



2014中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2014



大数据技术探索和价值发现

数据库的优化与调优：从理论到实践

网易杭研——何登成



自我介绍

- 何登成
 - 网易——杭州研究院；
- 工作领域
 - 数据库引擎/分布式数据库/分布式KV
- 技术领域
 - 数据库/分布式/并发编程/性能优化
- 联系方式
 - 微博：[何_登成](#)
 - 博客：[何登成的技术博客](#)

Outline

- 数据库性能优化与调优：从理论到实践
 - 理论篇
 - 从串行到并行
 - 从畅通无阻到排队
 - 必须了解的硬件知识
 - 实践篇
 - MySQL各版本的优化
 - MySQL使用中的调优

理论篇

- 理论篇

- 从串行到并行

- 欢迎来到并行世界:
 - 并行世界, 串行的副作用:

Moore's Law

Amdahl's law

- 从畅通无阻到排队

- 并发的世界, 充满排队:

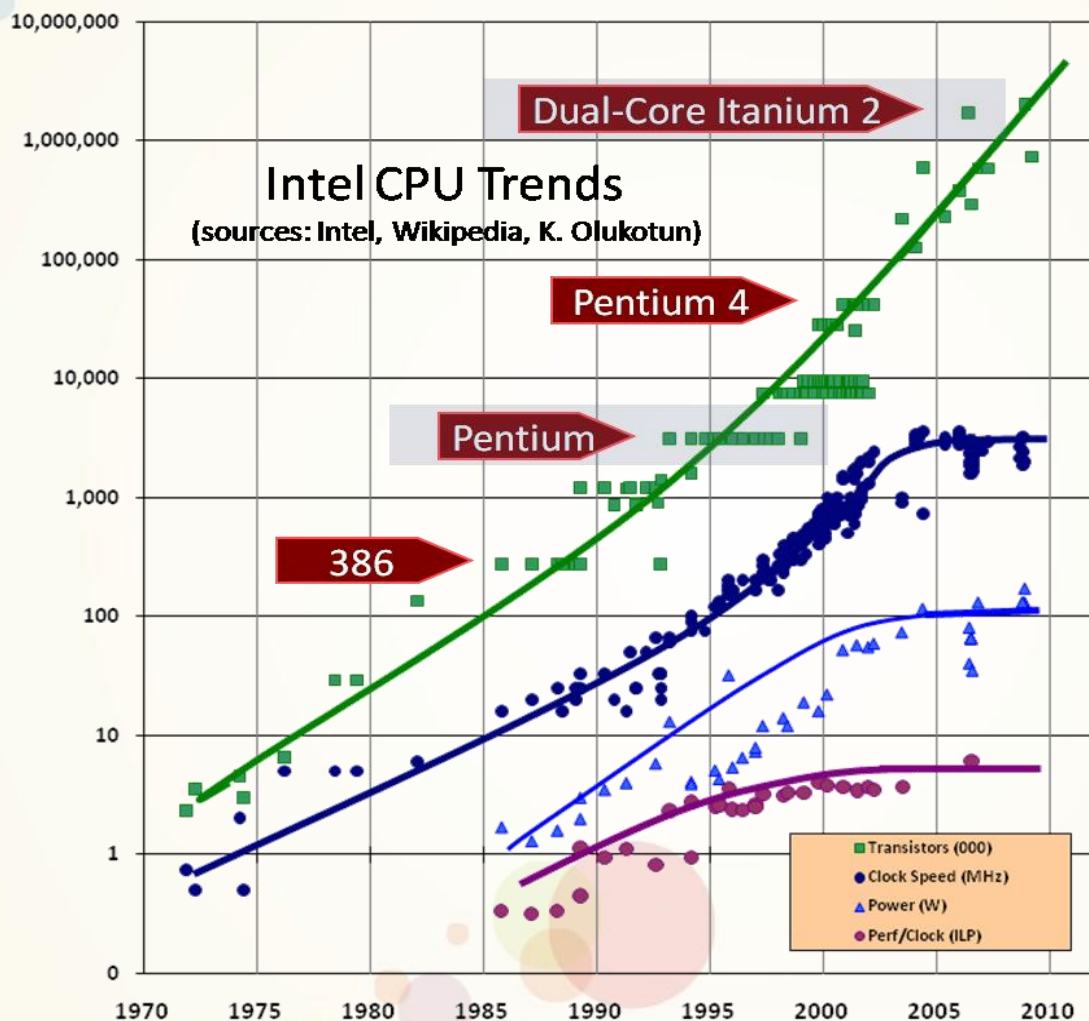
Queueing Theory

- 必须了解的硬件知识

- CPU/Cache/Memory/Disk ...

欢迎来到并行世界

- 摩尔定律(Moore's Law)失效了？
 - No，只是转变了形式而已。
- Herb Sutter
 - [The Free Lunch Is Over – A Fundamental Turn Toward Concurrency in Software.](#)
- 何谓Free Lunch？
 - 通过CPU主频提升，软件能够自动提升性能的时代已经一去不复返；
 - 是时候考虑多线程/并发程序了；

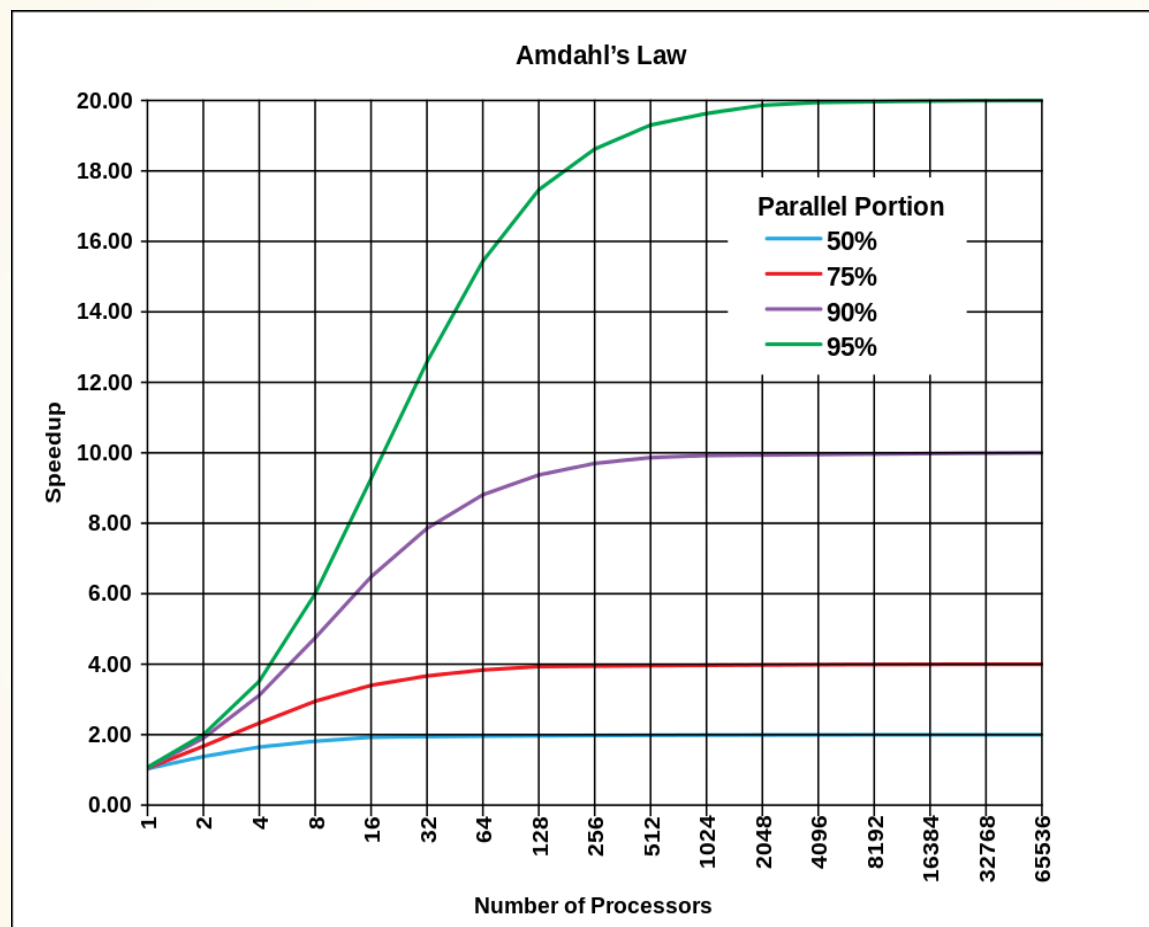


并行世界，串行的副作用

- Gene Amdahl怎么说
([Amdahl's law](#))?

$$S(N) = \frac{1}{(1 - P) + \frac{P}{N}}$$

- P: 程序可并行执行比率
- 1-P: 串行比率
- N: N个Processors
- S(N): 加速比
- 解读: 程序的极限性能,
最终取决于程序串行部分
所占比率;
- 尽量消除程序中的串行部
分;



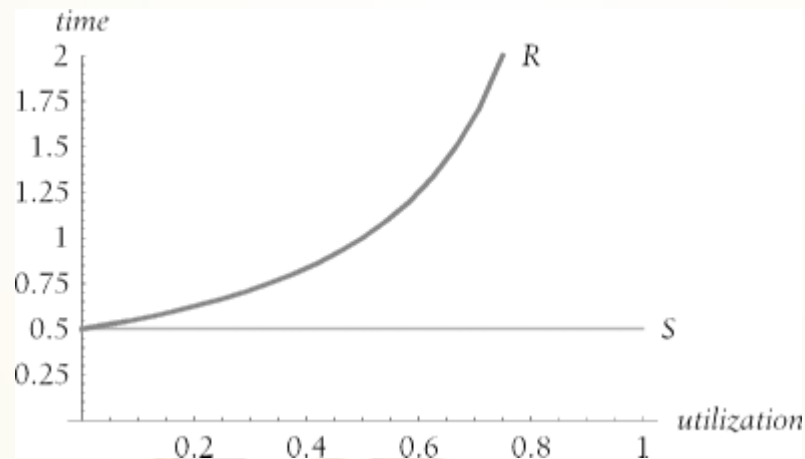
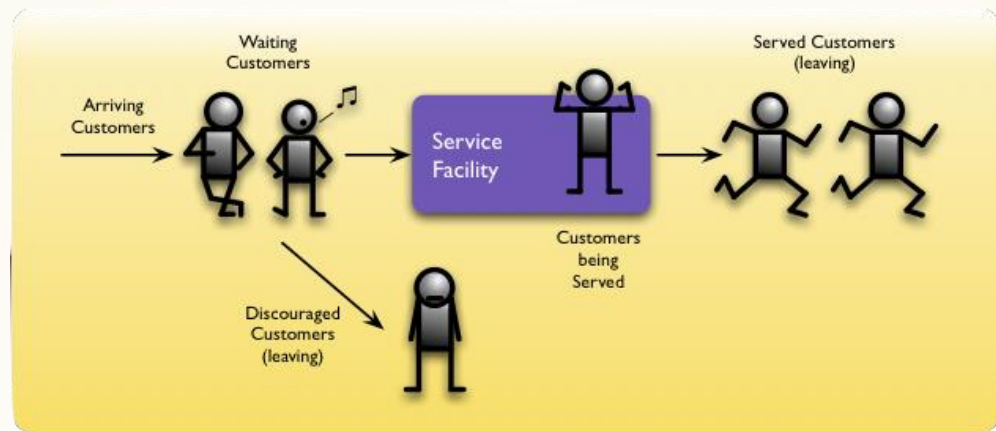
无限制的并行，您无法达到

- 现实生活中



无限制的并行，您无法达到(续)

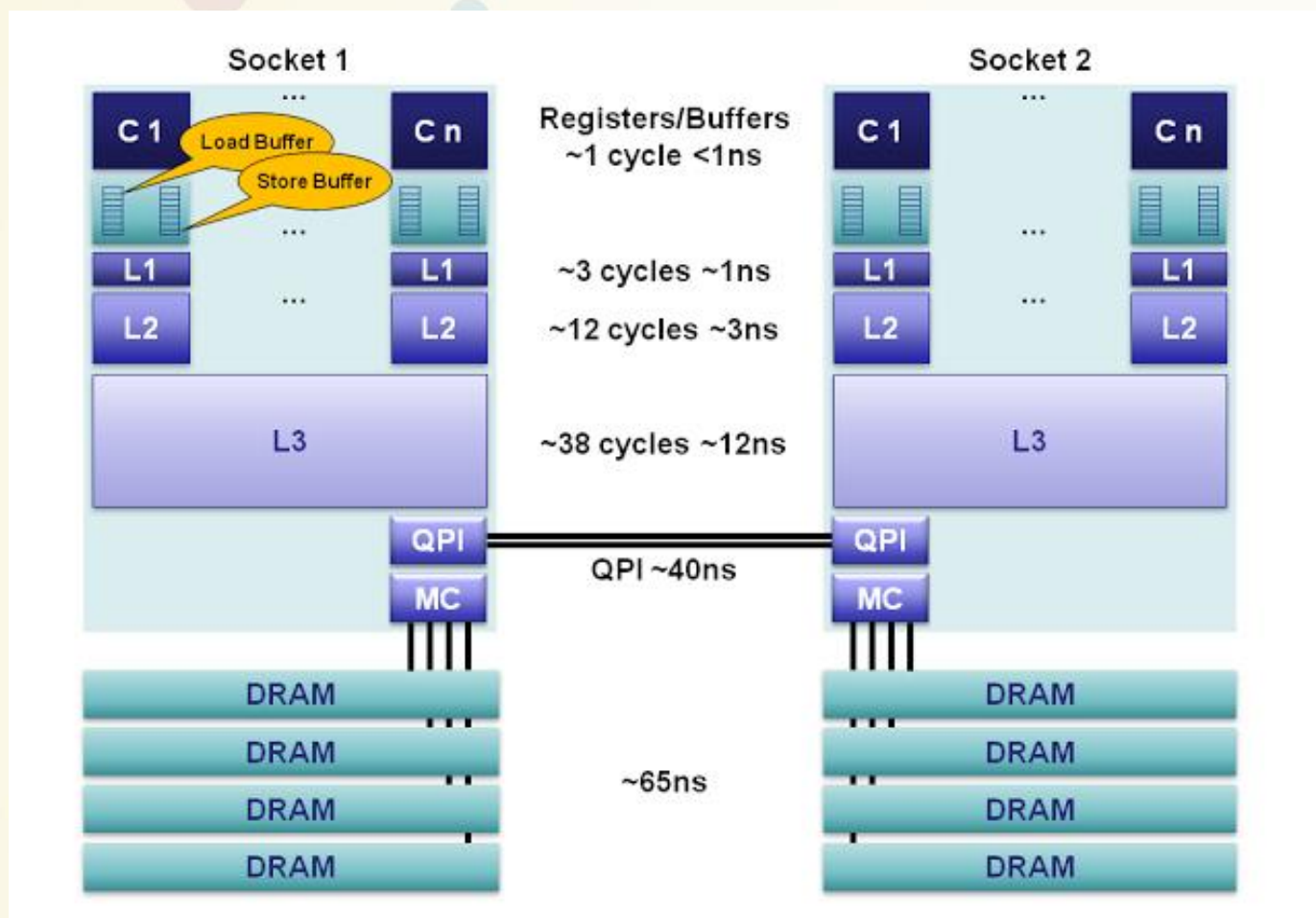
- 硬件资源瓶颈/软件内部等待
 - CPU/Memory/Disk...
 - Mutex/Spinlock/...
- 排队论([Queueing Theory](#))
 - 资源利用率;
 - 响应时间/等待时间;
 - 吞吐量;
 - 这些指标，均与性能相关;
- [Little's Law](#)
$$L = \lambda W$$
 - L: 平均队列长度;
 - Lambda: 平均吞吐量;
 - W: 平均响应时间;



写程序，为什么需要了解硬件？

- 看看他们怎么说？
 - **Hardware and software working together in harmony.**
 - [Martin Thompson](#)
 - Know Hardware to Design Better Software.
 - 未找到出处，暂时算我说的吧☺
- 硬件与软件性能息息相关
 - 硬件各组件的Latency和Throughput;
 - 语言那些与硬件相关的特性;

当前的硬件发展到了什么阶段？



- 注：来自Martin Thompson [CPU Cache Flushing Fallacy](#) 一文；

硬件各组件Latency对比

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access (DRAM, from CPU)	120 ns	6 min
Solid-state disk I/O (flash memory)	50–150 µs	2–6 days
Rotational disk I/O	1–10 ms	1–12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Internet: San Francisco to Australia	183 ms	19 years
TCP packet retransmit	1–3 s	105–317 years
OS virtualization system reboot	4 s	423 years
SCSI command time-out	30 s	3 millennia
Hardware (HW) virtualization system reboot	40 s	4 millennia
Physical system reboot	5 m	32 millennia

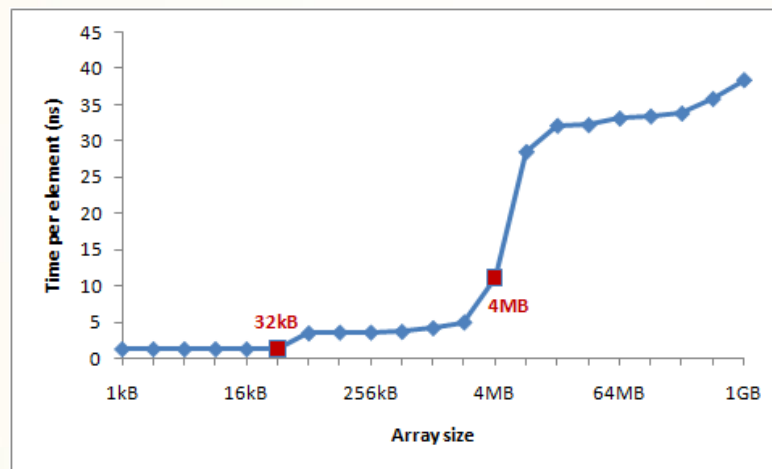
注：来自Gregg Brendan 《Systems Performance: Enterprise and the Cloud》一书；

授人以鱼不如授人以渔

- 你应该亲自测试这些硬件性能指标

- Igor Ostrovsky — [Gallery of Processor Cache Effects](#)

- 测试内存Latency;
 - 测试Cache Lines大小;
 - 测试L1/L2/L3 Cache大小;
 - ...



- [ccBench](#) — [Everything You Always Wanted to Know about Synchronization but Were Afraid to Ask](#) (SOSP' 13)

- 小工具，可以测试CPU Cache/Cache Coherence/Atomic Operations性能

理论结束，该来点实战了

- 吐槽时间

- 前面说了这么多，真的跟MySQL数据库的优化与调优有关系吗？你不是在忽悠吧？
- 接下来，让我们通过实践篇，来看看前面的这些理论，在MySQL数据库中是如何得到实践的。

数据库的优化与调优：实践篇

- 实践篇
 - MySQL各版本的优化
 - 基础优化：串行 → 并行
 - 进阶优化：减少系统中的串行点
 - 高级优化：软硬件相辅相成
 - MySQL使用中的调优
 - 资源利用率没到100%
 - 认识MySQL，构建平衡系统

基础优化-拿串行开刀(一)

- 臭名远播的[prepare_commit_mutex](#)
 - 目的：保持事务在InnoDB存储引擎与MySQL Binlog中提交顺序的一致性；
 - InnoDB Prepare：持有此Mutex
 - MySQL Binlog Commit；
 - InnoDB Commit：释放此Mutex
 - 事务串行化提交；
- MySQL Group Commit
 - [Oracle MySQL Group Commit](#)
 - Starting with MySQL 5.6
 - [MariaDB Group Commit](#)
 - [Starting with MariaDB 5.3](#)
 - How?

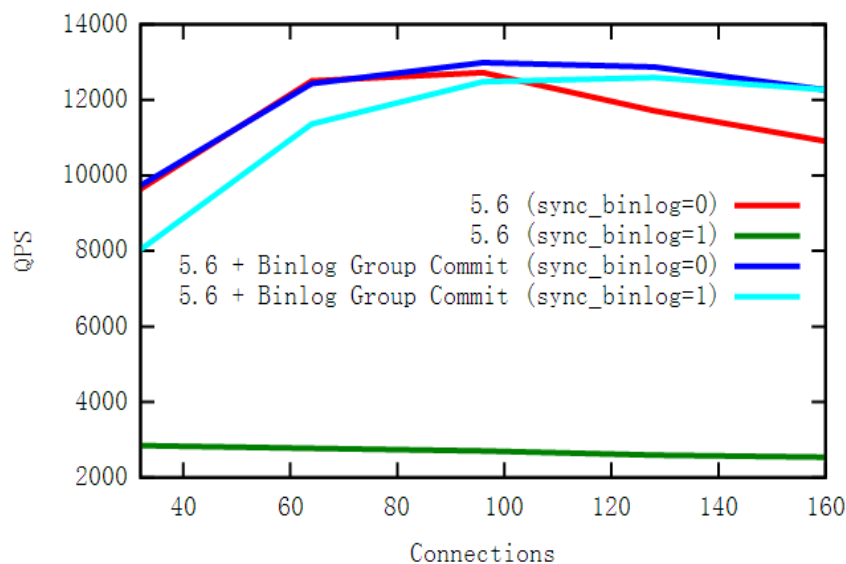


Figure 7. Benchmark of Binary Log Group Commit

基础优化-拿串行开刀(二)

- 你的Slave延迟了多久？

- MySQL Replication

- 基于Binlog的复制

- 两个线程

- I/O Thread: 负责与Master通讯，接收Binlog
 - SQL Thread: 负责Binlog在Slave的回放，单线程

- Parallel Slave Replication

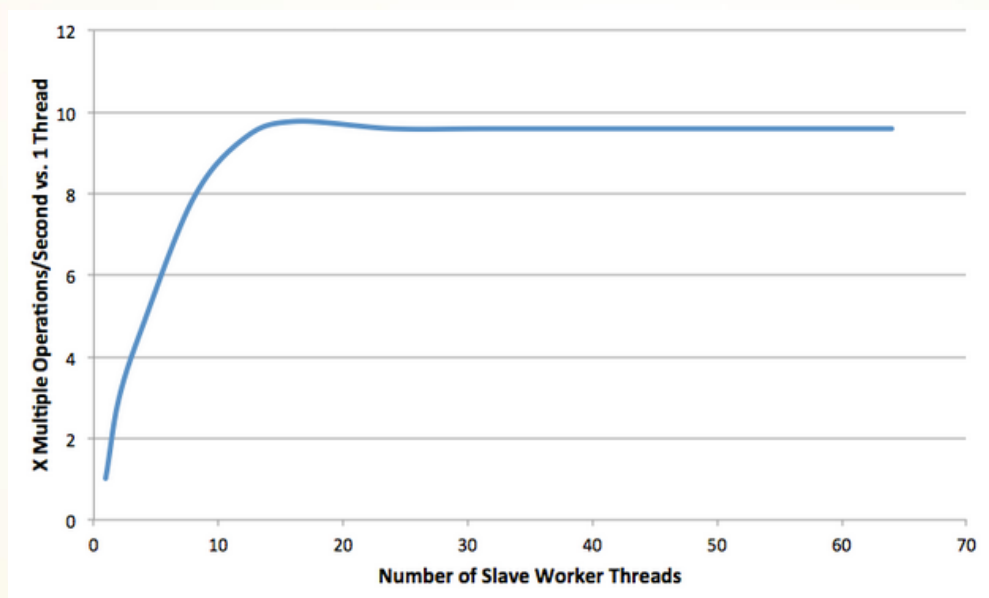
- Oracle MySQL

- [Starting with MySQL 5.7.2](#)

- [MariaDB Implementation](#)

- [Starting with MariaDB 10.0.5](#)

- 效果(见右图)



进阶优化-减少系统串行点(一)

- [kernel_mutex](#)

- InnoDB引擎内部有很多共享资源，如：内存分配，日志系统，事务与锁表，ReadView...
- 早期InnoDB版本，所有的这些共享资源，
- kernel_mutex，针对MySQL/InnoDB系统来

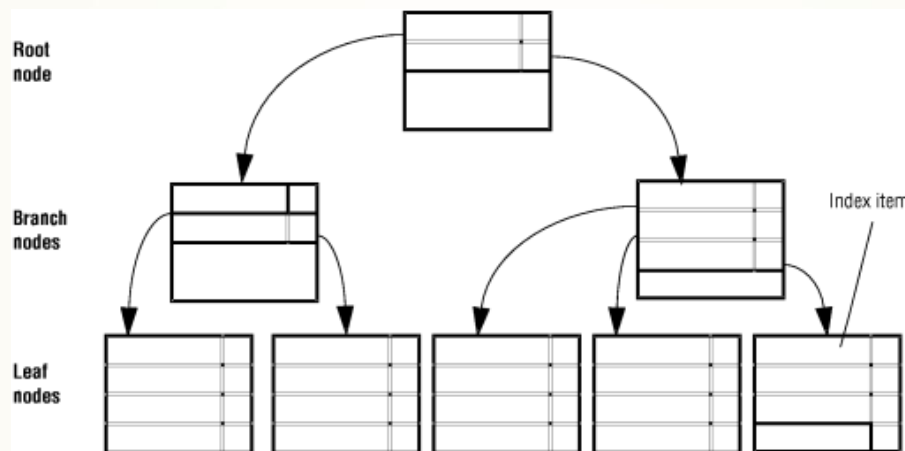
- 一字真言

- 拆!!!
- [Kernel mutex removed](#)
- 事务 & ReadView: `trx_sys_t::lock`
- 锁表: `lock_sys_t::mutex`
- ...



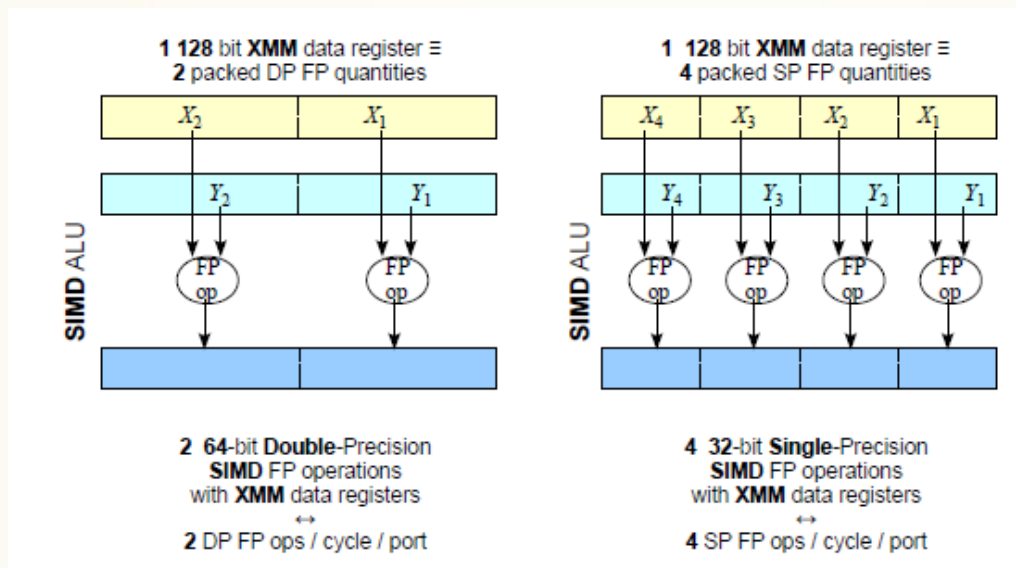
进阶优化-减少系统串行点(二)

- Index Tree Lock
 - InnoDB每一个Index，有一把RWLock(读写锁);
 - 读/写，加S锁;
 - 索引页面分裂/合并，加X锁;
 - 索引页面分裂时，整个索引树不可访问 ☹
- 二字真言
 - 算法！！
 - C. Mohan, [Aries/IM](#)
 - SMO操作，只跟SMO本身冲突，不会堵塞用户的读写;



高级优化-软硬件相辅相成(一)

- 高级CPU指令
 - [SIMD SSE 4](#)
 - Single-Instruction-Multiple-Data
- 合适的应用场景
 - InnoDB页面计算Checksum;
 - 硬件指令加速;



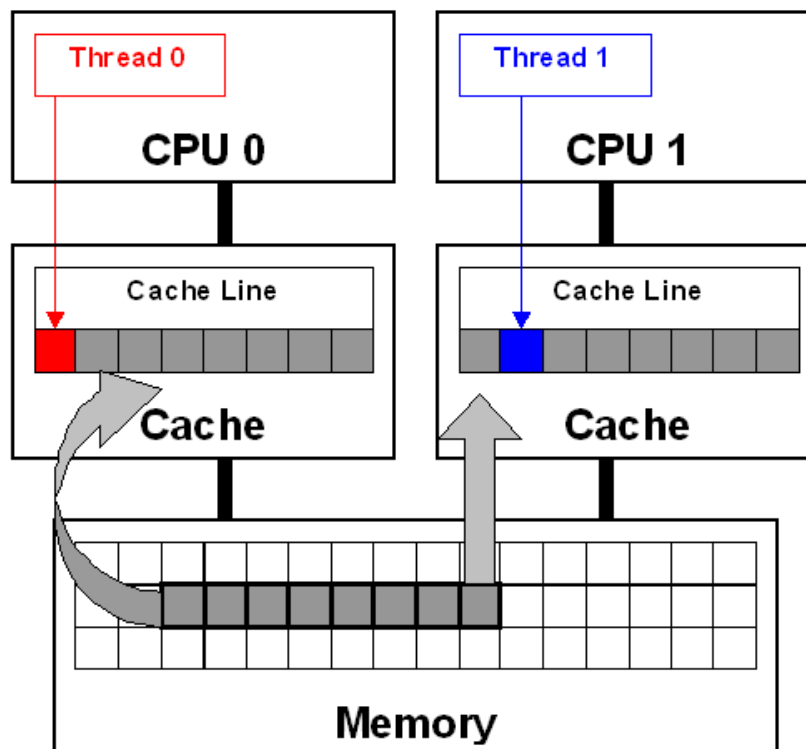
高级优化-软硬件相辅相成(二)

- [Cache Coherence](#) & False Sharing

- Cache Coherence
 - 多线程并发读写同一内存;
- False Sharing
 - 64 Bytes Cache Line

- [G5 Patch](#)

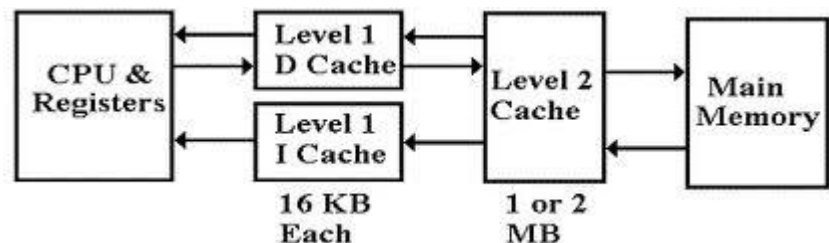
- 问题起因
 - `srv_n_rows_read++`
- Per-Thread Counter
 - 减少Cache Coherence
- Counter Padding
 - 消除False Sharing
 - `ib_counter_t::m_counter[(N + 1) * (CACHE_LINE_SIZE) / sizeof(Type)]`



高级优化-软硬件相辅相成(三)

- Compile Optimization

- CPU执行
- L1 ICache高速指令缓存



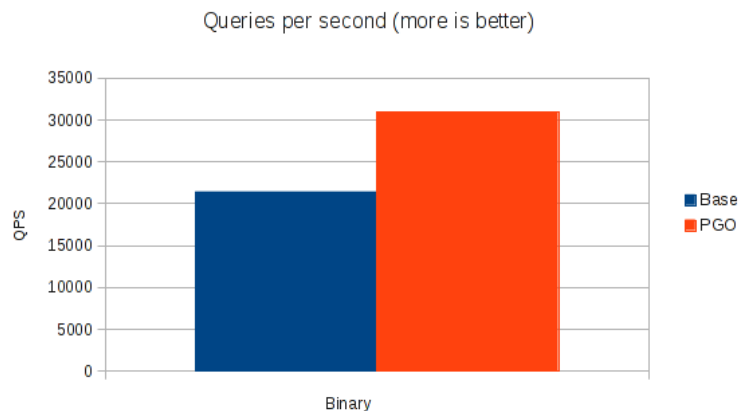
- 减少ICache Miss

- Perf: 定位Cache Miss

- PGO: 优化编译

- 优化效果

- ICache Miss: 10% to 8%
- Performance: 44%提升

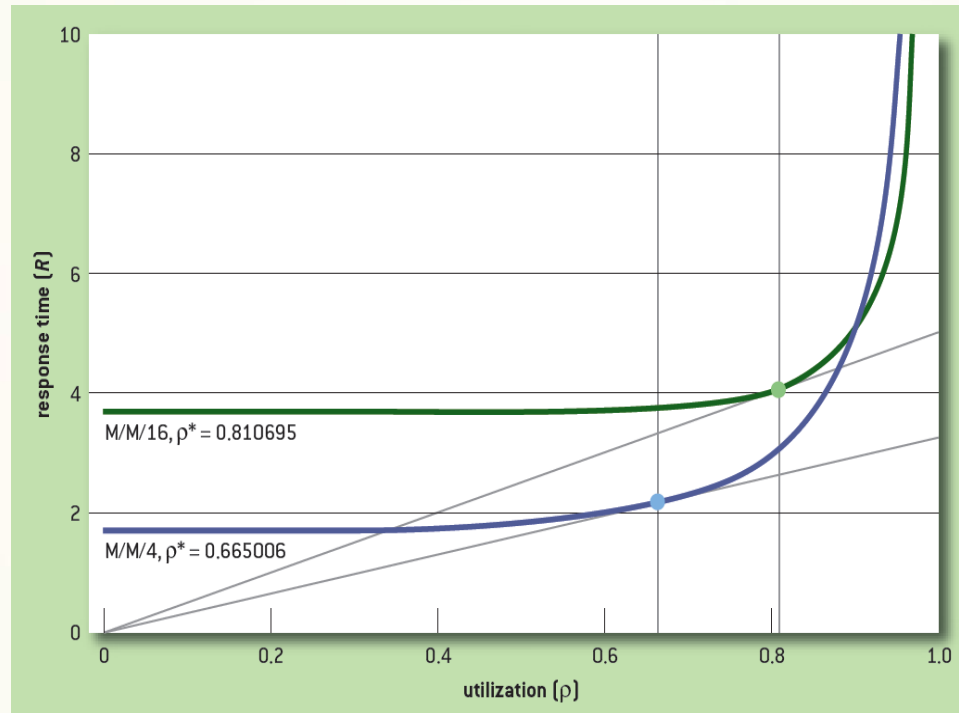


MySQL调优—基础

- mysql变量/参数大集合
 - [mysqld Option/Variable Reference](#)
- 必须掌握的调优参数
 - [innodb_flush_log_at_trx_commit](#) & [sync_binlog](#)
 - 在BGC之前，调整这两个参数，能极大降低磁盘的I/O请求(减少排队)，降低请求响应时间，提升TPS；
 - [innodb_log_file_size](#) & [innodb_log_files_in_group](#)
 - 控制日志文件大小。在写入频繁的系统下，小日志文件更易引发系统的同步刷脏页(引入新的串行点)，TPS急剧下降；
 - [innodb_io_capacity](#) & [innodb_read_io_threads](#) & [innodb_write_io_threads](#) & [innodb_purge_threads](#)
 - 根据硬件配置以及系统的读写请求，恰到好处的配置这些参数，能够最大限度的发挥磁盘的性能，同时保证系统较好的响应时间；
 - 建议将MySQL所有的参数都予以掌握，并根据实际情况，调整参数获取最佳性能；

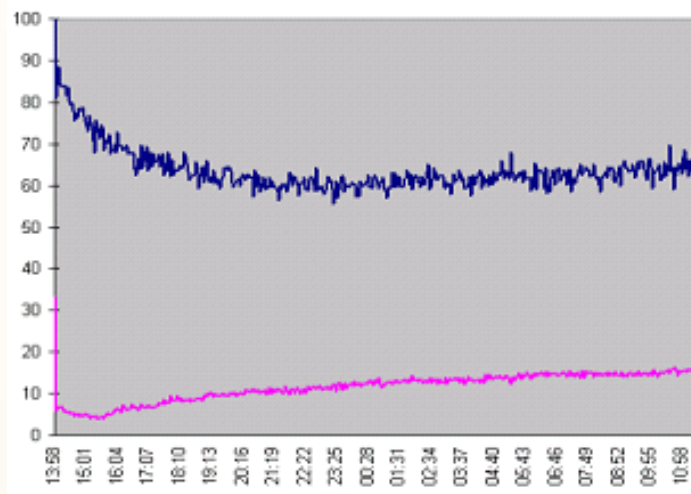
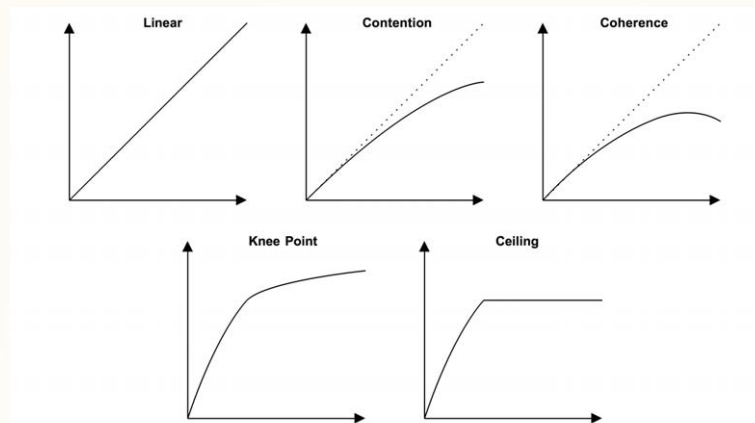
MySQL调优-关注资源利用率

- 一个数据库系统，其硬件资源(CPU、Disk、SSD...)利用率是不是越高，说明系统调优的越好？
- 用户关注
 - 响应时间(Latency)
- 老板关注
 - 资源有没有浪费？
- 调优关注
 - 保证用户响应时间的基础上，最大限度提高资源利用率；
 - [Finding The Knee](#)



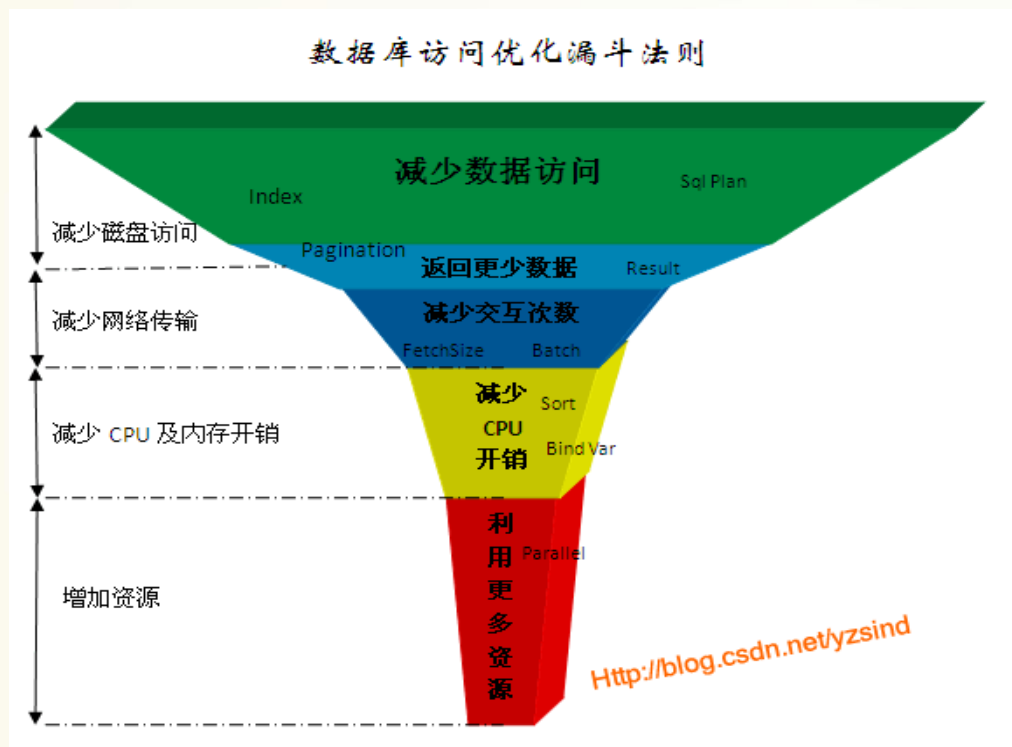
MySQL调优-认识MySQL，构建平衡系统

- 认识MySQL
 - MySQL能支持多少并发？消耗多少CPU？
 - 指导采购不同的硬件配件；
- 构建平衡系统(Balanced Systems)
 - Balanced Systems
 - 硬件有着基本相同的资源利用率吗？
 - 每个硬件的利用率都处于拐点吗？
 - 指导数据库服务器选型；
 - 合适的CPU、磁盘、网络、SSD组合；



MySQL调优-终极调优

- 面向程序员的数据访问性能优化法则



- 写在最后的一句话

- 数据库层面做再大的性能优化，都抵不上应用层的优化。同样是MySQL，既可以用来支撑Google/FaceBook/Taobao应用，也可能连你的个人网站都撑不住；

Q&A

THANKS

