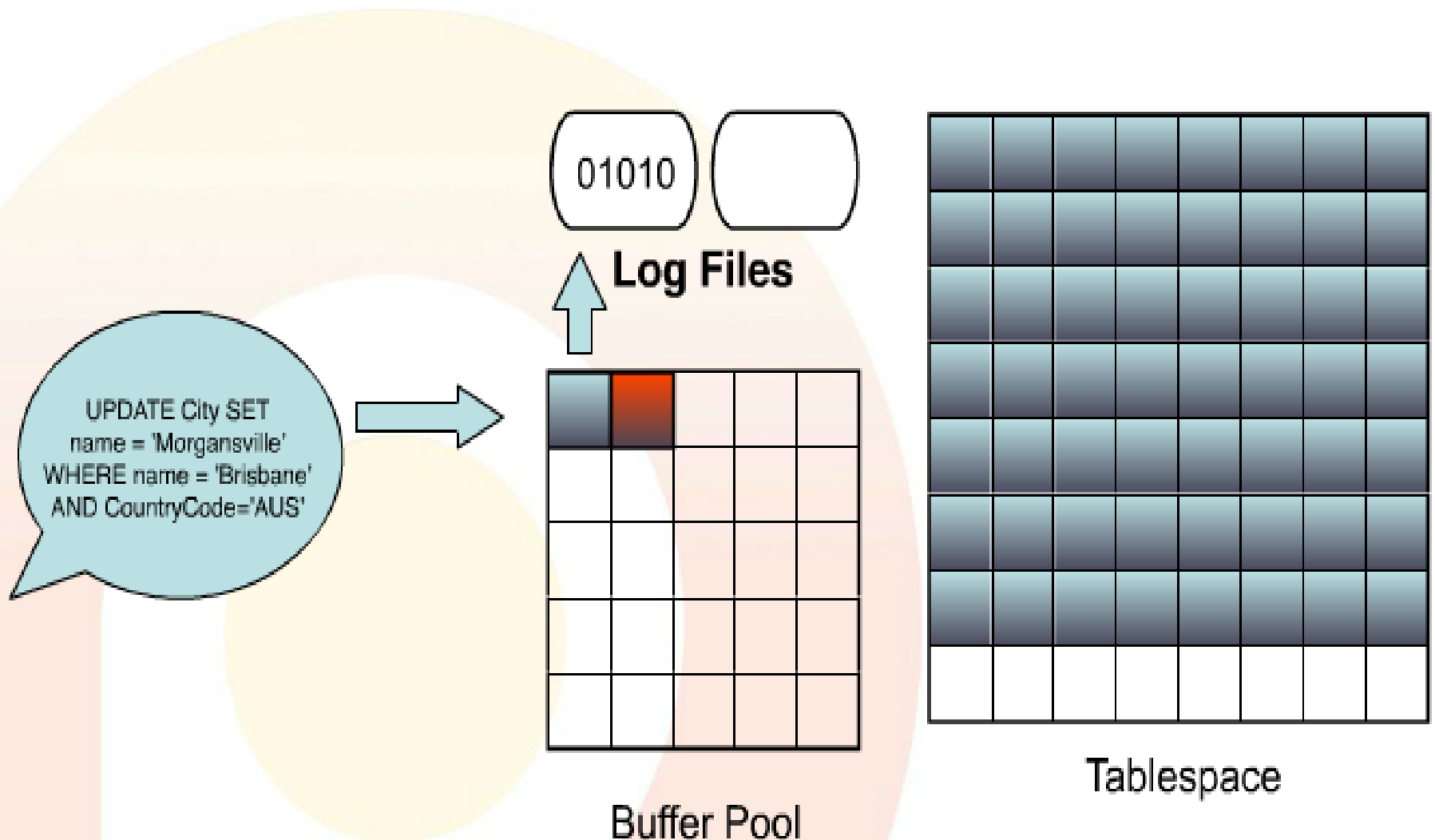
One Story of Insert In MySQL(InnoDB)



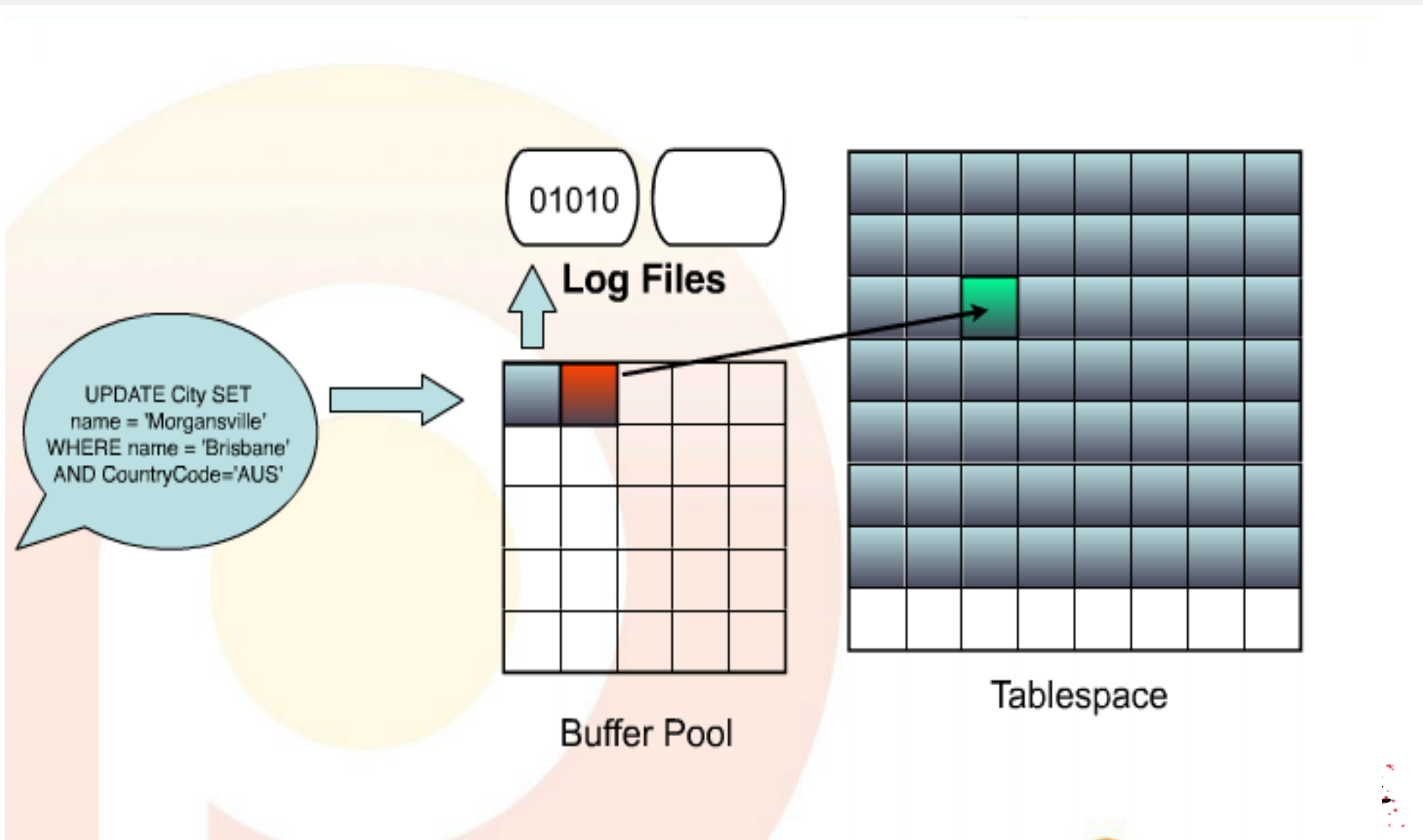
One Story of TPS In MySQL(InnoDB)



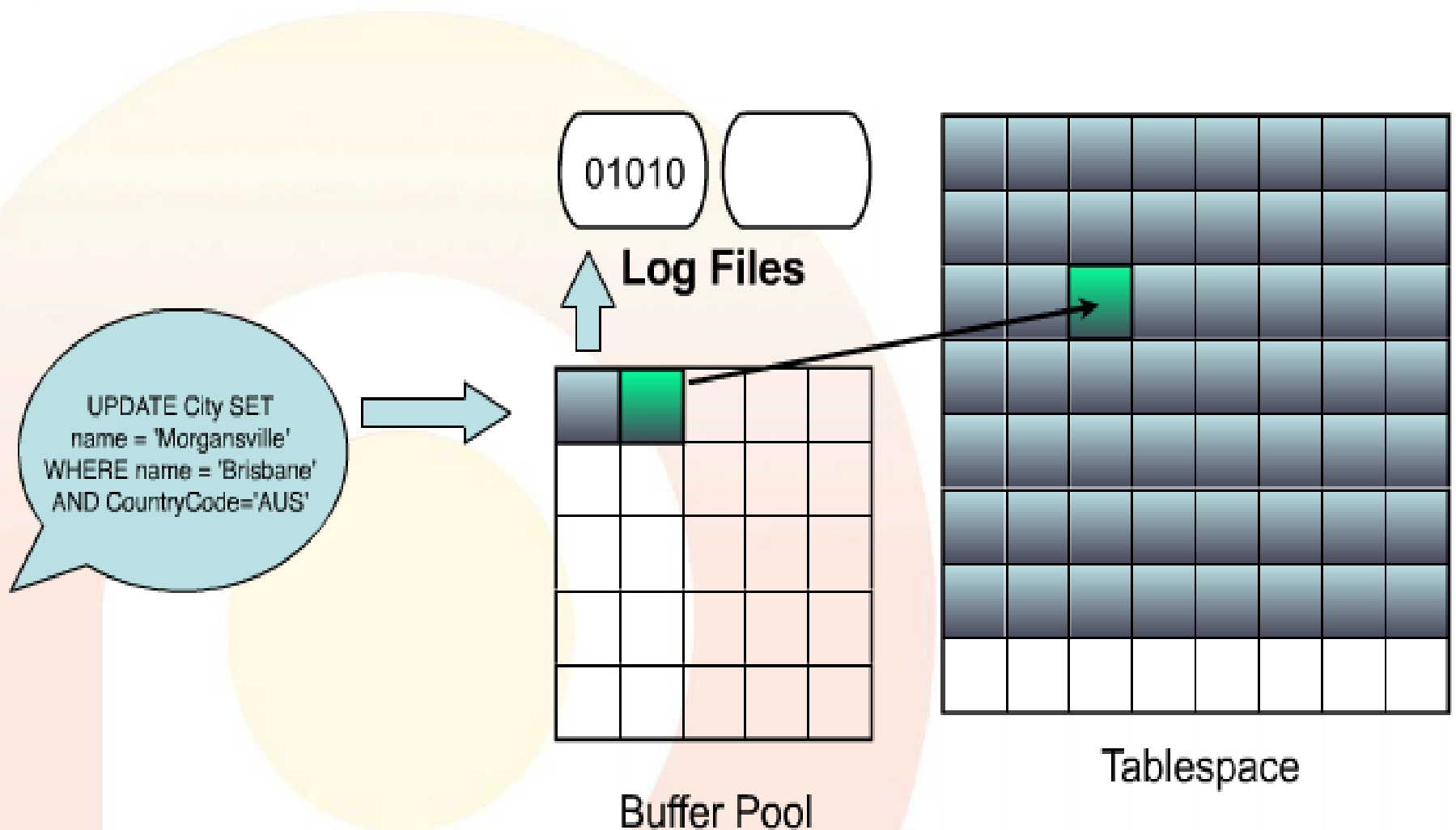
One Story of TPS In MySQL(InnoDB)



One Story of TPS In MySQL(InnoDB)



One Story of TPS In MySQL(InnoDB)



故事小结

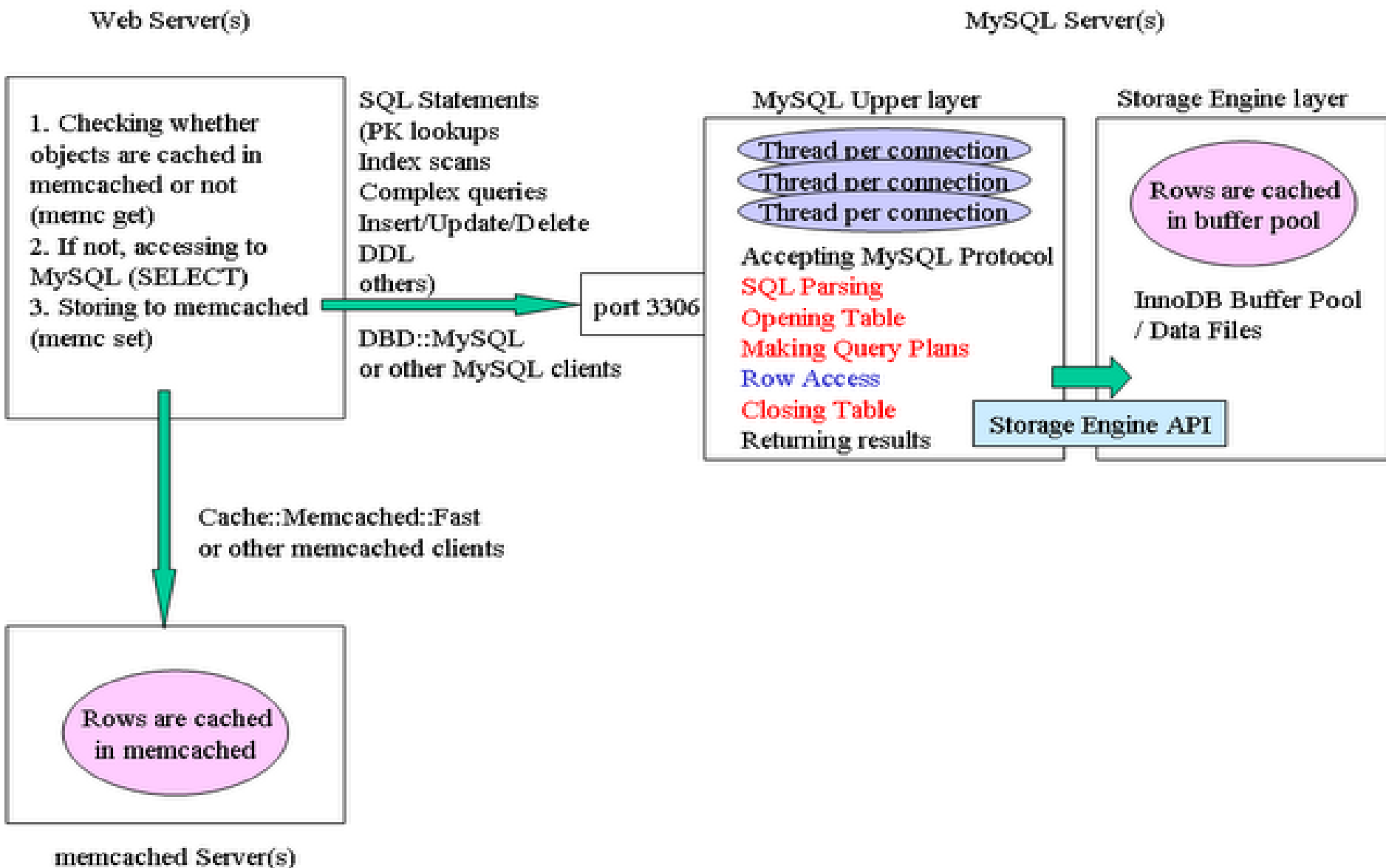
- 如何更快的让查询返回我们想要的数据库?
- 如何更快的让我们的数据写入?
- 我们今天讲的MySQL新技术，就是围绕这两个故事来开展

让查询更快的返回

- 我们做了哪些努力？
- 整体架构
 - App前端缓存-Tair
- MySQL(InnoDB)
 - Buffer Pool 缓存数据和索引信息



常见Tair+MySQL的应用架构



Tair变化情况



Tair+MySQL架构的优缺点

- **优点**

- Tair内部获取数据是hash get,速度比MySQL的B-Tree速度要好
- Tair服务器可以缓存大部分的热点数据

- **缺点**

- 应用程序增加一层逻辑判断
- Tair能帮助提速查询，但不能直接提升数据更新速度
- 硬件成本，运维成本提高
- 对于高QPS的应用，Tair服务器不能有异常



MySQL(InnoDB) Buffer Pool的小结

- Buffer Pool越大，能缓存的数据和索引就越多，QPS就越高
- Buffer Pool缓存命中率越高, DB热点数据查询性能就越好
- Buffer Pool依赖的是物理内存大小,一般是物理内存的60%-80%
- But...
 - 内存是昂贵的
 - 内存不是持久性的存储
 - SAS盘的IOPS有限



原有的MySQL服务器架构

- 内存 24G/48G/96G
- InnoDB buffer Pool 分配 物理内存的60%到80%
- 磁盘 8块到12块SAS盘 做Raid 10
- 网卡 千兆网卡
- SAS盘IOPS有限
- 核心数据库双十二例子
- `innodb_buffer_pool_size = 36G`
- `innodb_flush_log_at_trx_commit = 1`



双十二某核心库 单台DB负载情况

Load 趋势图



db qps 趋势图



双十二 某核心系统 Tair情况



- 某核心系统 读多写少的业务场景，可以让Tair尽情发挥
- 但不是所有的应用都和某核心系统那样，信息很少更新
- 其他核心数据库很多情况下不能走Tair
- 其他核心系统在DB进行的QPS和TPS,比某核心系统的挑战更大

第二个核心系统MySQL集群的故事

- 原有架构

- 48G内存 Raid 10 十二块SAS 盘
- 16主16备 两套备库

- 问题

- 高峰期主、备库load在10左右，应用将平均响应时间报警设置为2000ms还是每天告警不断，在浪费了不少短信费的同时也困扰了监控值班同学，最后不得不关掉报警
- TOP API每天因查询超时失败率在9-20%，天天催着业务方做优化、做升级，着实痛苦
- 业务上做了几次DDL，并对数据库新加字段做初始化，这个初始化过程非常辛苦。在升级SSD前，初始化8亿数据时，单机10个线程、总共100个线程来做更新操作，耗时3个晚上，而且第二天主备延迟极高
- 因为主库查询慢影响了后台客服小二查询评价数据，挨了一个P3级故障

第二个核心系统MySQL集群的故事

- **双十一前**

- 迁移到SSD机器
- 依然是16主16备，一套备库

- **双十一后**

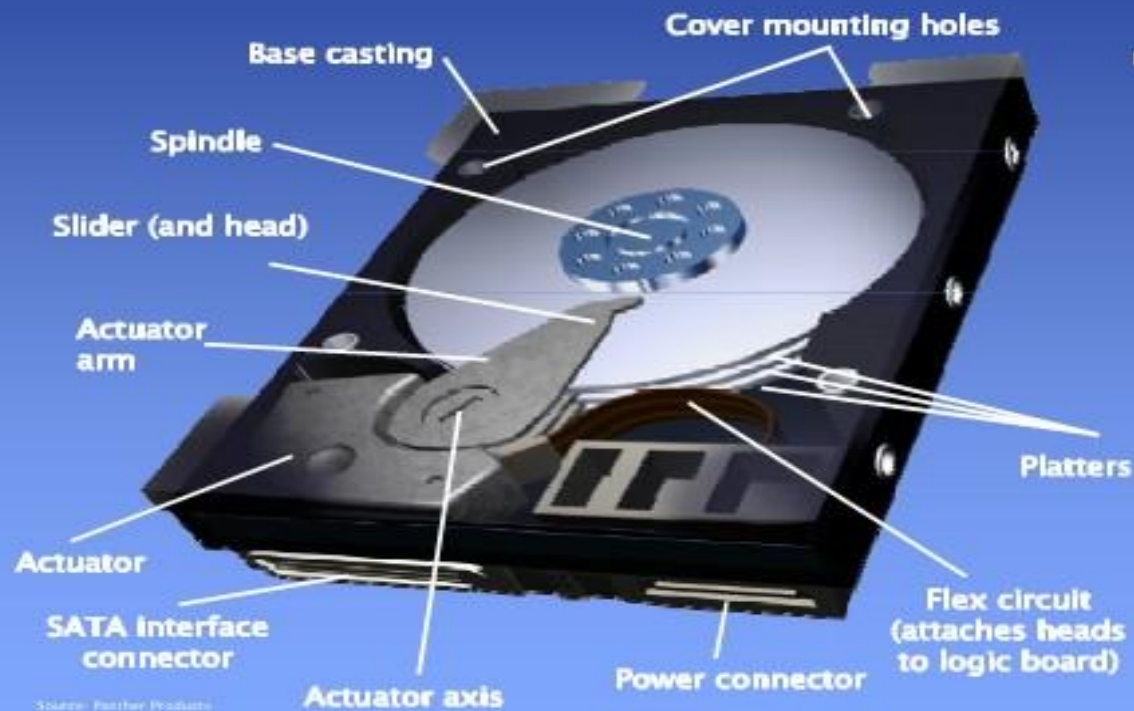
- DB很淡定的撑过了**5倍的查询**，给力！
- 项目上线后，对于好中差计数不准的，只能根据客服反馈来手动订正，因为db问题，没法进行全量count对账。现在，**白天开启对账，db压力也很小**，解决了客服的烦恼，真正的从底层解决了用户体验



服务器硬件新技术



The Complexity of HDDs



■ HDD Advantages

- Density
- Price/GB

Source: Peerlier Products

Santa Clara, CA USA
August 2008

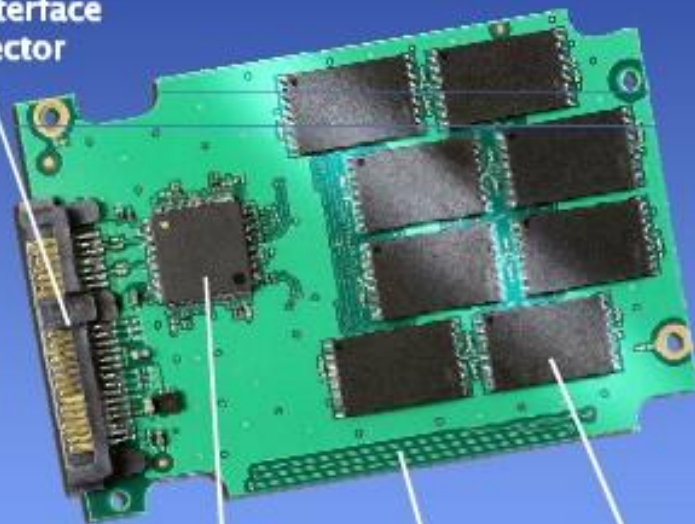


服务器硬件新技术



The Simplicity of SSDs

SATA Interface
connector



Controller

Printed Circuit

NAND

■ SSD Advantages

- Performance
- Size
- Weight
- Ruggedness
- Temperature Range
- Power

MySQL服务器新架构

Tested HDD/SSD for this session

- SSD
 - Intel X25-E (SATA, 30GB, SLC)
 - Fusion I/O (PCI-Express, 160GB, SLC)
- HDD
 - Seagate 160GB SAS 15000RPM



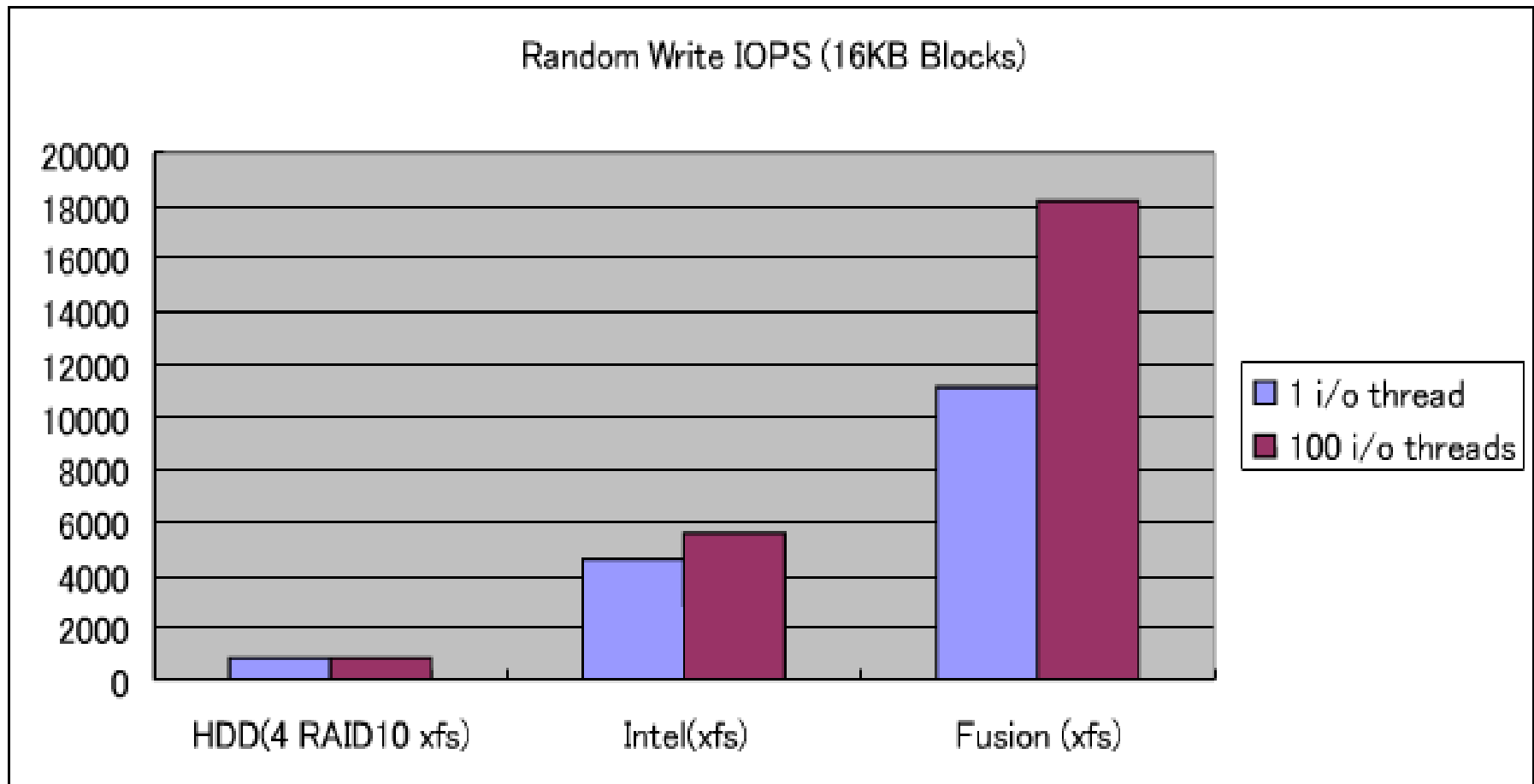
测试场景---MySQL服务器新架构

- 随机读
- 随机写
- 顺序读
- 顺序写
- IOPS和I/O block大小



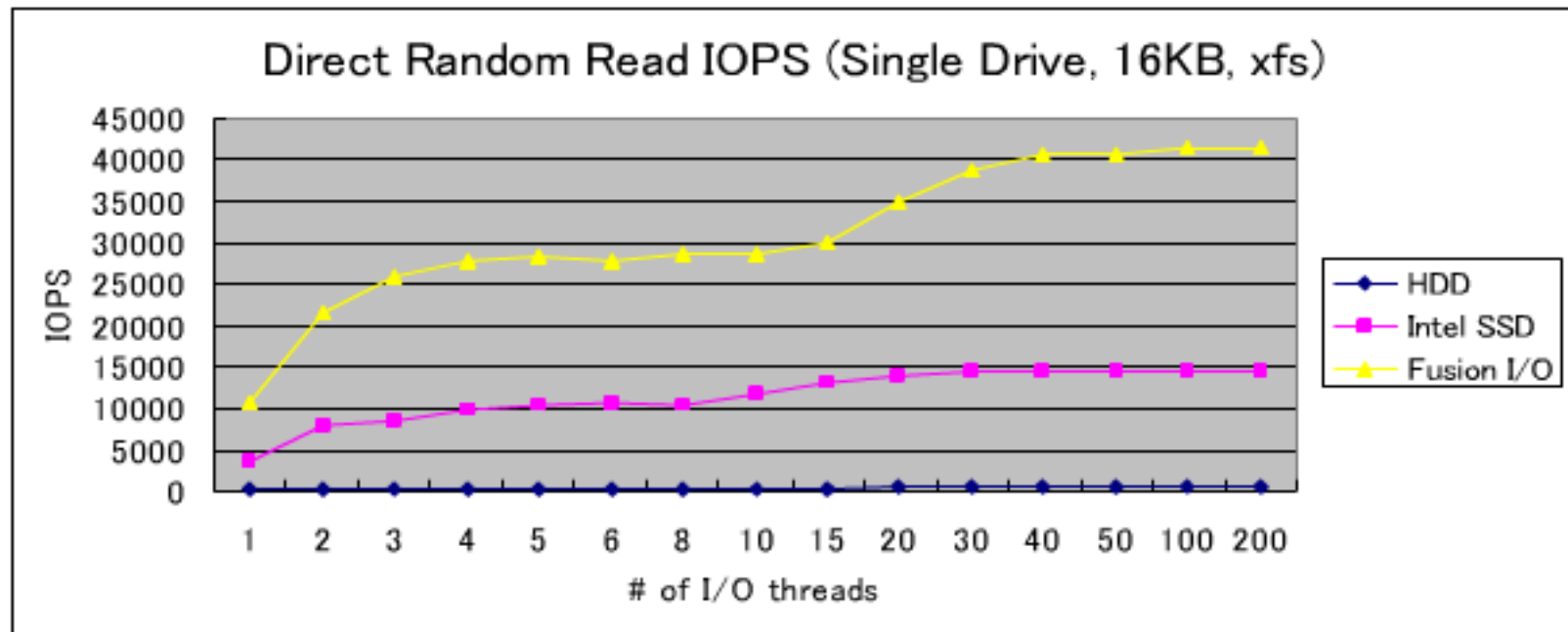
MySQL服务器新架构

Write performance on SSD



MySQL服务器新架构

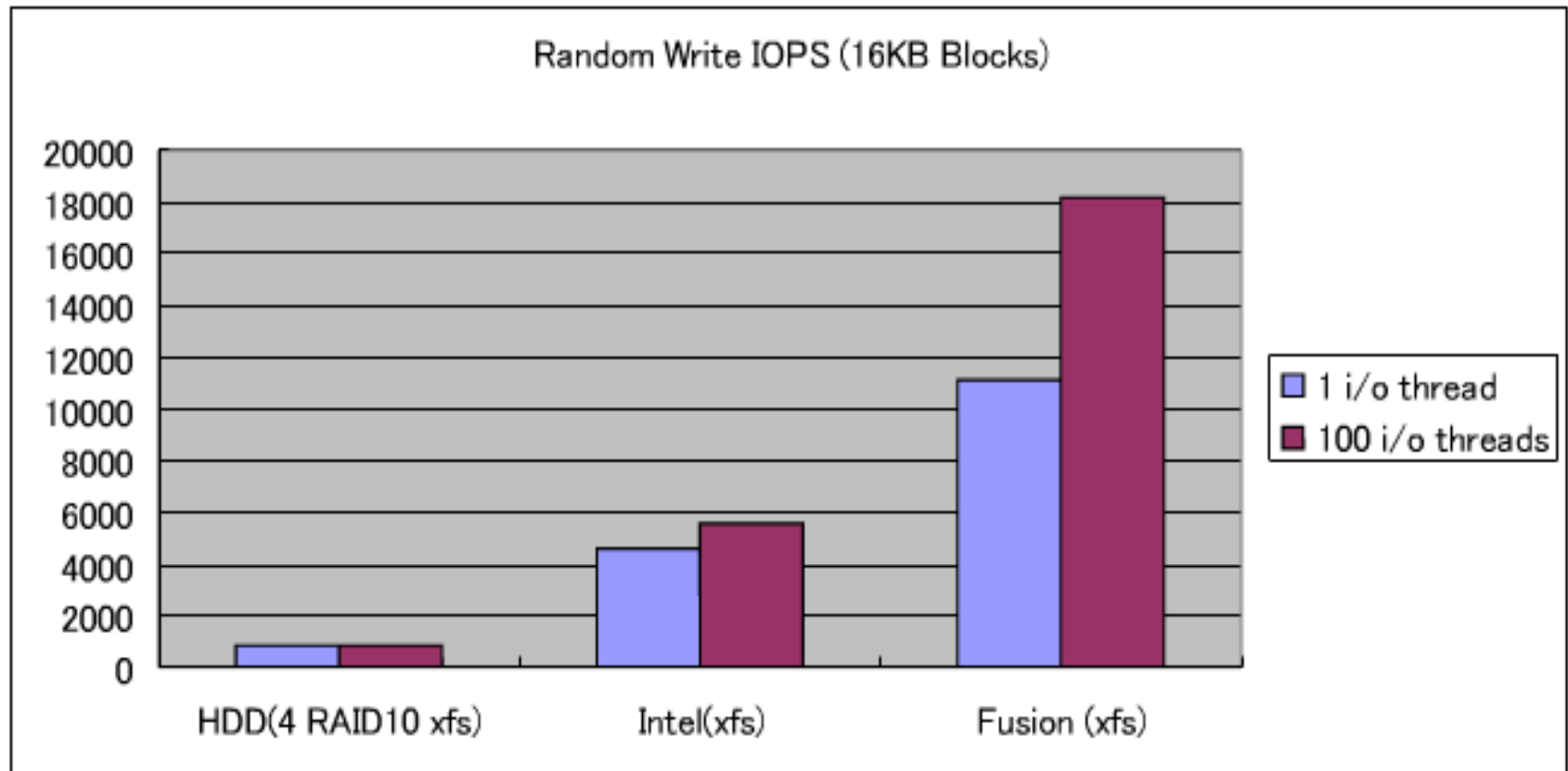
Random Read benchmark



- HDD: 196 reads/s at 1 i/o thread, 443 reads/s at 100 i/o threads
- Intel : 3508 reads/s at 1 i/o thread, 14538 reads/s at 100 i/o threads
- Fusion I/O : 10526 reads/s at 1 i/o thread, 41379 reads/s at 100 i/o threads
- Single thread throughput on Intel is 16x better than on HDD, Fusion is 25x better
- SSD's concurrency (4x) is much better than HDD's (2.2x)
- Very strong reason to use SSD

MySQL服务器新架构

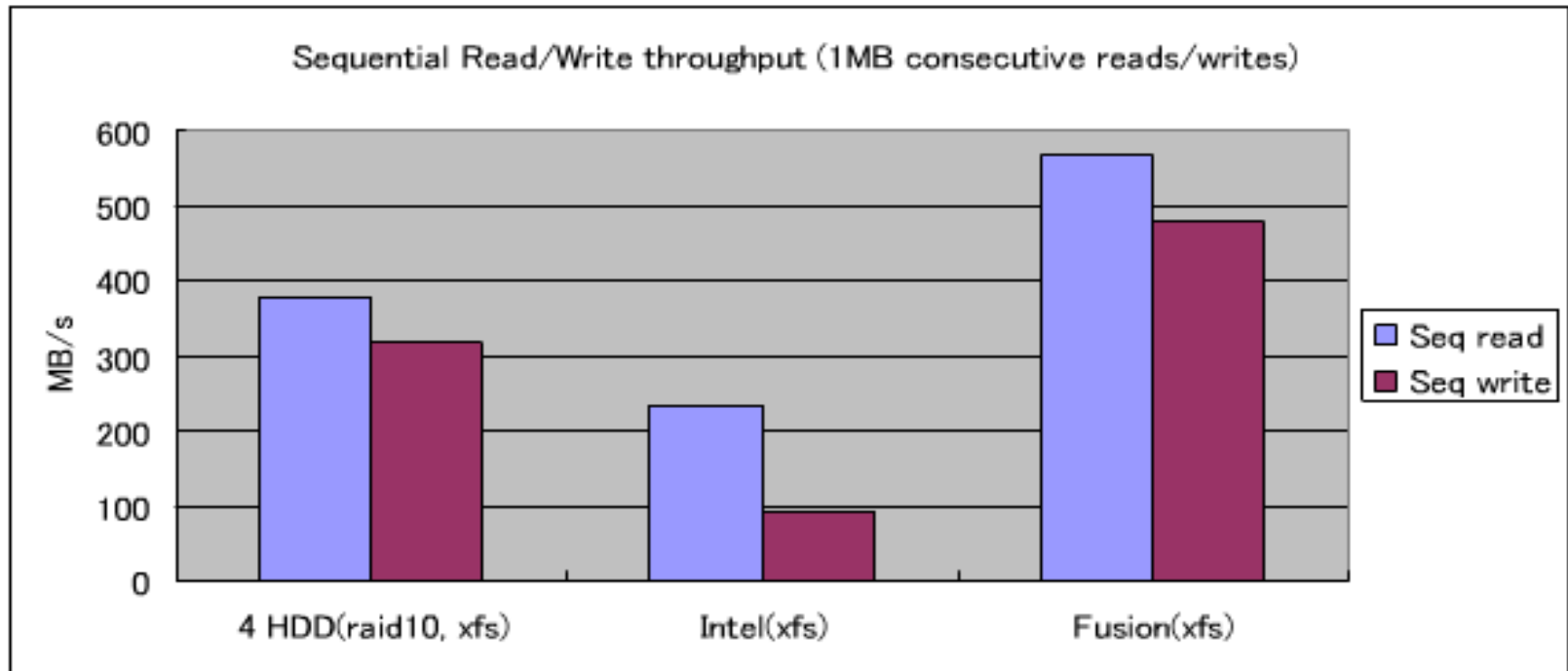
Write performance on SSD



- Very strong reason to use SSD

MySQL服务器新架构

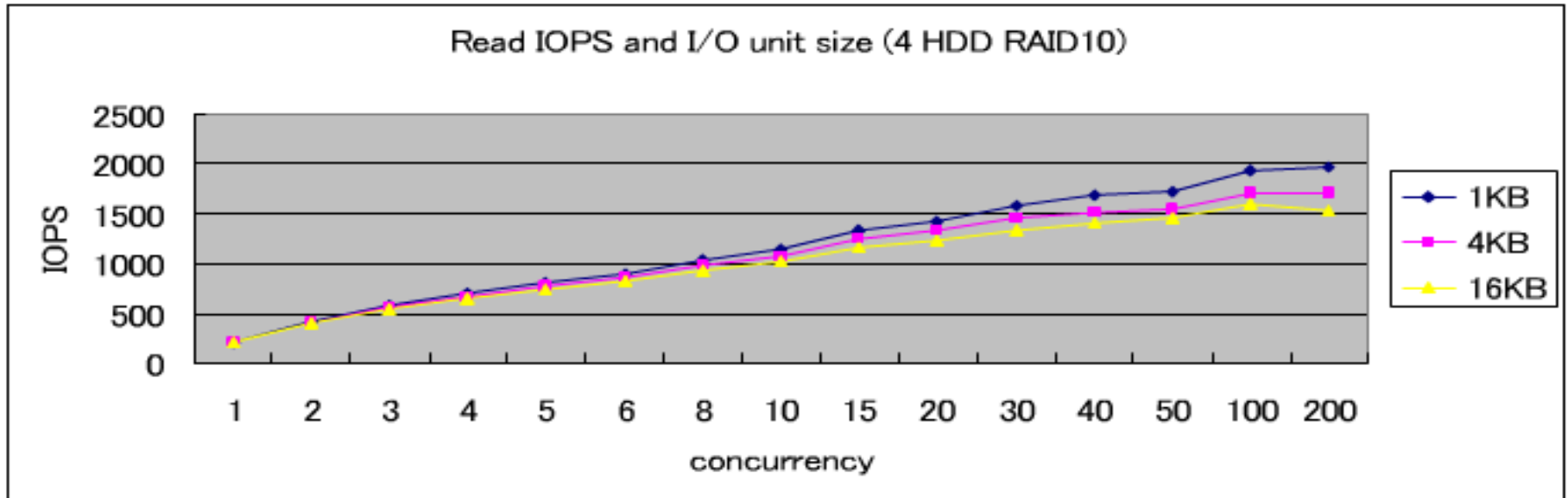
Sequential I/O



- 顺序IO场景:全表扫描,MySQL binlog, ibdata
- SAS盘的顺序写性能也不会太差

MySQL服务器新架构

Changing I/O unit size

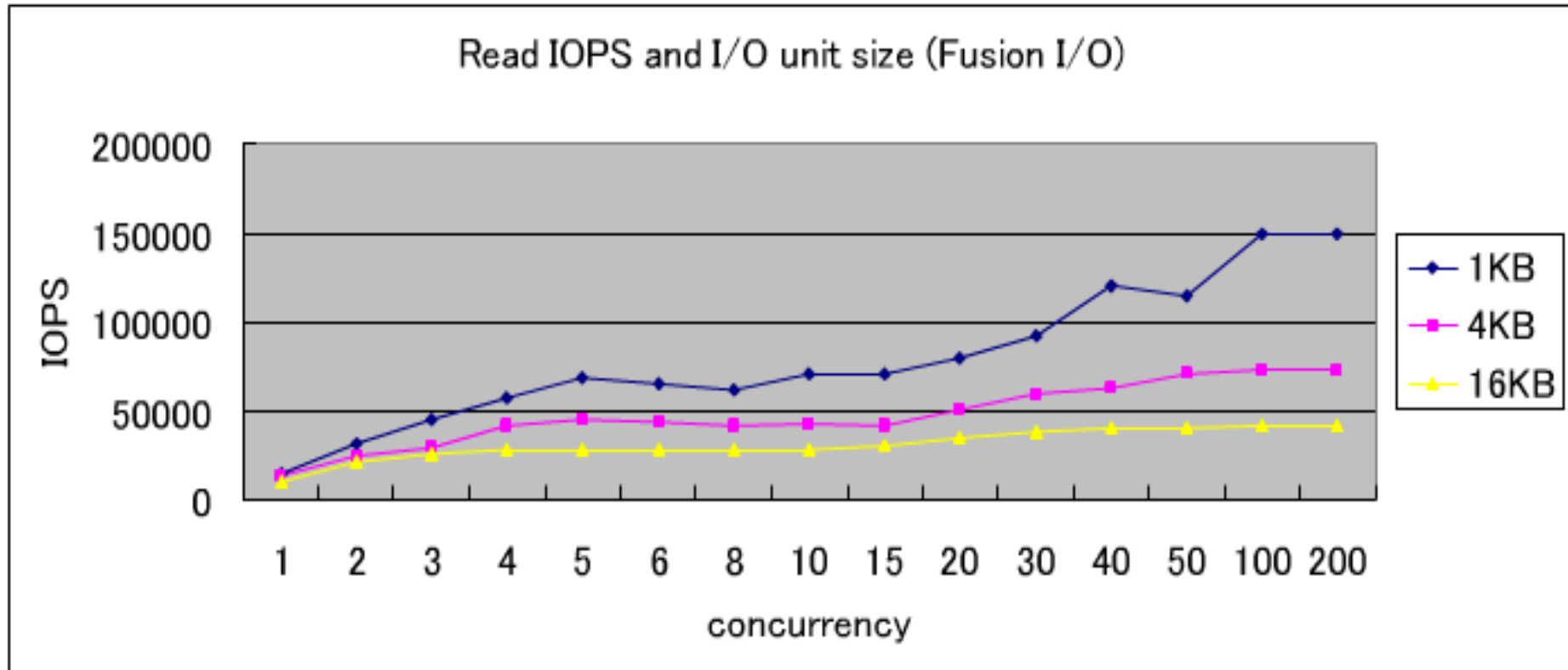


- On HDD, maximum 22% performance difference was found between 1KB and 16KB
- No big difference when concurrency < 10



MySQL服务器新架构

Changing I/O unit size on SSD



- Huge difference
- On SSDs, not only IOPS, but also I/O transfer size matters

第三个核心系统业务压测数据

配置	cpu 2*4c E5620 Mem 72G BP 56G Disk FIO 320G Data 166G 12 sas
结果	●压力： QPS 26000 TPS 1630 ●Sas iops read 120 write 900

- %user 45% %iowait 8.20
- BP hit 99.3% flashcache hit 98.2%
- /proc/flashcache_setutil 79441(2MB)↑util99%





MySQL新技术在淘宝的使用

三大核心系统MySQL集群概况

	A系统	B系统	C系统
存储	12块SAS盘 RAID10	Flashcache+320G FusionIO+SAS RAID 10	Intel SSD+SAS RAID10
网卡	千兆网卡	千兆网卡	千兆网卡
内存	48G	96G	96G
CPU	24xIntel(R) Xeon(R)	24xIntel(R) Xeon(R) CPU X5670	24xIntel(R) Xeon(R) CPU
单库大小	140G	210G	160G
集群	16个库 8主8备 2套备库	16个库 16主16备 共两套备库	32个库16主16备 一主两备
双十一	双机房共20台tair , 单台最高QPS 6w/s, 命中率100% DB 单台QPS单台 最高1000/s	单台DBQPS 20000,响应时间 0.3ms左右 网络流量从250Mbit/s增加到高峰400Mbit/s	主库单台 最高QPS 1W/s TPS 最高2K/s
风险点	?	?	?
改进	?	?	?

三大核心系统风险点+措施

- **A系统**

- 极度依赖Tair，Tair目前无法实现在线更换
- Tair如果挂掉，DB在高峰期间直接被秒杀
- 明年Q1前完成 DB硬件升级

- **B系统**

- 网卡流量在高峰期只有55%的余量
- 减库存单条同时高并发更新,导致死锁的问题通过业务来避免
- 直接升万兆网卡？

未来淘宝MySQL的走向猜测

- SSD让MySQL服务器的性能大幅提升，对比其他NoSQL方案，不再黯然
- 单台服务器上面会跑多个MySQL实例，3306,3406,3506...
- IOPS从马车时代进入到火车时代
- MySQL可以进行并发DDL





MySQL新技术在淘宝的使用

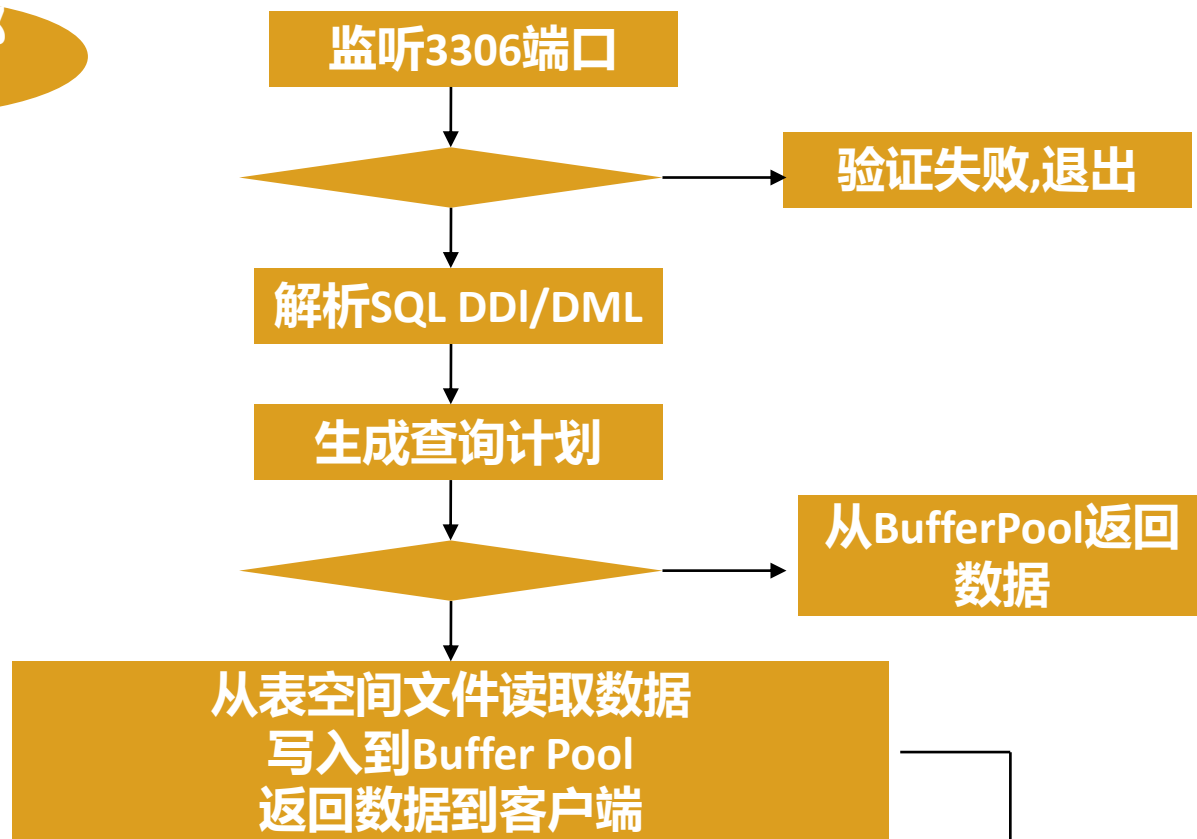
基于MYSQL的NOSQL方案 HANDLERSOCKET的使用介绍



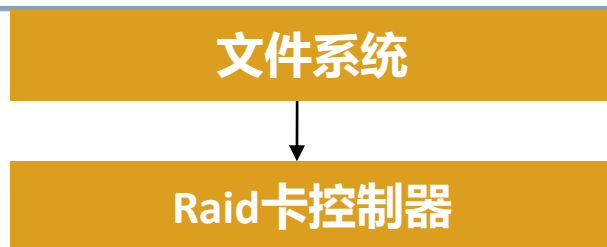
追風堂

MySQL原有查询流程

MySQL内部 流程



硬件



小结MySQL处理查询的流程

- **和获取数据无关的流程**

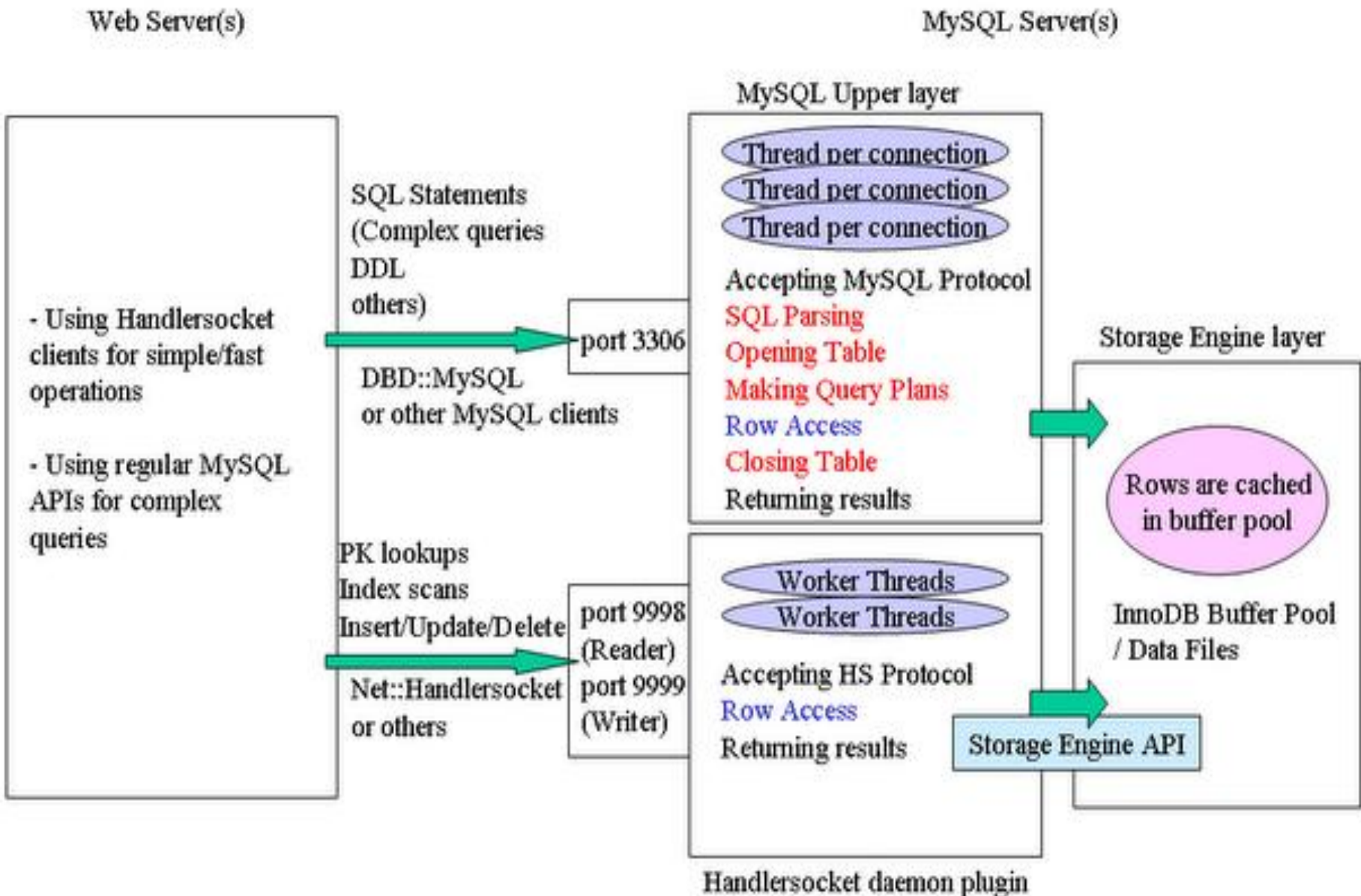
- 连接池
- 验证用户
- 解析SQL到底是DDL还是DML
- 生成查询计划

- **实际情况**

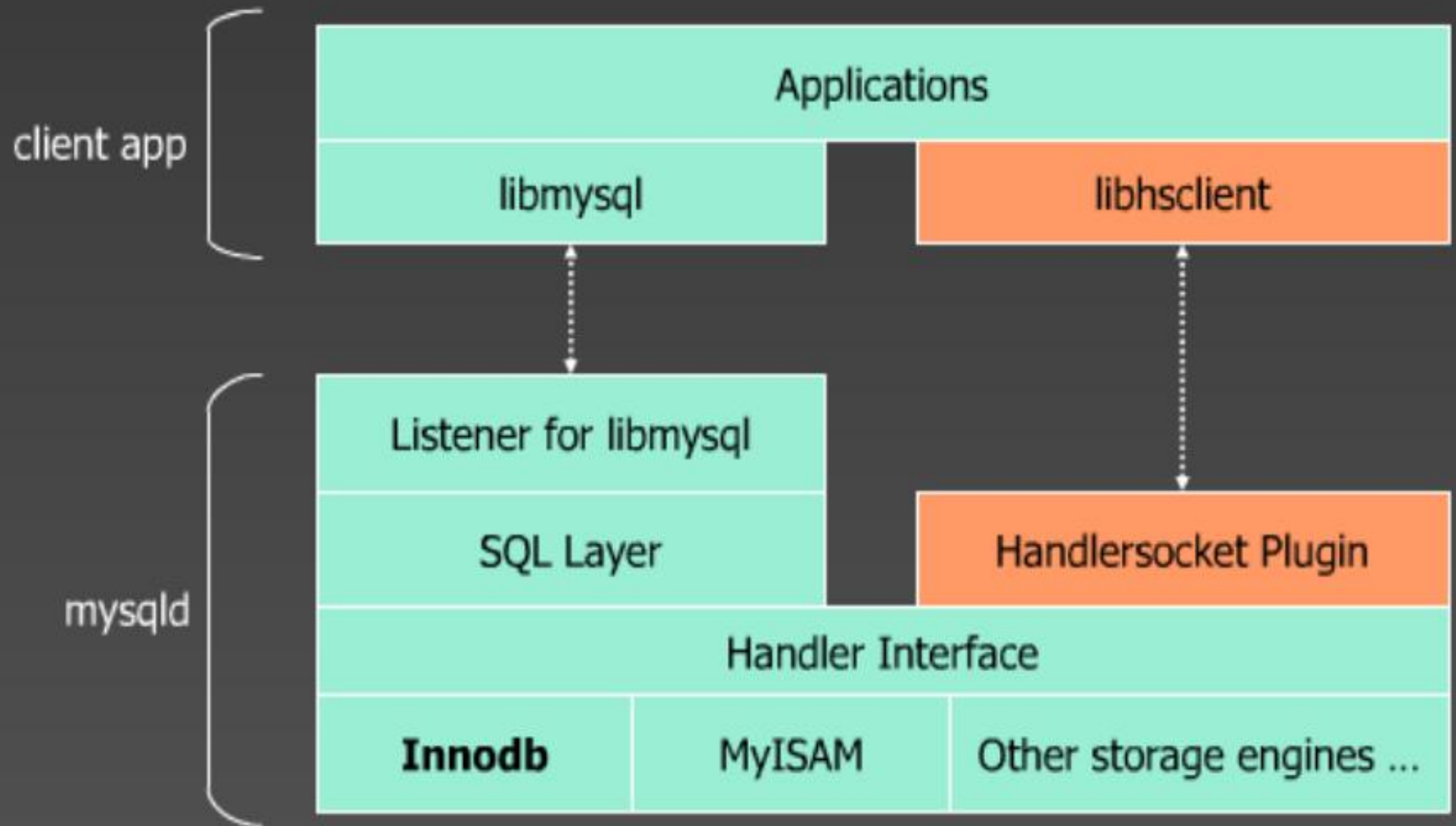
- 我们的SQL很多时候是key-value式的查询
- 我们只想尽快拿到想要的数据库
- 如何在不升级硬件的前提下提高QPS/TPS?



HandlerSocket架构图



简化的HS架构图

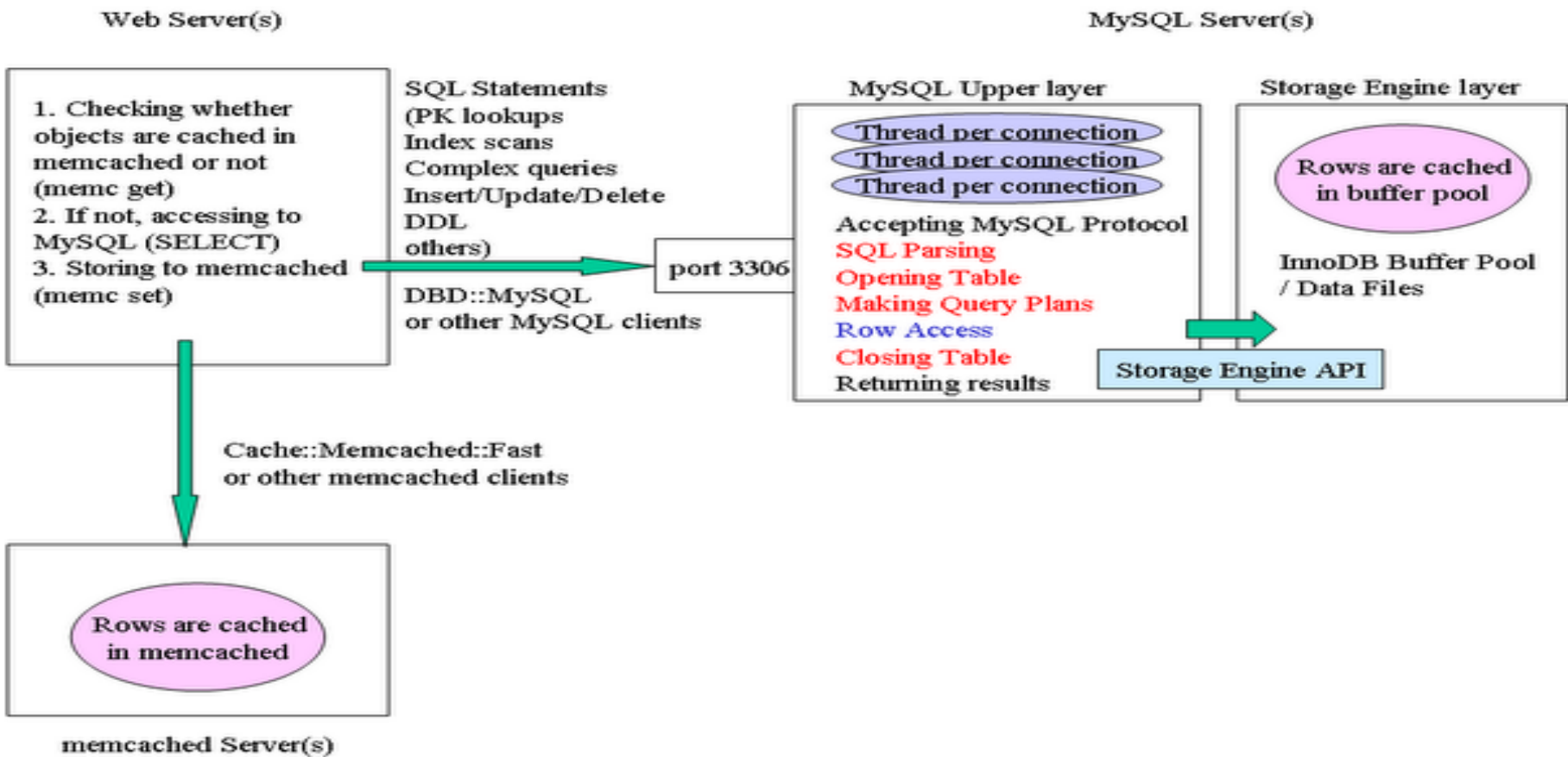


小结HS

- HS绕开了MySQL内部验证流程,不做SQL Parsing,不做查询计划
- HS使用自己的验证流程(my.cnf配置用户密码)
- HS打开表后不会立即关闭,会独占表锁,这样可以减少因为频繁打开关闭表带来的性能耗损。
- 做DDL的时候要在低峰期或者App控制连接数量



HS与MySQL+Memcache/Tair的区别



- 若是HandlerSocket的实现更加完美，我们就可以考虑替换Memcached/Tair缓存记录的架构层

3306端口插入数据-HandlerSocket测试报告

- 使用sysbench 初始化数据
- `$ time ./sysbench --test=oltp --mysql-table-engine=innodb --oltp-table-size=10000000 --mysql-user=root --mysql-socket=/tmp/mysql.sock --mysql-db=test prepare`
- sysbench 0.4.12: multi-threaded system evaluation benchmark
- No DB drivers specified, using mysql
- Creating table 'sbtest'...
- Creating 10000000 records in table 'sbtest'...
-
- sysbench 初始化数据结束
- real 2m14.448s
- user 0m1.949s
- sys 0m0.196s
- Oprofile信息



Oprofile信息

CPU: CPU with timer interrupt, speed 0 MHz (estimated)

Profiling through timer interrupt

samples	%	image name	symbol name
10451	8.1058	mysqld	MYSQLparse(void*)
8926	6.9230	mysqld	log_group_write_buf
7606	5.8992	mysqld	buf_calc_page_new_checksum
4382	3.3987	mysqld	mtr_commit
4313	3.3451	mysqld	btr_search_guess_on_hash
4311	3.3436	libc-2.5.so	memcpy
4259	3.3033	mysqld	row_mysql_store_col_in_innabase_format
4155	3.2226	mysqld	get_text(Lex_input_stream*, int, int)
3823	2.9651	mysqld	rec_get_offsets_func
3702	2.8713	mysqld	btr_cur_optimistic_insert
2911	2.2578	mysqld	lex_one_token(void*, void*)
2489	1.9305	mysqld	mlog_open_and_write_index
2176	1.6877	mysqld	trx_undo_report_row_operation
1959	1.5194	mysqld	row_ins_step
1949	1.5116	mysqld	buf_page_get_gen
1890	1.4659	mysqld	ismbchar_gbk
1839	1.4263	mysqld	rec_get_converted_size_comp
1799	1.3953	mysqld	fill_record_n_invoke_before_triggers(THD*, List<Item>&, List<Item>&, b
1701	1.3193	mysqld	page_cur_insert_rec_write_log
1700	1.3185	mysqld	cmp_dtuple_rec_with_match
1622	1.2580	libc-2.5.so	memset
1598	1.2394	mysqld	my_numchars_mb
1568	1.2161	mysqld	btr_search_update_hash_node_on_insert
1551	1.2030	mysqld	mtr_memo_slot_release
1484	1.1510	libc-2.5.so	_int_malloc
1479	1.1471	libc-2.5.so	_int_free
1461	1.1331	mysqld	page_cur_insert_rec_low
1309	1.0153	mysqld	setup_fields(THD*, Item**, List<Item>&, enum_mark_columns, List<Item>*
1290	1.0005	mysqld	btr_cur_search_to_nth_level
1278	0.9912	mysqld	ha_innabase::write_row(unsigned char*)
1234	0.9571	mysqld	conv_and_convert(char*, unsigned int, charset_info st*, char const*

HS-使用perl单线程初始化1000w数据

CPU: CPU with timer interrupt, speed 0 MHz (estimated)

Profiling through timer interrupt

samples	%	image name	symbol name
18679	7.8554	mysqld	row_search_for_mysql
12000	5.0466	mysqld	rec_get_offsets_func
9959	4.1882	mysqld	btr_search_guess_on_hash
6985	2.9375	mysqld	buf_page_get_gen
6280	2.6410	mysqld	my_hash_sort_utf8
6170	2.5948	mysqld	buf_page_get_known_nowait
5514	2.3189	mysqld	log_group_write_buf
4833	2.0325	mysqld	ha_innobase::general_fetch(unsigned char*, unsigned int, unsigned
3540	1.4887	libc-2.5.so	memchr
3303	1.3891	mysqld	btr_cur_optimistic_insert
3293	1.3849	mysqld	mtr_commit
2973	1.2503	libc-2.5.so	_int_malloc
2846	1.1969	libc-2.5.so	memcpy
2823	1.1872	mysqld	pfs_mutex_enter_func
2779	1.1687	mysqld	lock_sec_rec_cons_read_sees
2719	1.1435	mysqld	evaluate_join_record(JOIN*, st_join_table*, int)
2691	1.1317	mysqld	trx_undo_assign_undo
2665	1.1208	mysqld	buf_calc_page_new_checksum
2628	1.1052	mysqld	row_ins_step
2595	1.0913	mysqld	end_send_group(JOIN*, st_join_table*, bool)
2567	1.0795	mysqld	cmp_dtuple_rec_with_match
2469	1.0383	mysqld	btr_cur_search_to_nth_level
2422	1.0186	mysqld	buf_block_align
2371	0.9971	mysqld	row_ins_index_entry_low

性能对比

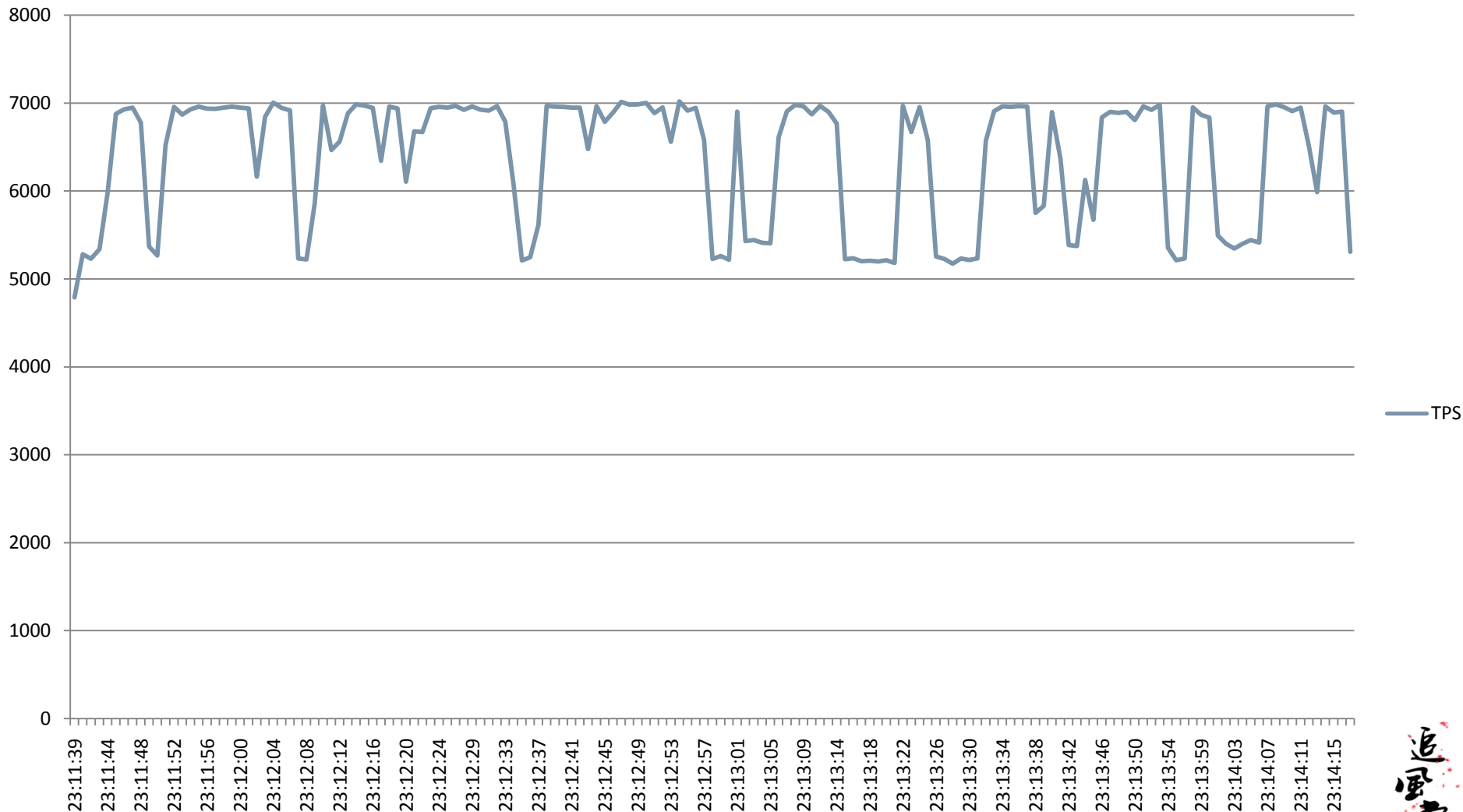
- 3306端口写入到InnoDB表 100w数据 单线程
- HS 9998端口写入到InnoDB表 100w数据 单线程



3306端口写入

InnoDB 单表 单线程 100w Insert

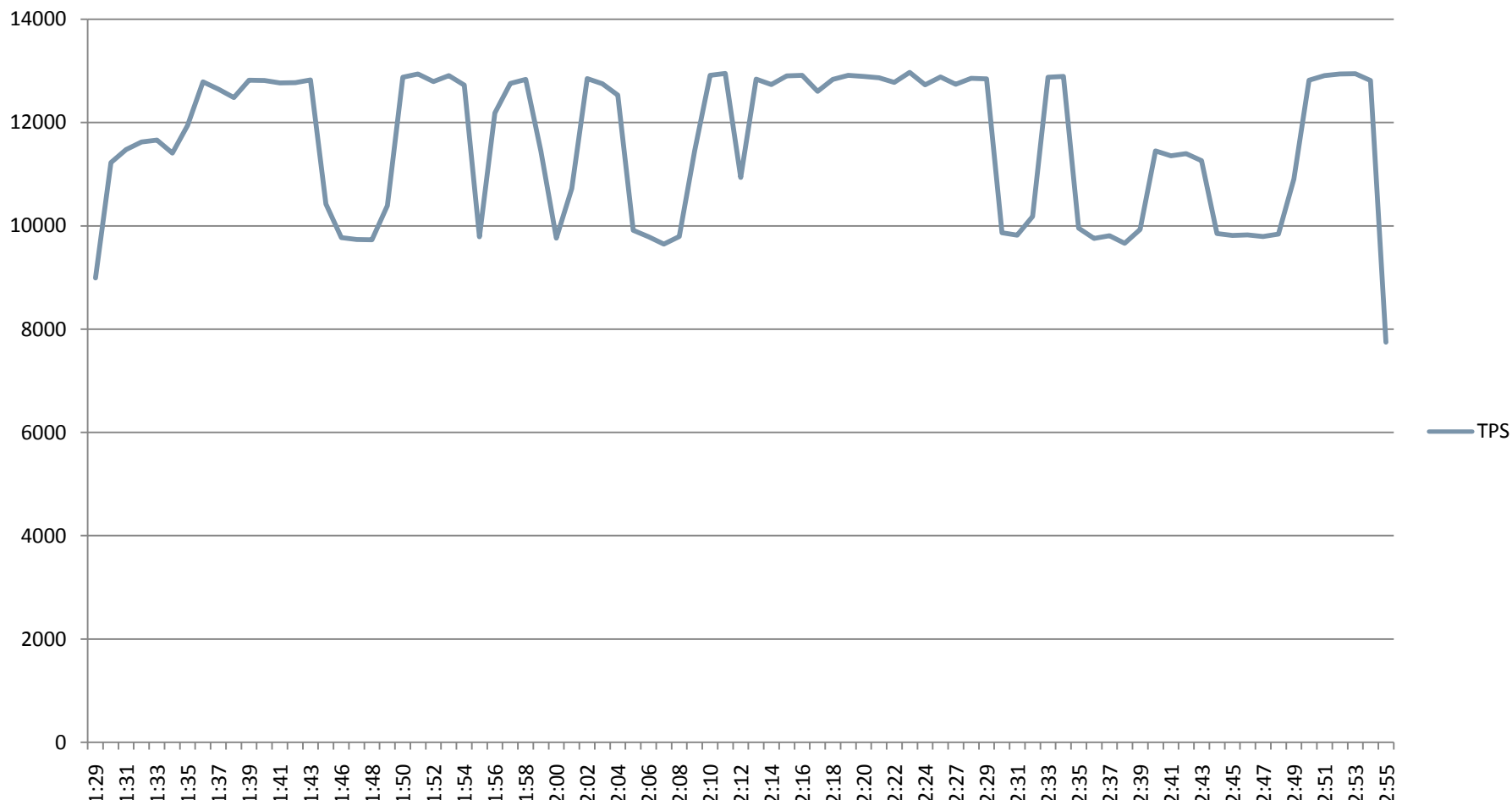
innodb_adaptive_flushing_method=keep_average



HS 9998端口写入

HS单线程顺序插入100W数据到InnoDB表

innodb_adaptive_flushing_method= estimate



时间对比

- **3306 InnoDB单表单线程写入100w数据消耗时间**

- time ./mysql_perl.pl
- real 2m34.074s
- user 0m9.796s
- sys 0m2.363s

- **9998 HS 端口 写入100w数据 单线程**

- time ./insert_hs.pl
- real 1m26.516s
- user 0m20.445s
- sys 0m31.359s



HS使用小结

- 分库分表的时候，如果表带有auto_increment的id，并发写入的时候，会导致hs plugin处理自增字段出错
- HS会一直拿住表锁，DDL会被堵塞，要想HS释放表锁，还得通过app控制
- 某系统的4台hs 凌晨4点多出现swap非常严重的情况，还导致的mysql重启，原因是Java应用没有重用table cache
- HandlerSocket思路是有价值的，但还需要时间去完善



- 简单的CURD语句
- 绕开了MySQL的SQL解析层
- 简单的事情用简单的工具来做
- 性能..性能..性能..
- 适用场景
 - 高并发读
 - 数据安全性不太重要的场景
 - 预热数据库

Percona VS MySQL

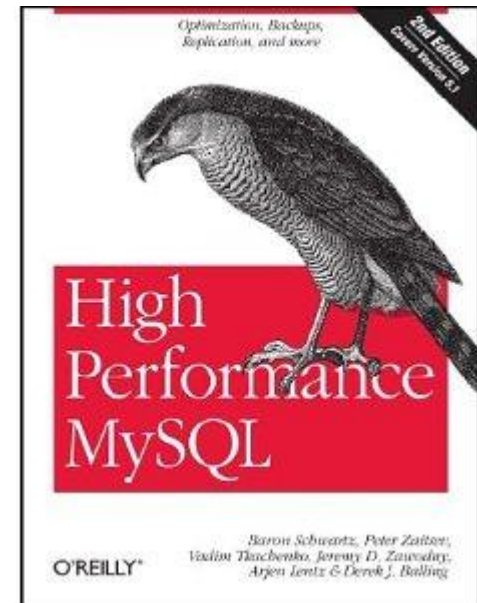


About Percona



Peter Zaitsev

Founder, CEO



O'REILLY
MySQL
Conference & Expo

MySQL Performance Blog

Percona's MySQL & InnoDB performance and scalability blog



MySQL Performance Blog

Percona's MySQL & InnoDB performance and scalability blog



Vadim Tkachenko

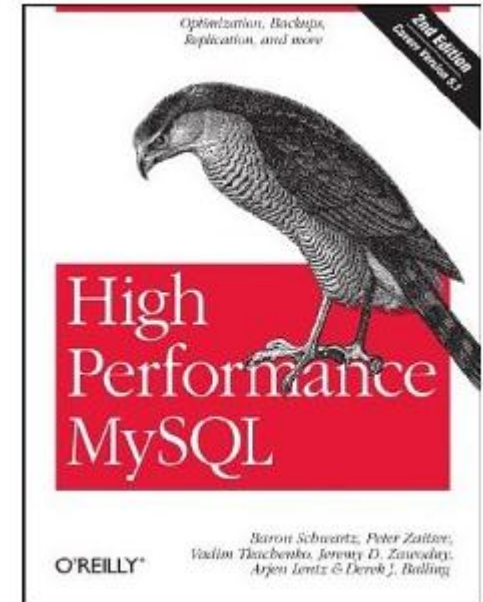
Co-Founder, CTO



SSD Performance Blog



About Percona



About Percona

- *Percona Server* is an enhanced drop-in replacement for *MySQL*.
- Percona Server 是MySQL的增强版的替代品

With *Percona Server*

- Your **queries will run faster and more consistently.**
- You will consolidate servers on powerful hardware.
- You will delay sharding, or avoid it entirely.
- You will save money on hosting fees and power.
- You will **spend less time tuning and administering.**
- You will achieve higher uptime.
- You will **troubleshoot without guesswork.**
















Percona5.5.8 VS MySQL 5.5.8

A high-resolution image of a red sports car, likely a Ferrari, shown from a front-three-quarter perspective. The car is dark red with black accents and is set against a dark, reflective background.

MySQL 5.5
性能和可伸缩性
正式发行
了解更多 »

Percona5.5.8 VS MySQL5.5.8

Extra Diagnostic Features	Percona Server 5.5.8	MySQL 5.5.8
PERFORMANCE_SCHEMA Diagnostics Tables		
INFORMATION_SCHEMA Tables	60	37
Global Performance and Status Counters	366	308
Per-Table Performance Counters		
Per-Index Performance Counters		
Per-User Performance Counters		
Per-Client Performance Counters		
High-Resolution Process List Timing		
Detailed Query Execution and Plan Log		
Global Query Response Time Statistics		
InnoDB Data Dictionary as I_S Tables		
Access to InnoDB Data Statistics		
Enhanced SHOW INNODB STATUS		
Enhanced Mutex Diagnostics		












Percona5.5.8 VS MySQL5.5.8

Performance & Scalability Enhancements	Percona Server 5.5.8	MySQL 5.5.8
Support for > 1024 Concurrent Transactions		
Support for Multiple I/O Threads		
Dedicated Purge Threads		
Self-Tuning Checkpoint Algorithm		
Fine-Grained Mutex Locking		
Lock-Free Algorithms		
Partitioned Adaptive Hash Search		
Separate Doublewrite File		
Fast Checksum Algorithm		
Buffer Pool Pre-Load		
Fast Shut-Down		
Support for FlashCache		
Read-Ahead Improvements		



Percona5.5.8 VS MySQL5.5.8

Extra Features for DBA/Operations Staff	Percona Server 5.5.8	MySQL 5.5.8
Configurable Page Sizes		
Shared-Memory Buffer Pool		
Import Tables From Different Servers		
Configurable Data Dictionary Size		
Configurable Insert Buffer Size		
Active Change Buffer Purging		
Error/Warning Logging Enhancements		
Configurable Fast Index Creation		
Fast Index Renaming		

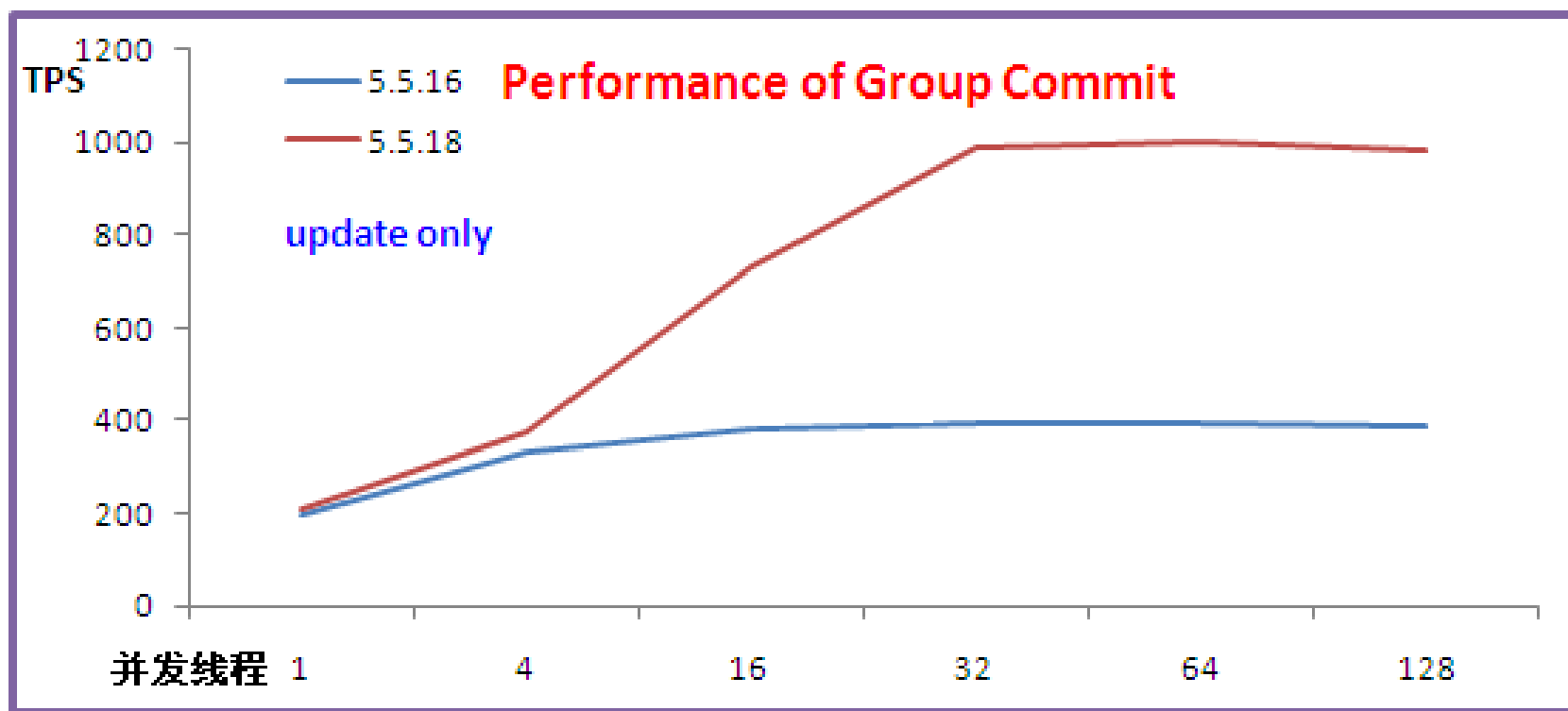


Percona5.5.18 Group Commit

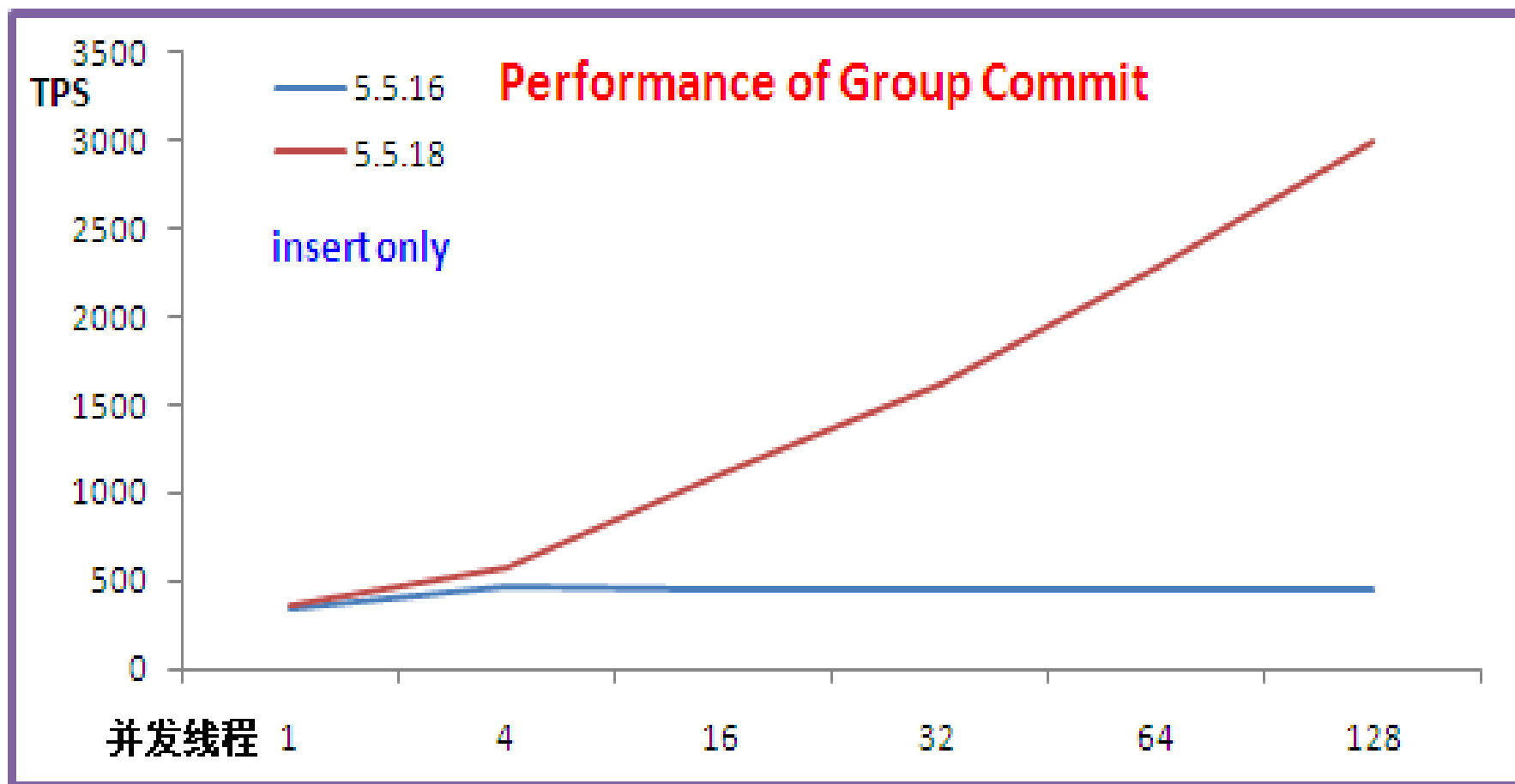
- 5.5.18开始实现了Binary Group Commit的特性，在数据安全级别更高的情况下，提供了更高的性能
- 测试数据集合大小约125G
- 测试SQL为全量数据中，随机读取。每条SQL按主键查询或操作。混合是比率Select:update:insert = 10:1:
- Supersmack
- `sync_binlog=1 innodb_flush_log_at_trx_commit = 1`



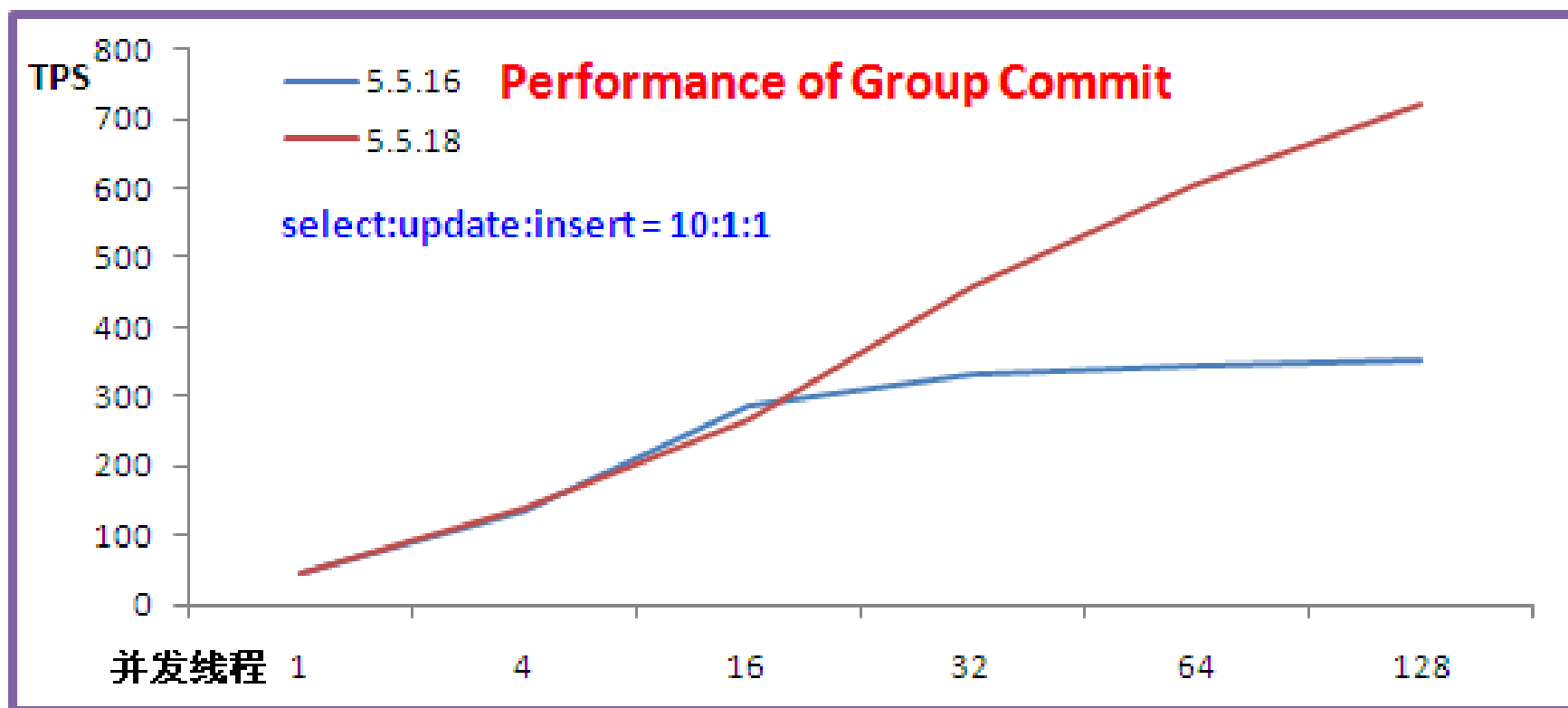
UPDATE



INSERT



混合场景



我们用了哪些Percona的相关产品

- Xtrabackup
- Percona Toolkit 2.0.1
 - ioprofile
 - tcprstat
 - pt-collect and others
- Percona5.5.18 webww , 2012Q1 UIC 集群
 - 更详细的日志信息
 - 更详细的slowlog信息
 - Group commit (提升TPS)
 - And others



性能/瓶颈/问题

- **新方案,难免遇到问题**
 - Percona5.5 kernel_mutex问题
- **Objdump -t 确认是否有符号信息**
- **Oprofile (MySQL编译加-g参数)**
- **Systemtap (kernel-debuginfo)**
- **gdb+poorman (不建议在线上使用)**
- **Pstack (不建议在线上使用)**
- **Tcprstat (计算DB端的平均响应时间)**
- **perf top (redhat 6u+ 2.6.32 kernel自带)**



MySQL新技术在淘宝的使用

谢谢大家



追風堂