互联网数据库应用实践和趋势

杨尚刚
zolker

个人简介

- 2011年加入新浪
- 负责新浪微博核心数据库架构设计
- 负责新浪数据库平台底层软硬件平台优化
- 2015年加入美图网
- 理念：设计简洁的架构

数据库排名

283 systems in ranking, November 2015

Rank			DBMS	Database Model	Score		
Nov 2015	Oct 2015	Nov 2014			Nov 2015	Oct 2015	Nov 2014
1.	1.	1.	Oracle	Relational DBMS	1480.95	+13.99	+28.82
2.	2.	2.	MySQL	Relational DBMS	1286.84	+7.88	+7.77
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1122.33	-0.90	-97.87
4.	4.	↑ 5.	MongoDB +	Document store	304.61	+11.34	+59.87
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	285.69	+3.56	+28.33
6.	6.	6.	DB2	Relational DBMS	202.52	-4.28	-3.71
7.	7.	7.	Microsoft Access	Relational DBMS	140.96	-0.87	+2.12
8.	8.	↑ 9.	Cassandra +	Wide column store	132.92	+3.91	+40.93
9.	9.	↓ 8.	SQLite	Relational DBMS	103.45	+0.78	+8.17
10.	10.	↑ 11.	Redis +	Key-value store	102.41	+3.61	+20.06

如何选择数据库

- 应用场景
- 数据量
- 可用性要求
- 数据安全性要求
- 运维复杂度

中国菜刀

MySQL

瑞士军刀

MongoDB

大象兵

HBase

金箍棒

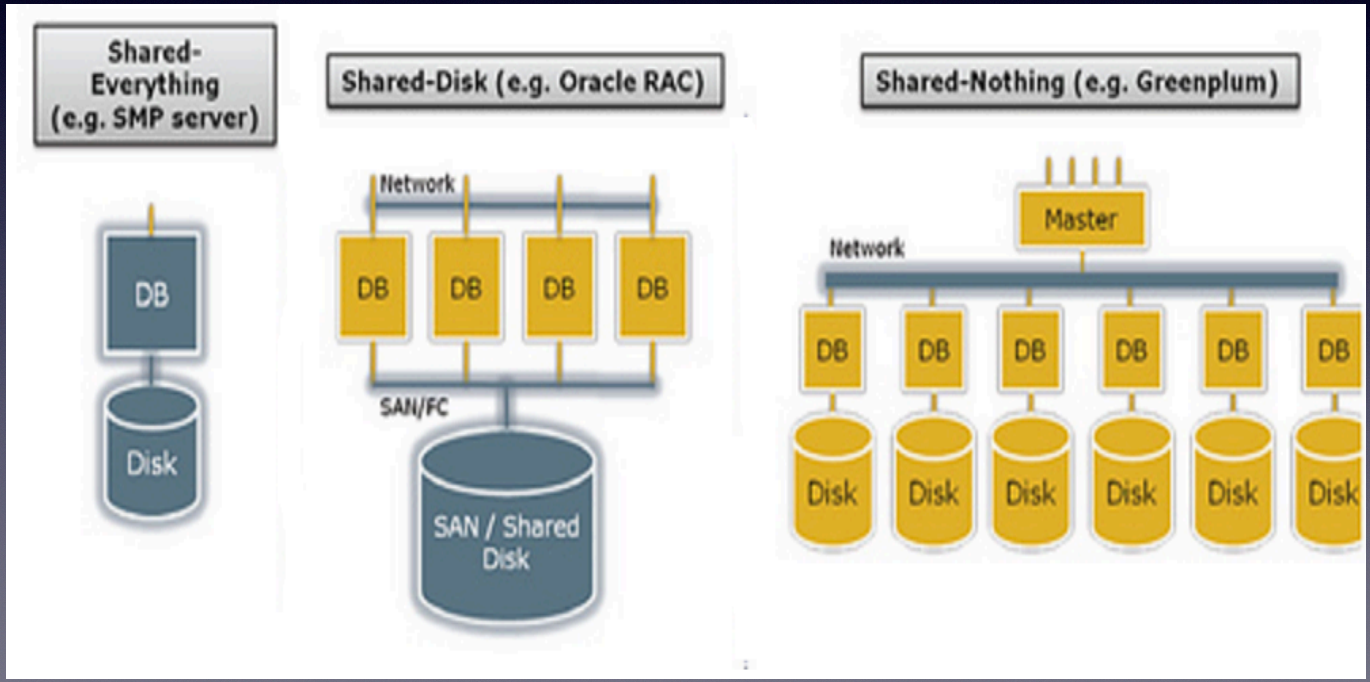
Redis

数据库对比

数据库	优点	缺点	场景
MySQL	结构简单，部署方便，社区成熟，稳定性非常好，良好的事务和SQL支持	扩展性差，软件本身性能瓶颈大，没有成熟的集群方案，schema限制	百亿以内的数据存储，对数据安全性和事务支持有要求
MongoDB	Schema－free，快速开发，本身支持集群和sharding，支持空间索引	锁的粒度大，并发性能差，性能受限于内存，解决方案上有待考验	LBS，缓存，小文件存储
HBase	类Bigtable系统，基于Hadoop生态系统，良好的扩展性，高写入能力，数据自动分片	架构复杂，运维成本高	搜索，数据写入非常高，监控数据
Redis	高性能，部署简单，丰富的数据类型支持，支持数据持久化，集群方案支持	性能受限于内存，单进程问题	适合小数据高读写场景

数据库架构

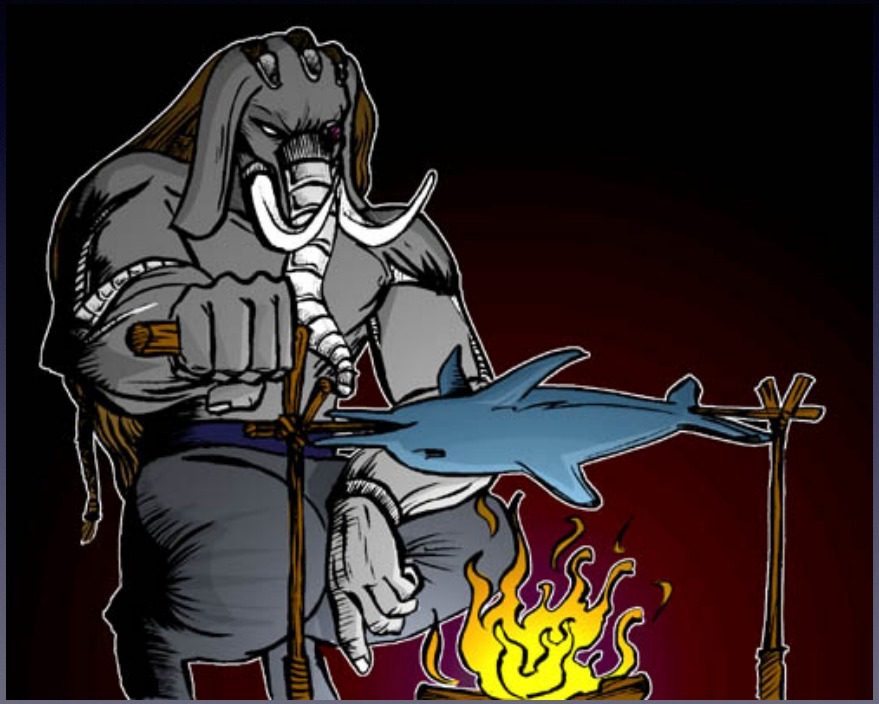
- shared noting
- shared everything
- module



MySQL当前存在问题

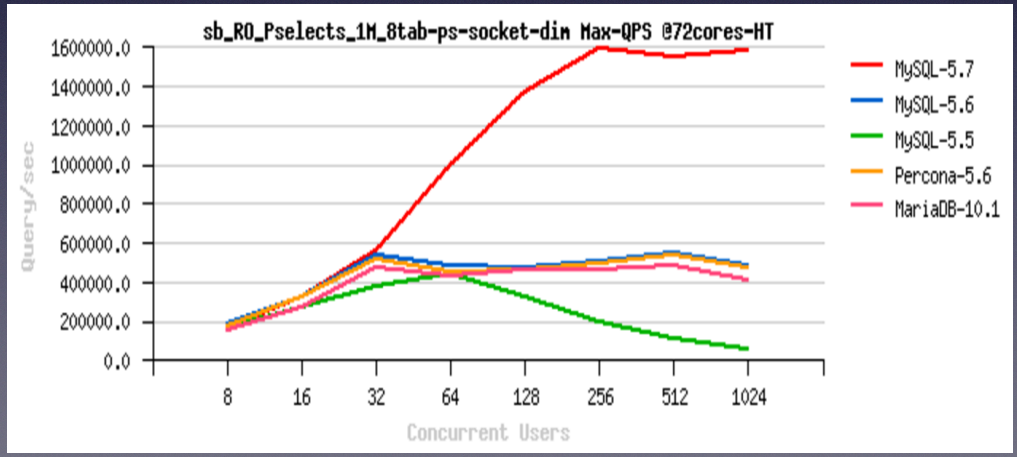
- 优化器对复杂SQL支持不好
- 对SQL标准支持不好
- 大规模集群方案不成熟，主要指中间件
- 逻辑复制
- Online DDL
- HA方案不完善
- 备份和恢复方案还是比较复杂，需要依赖外部组件
- 展现给用户信息过少
- 众多分支

PostgreSQL VS MySQL?



MySQL优势

- 扩展性
- 可维护性好
- 良好的生态环境
- 性能并不差

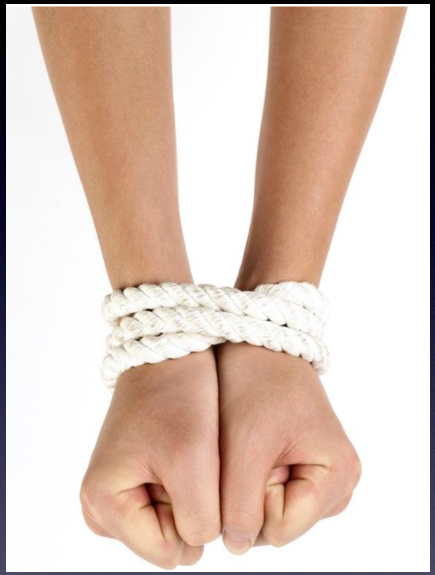


DBA Life

- 满足各种各样的开发需求
- 各式各样的Schema审核
- SQL优化
- 各种救火和处理报警 :主库故障，缓存“雪崩”
- 各种业务和项目上线
- 业务沟通

解放DBA的双手

- 制定运维规范
- 流程自动化
- 减少沟通成本



规范化

- 部署规范
- 软件规范
- 业务开发规范
- 日常运维规范

软件规范

- 操作系统
- MySQL版本
- 相关工具版本

我的建议
MySQL社区版>Percona Server
>MariaDB>MySQL 企业版

业务开发规范

- 数据库开发规范：

开发规范是针对内部开发的一系列建议或规则

由DBA制定，如果有DBA的话

开发规范也包含几部分：基本命名和约束规范，字段设计规范，索引规范，使用规范

- 意义：

保证线上数据库schema规范

减少出问题概率

方便自动化管理

需要长期坚持，是一个双赢的事情

开发规范示例

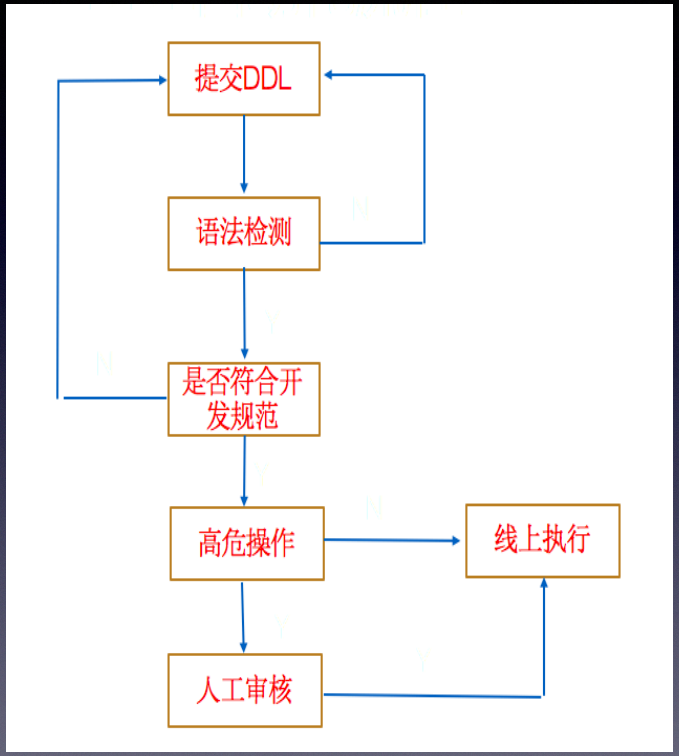
- 表字符集选择UTF8 ，如果需要存储emoj表情，需要使用UTF8mb4(MySQL 5.5.3以后支持)
- 存储引擎使用InnoDB
- 变长字符串尽量使用varchar 和varbinary
- 不在数据库中存储图片、文件等
- 每张表数据量控制在1亿以下

运维自动化

- DDL自动化
- 备份系统
- 慢日志系统

DDL自动化

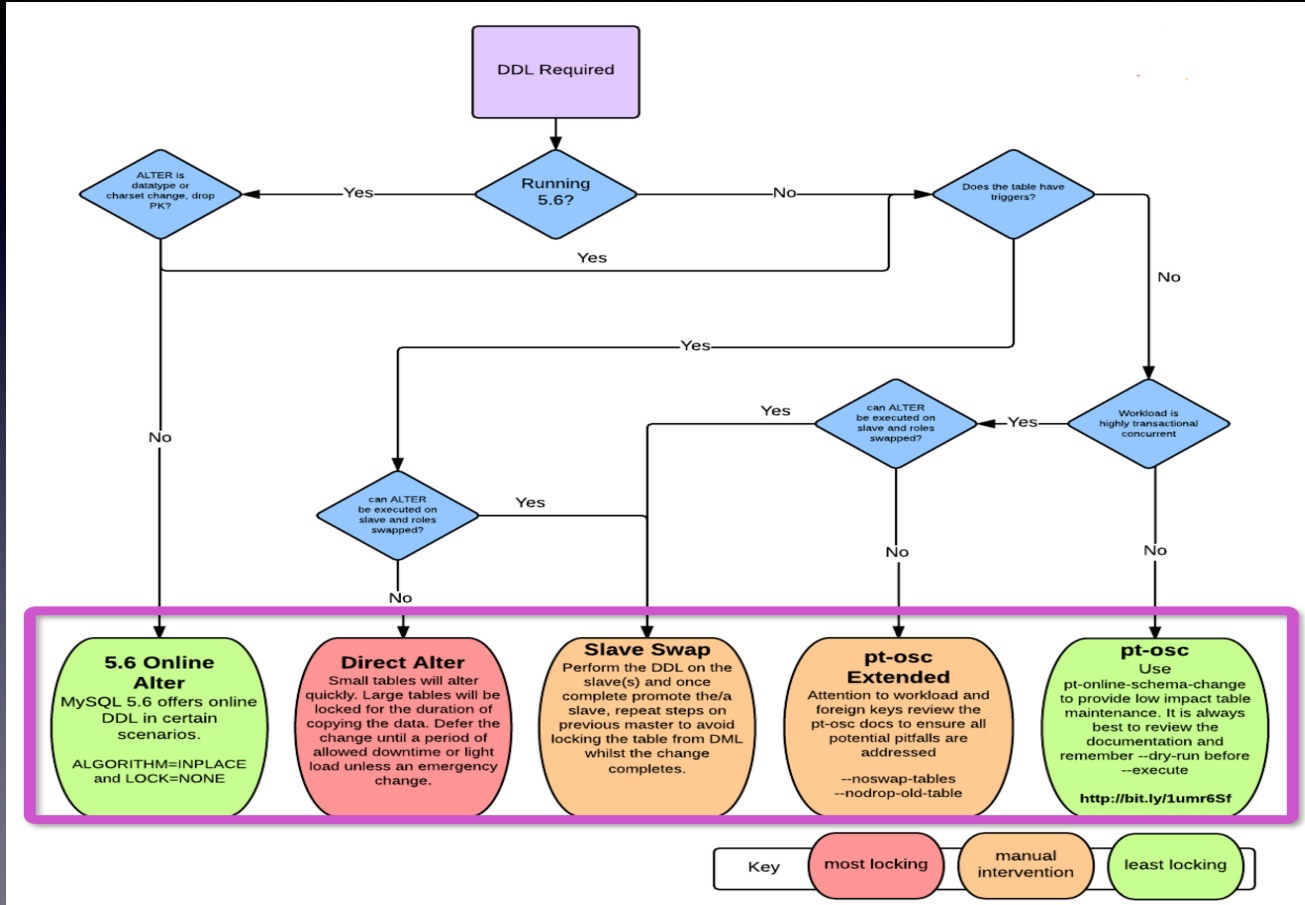
- 建表自动化
- 审表自动化
- 改表自动化



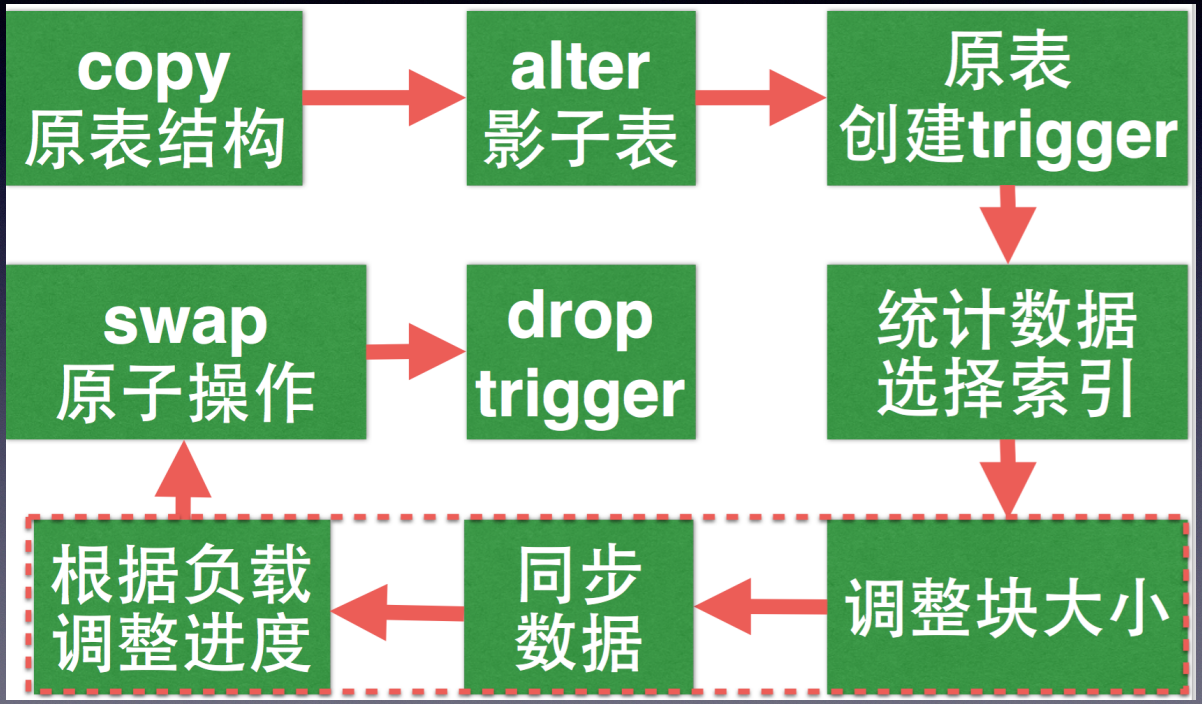
Online DDL

- 原生MySQL执行DDL是需要锁表的
- 对服务影响很大
- MySQL在这方面支持的是比较差的，对DBA来说是很痛苦的，
- 如何做到Online DDL呢，是不是就无解了呢

可选DDL方案


DDL Required
Running 5.6?
Yes
No
ALTER is datatype or charset change, drop PK?
Yes
No
Does the table have triggers?
Yes
No
Yes
No
Workload is highly transactional concurrent
Yes
can ALTER be executed on slave and roles swapped?
Yes
No
No
can ALTER be executed on slave and roles swapped?
Yes
No
5.6 Online Alter
MySQL 5.6 offers online DDL in certain scenarios.
ALGORITHM=INPLACE and LOCK=NONE
Direct Alter
Small tables will alter quickly. Large tables will be locked for the duration of copying the data. Defer the change until a period of allowed downtime or light load unless an emergency change.
Slave Swap
Perform the DDL on the slave(s) and once complete promote the/a slave, repeat steps on previous master to avoid locking the table from DML whilst the change completes.
pt-osc Extended
Attention to workload and foreign keys review the pt-osc docs to ensure all potential pitfalls are addressed
--noswap-tables
--nodrop-old-table
pt-osc
Use pt-online-schema-change to provide low impact table maintenance. It is always best to review the documentation and remember --dry-run before --execute
http://bit.ly/1umr6Sf
Key
most locking
manual intervention
least locking

Online Schema Change原理



Online DDL 对比

CHANGE OPERATION	ONLINE DDL			PT-ONLINE-SCHEMA-CHANGE		
	ROW(S) AFFECTED	IS TABLE LOCKED?	TIME (SEC)	ROW(S) AFFECTED	IS TABLE LOCKED?	TIME (SEC)
Add Index	0	No	3.76	All rows	No	38.12
Drop Index	0	No	0.34	All rows	No	36.04
Add Column	0	No	27.61	All rows	No	37.21
Rename Column	0	No	0.06	All rows	No	34.16
Rename Column + change its data type	All rows	Yes	30.21	All rows	No	34.23
Drop Column	0	No	22.41	All rows	No	31.57
Change table ENGINE	All rows	Yes	25.30	All rows	No	35.54

OSC的一些坑

- 添加唯一键，导致数据丢失
- 延时备份
- 行格式下，只在从库使用OSC，丢数据

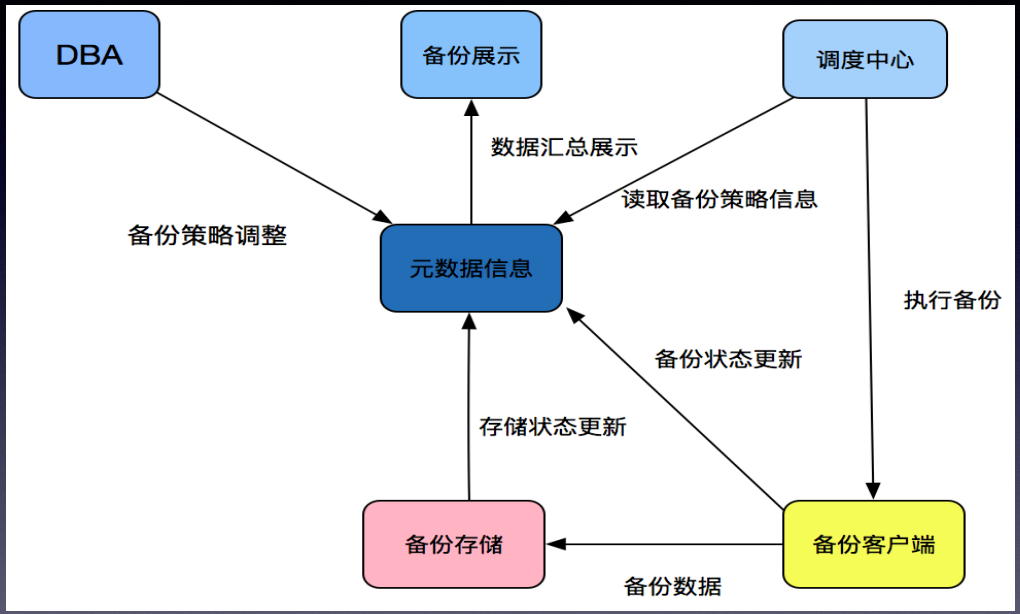
备份系统

- 数据库数据安全性是首先要保证的，也是最核心的
- 备份的意义是什么呢
- **数据恢复！**
- **数据恢复！**
- **数据恢复！**

备份方式

- 全量备份 VS 增量备份
- 热备 VS 冷备
- 物理备份 VS 逻辑备份
- 延时备份
- 全量binlog备份
- 建议方式
  - 热备 + 物理备份，核心业务：延时备份 + 逻辑备份

备份框架


DBA
备份展示
调度中心
数据汇总展示
读取备份策略信息
备份策略调整
元数据信息
执行备份
备份状态更新
存储状态更新
备份存储
备份客户端
备份数据

备份优化

- MyISAM表多flush tables with read lock时间过长问题
- Binlog备份

未来备份优化

- 采用分布式文件系统原因

解决存储分配的问题

解决存储NFS备份效率低问题

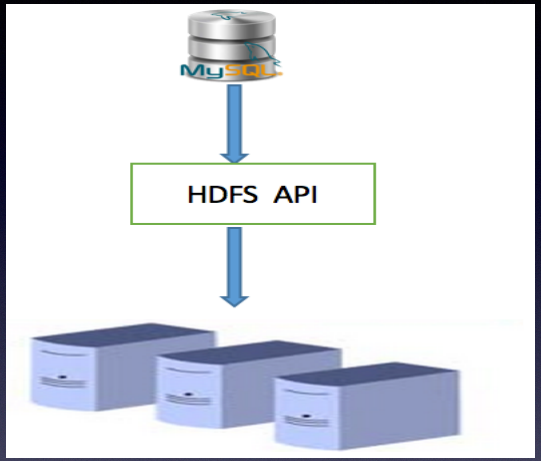
存储集中式管理

数据可靠性更好

- 使用分布式文件系统优化点

Pbzip压缩降低网络带宽消耗

erasure code方案

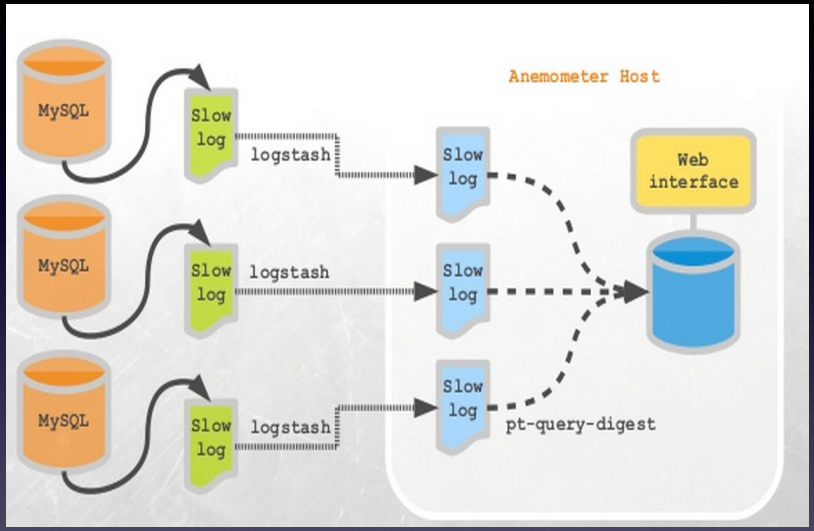


慢日志系统

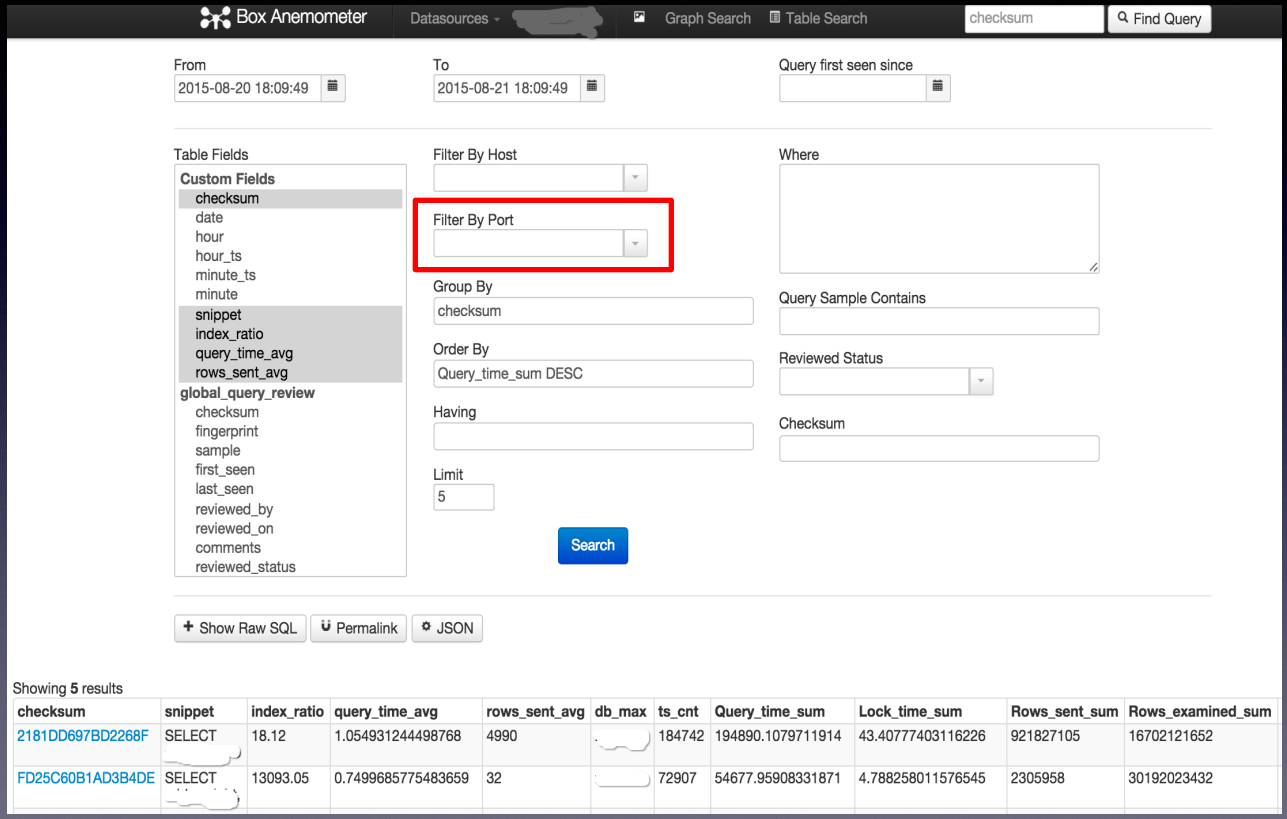
- 慢日志是MySQL内部提供的SQL执行的统计日志
- 用于线上数据库性能诊断
- 为SQL性能优化提供依据

依赖组件

- pt-query-digest
- Logstash
- Anemometer



系统界面



checksum	snippet	index_ratio	query_time_avg	rows_sent_avg	db_max	ts_cnt	Query_time_sum	Lock_time_sum	Rows_sent_sum	Rows_examined_sum
2181DD697BD2268F	SELECT	18.12	1.054931244498768	4990		184742	194890.1079711914	43.40777403116226	921827105	16702121652
FD25C60B1AD3B4DE	SELECT	13093.05	0.7499685775483659	32		72907	54677.95908331871	4.788258011576545	2305958	30192023432

一些优化

- Anemometer加入端口过滤条件支持
- 后续会加入对产品线和业务过滤
- 实时分析

性能优化

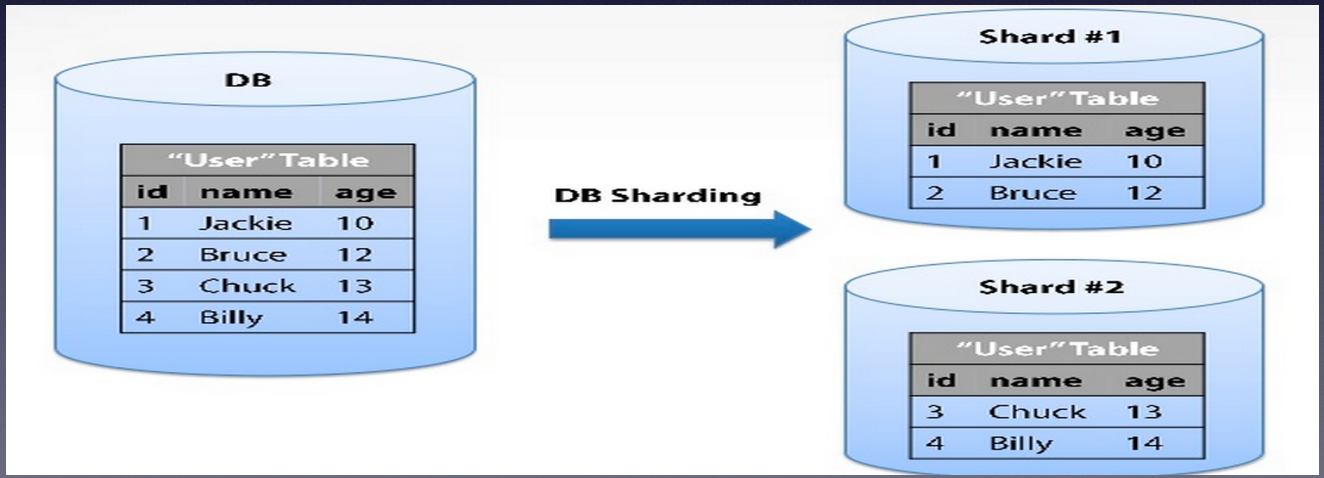
- MySQL优化
- 系统优化
- 硬件优化

MySQL优化

- 更大的redo日志
- bufferpool dump
- 禁用Query Cache
- InnoDB
- jemalloc
- group commit
- transportable tablespace

sharding

- Sharding is very complex, so it's best not to shard until it's obvious that you will actually need to!
- 垂直拆分和水平拆分
- 拆分前提：单表写入瓶颈或容量瓶颈



新浪微博单表60亿问题

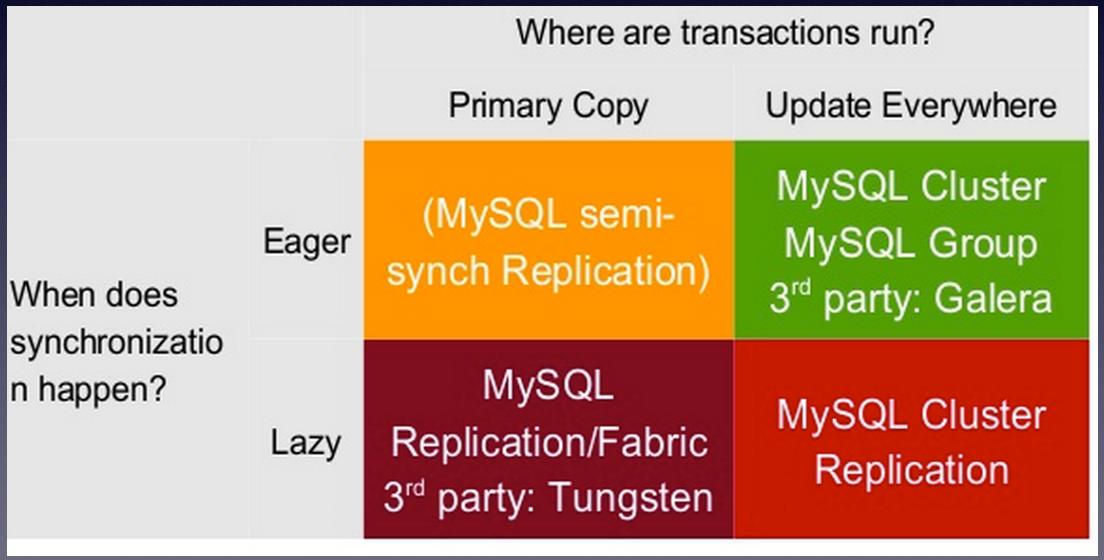
- 单一业务，实例级别拆分，单表依然有60亿+
- 单表文件大小1.2TB+
- 没有可靠中间件，拆分成本高
- 增加业务复杂度
- 运维成本高
- 评估TB级单表的风险
- 随着Online DDL等特性的成熟，这种大表以后也是一个趋势

MySQL 5.6 VS 5.7

VARIABLE	5.6.26	5.7.8
binlog_error_action	IGNORE_ERROR	ABORT_SERVER
binlog_format	STATEMENT	ROW
binlog_gtid_simple_recovery	OFF	ON
eq_range_index_dive_limit	10	200
innodb_buffer_pool_dump_at_shutdown	OFF	ON
innodb_buffer_pool_instances	8	1
innodb_buffer_pool_load_at_startup	OFF	ON
innodb_checksum_algorithm	innodb	crc32
innodb_file_format	Antelope	Barracuda
innodb_large_prefix	OFF	ON
innodb_log_buffer_size	8388608	16777216
innodb_purge_threads	1	4
innodb_strict_mode	OFF	ON
sync_binlog	0	1

MySQL复制

- GTID
- loss-less semi-replication
- group replication



Loss-less

MySQL 5.5:
semi-synchronous replication

- Master commit
- Slave receive
- Client ack

- MySQL 5.7.2:
loss-less semi-sync replication

- Slave receive
- Master commit
- Client ack

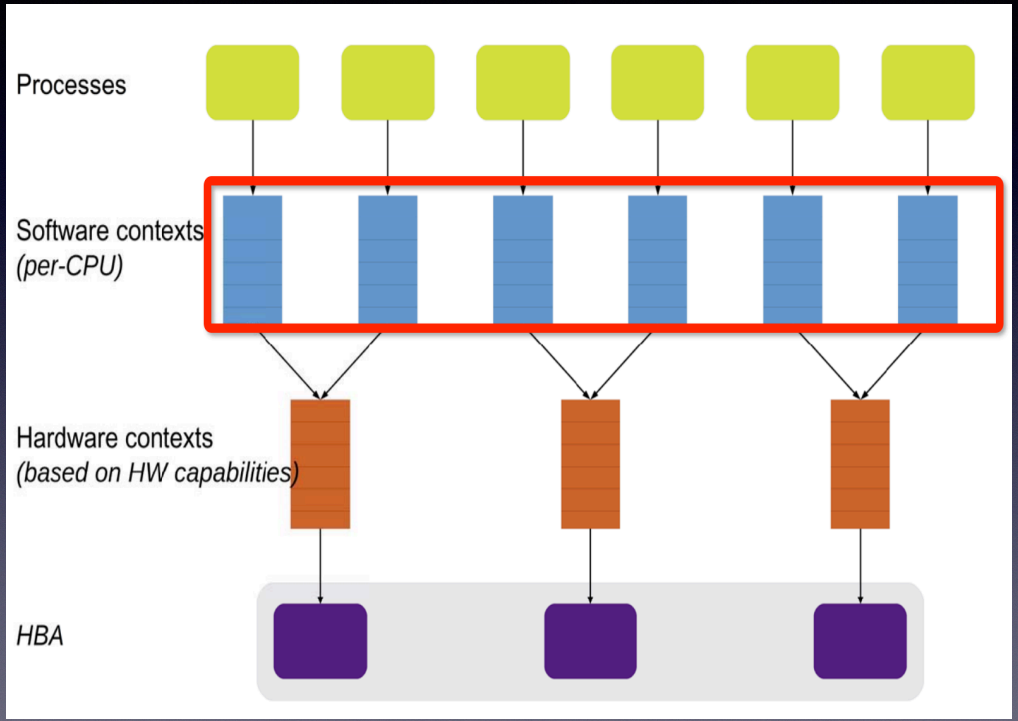
系统优化

- NUMA问题，建议关闭
- 调整swappiness
- 修改IO调度算法为noop/deadline
- 文件系统XFS/Ext4
- 系统limits限制
- 网卡多队列，当然一般可能遇不到这种场景

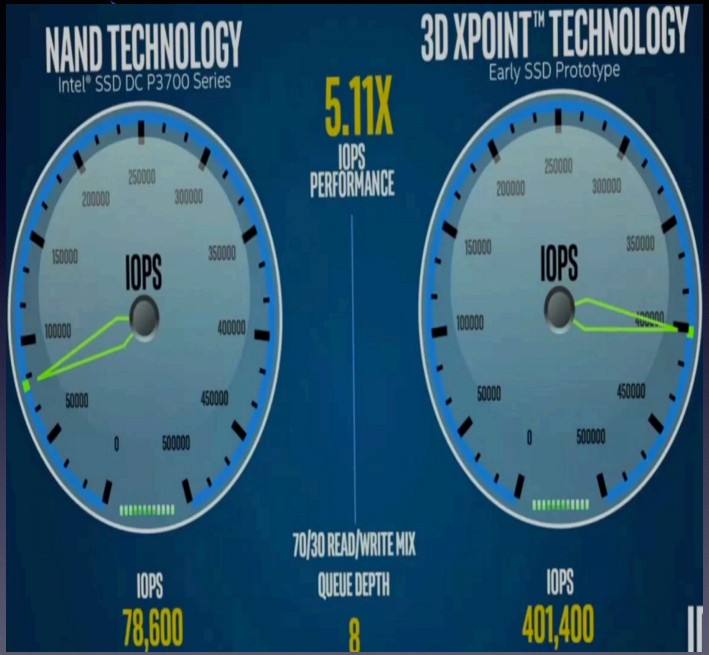
硬件优化

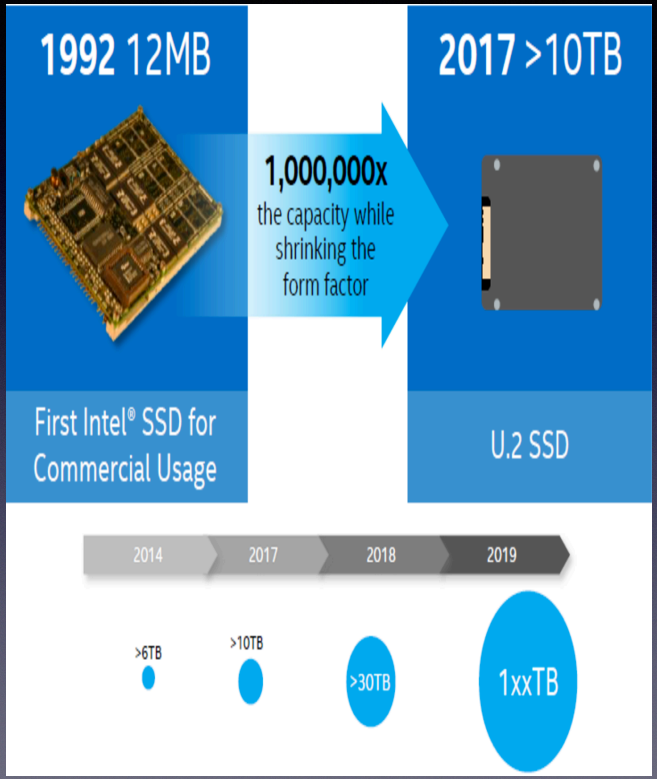
- 硬件发展很快，反而现在软件是瓶颈
- 使用SSD，SATA 或者PCIe
- CPU多核
- 大内存，单机多实例
- blk-mq
- scsi-mq

block-mq

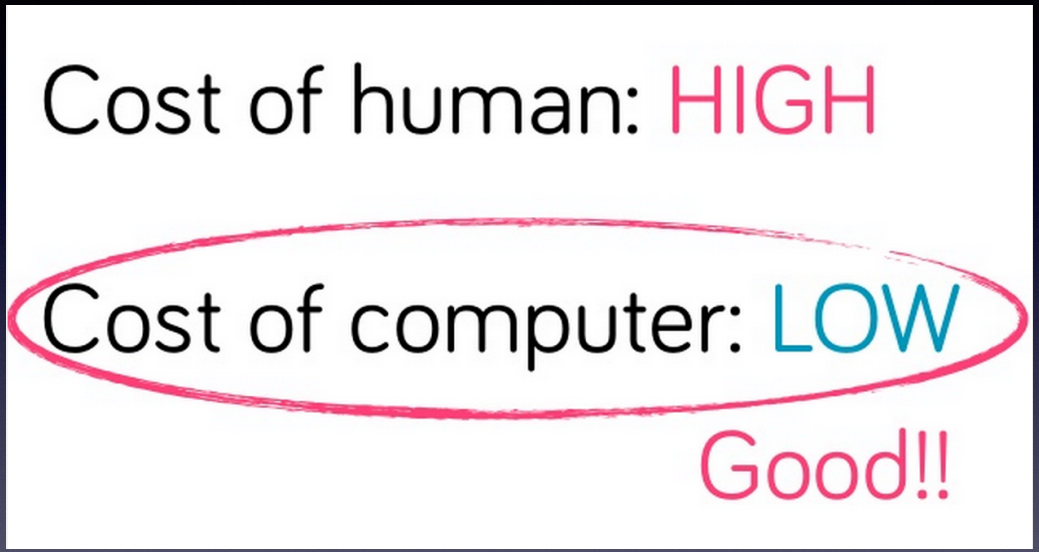

Processes
Software contexts
(per-CPU)
Hardware contexts
(based on HW capabilities)
HBA

硬件发展


NAND TECHNOLOGY
Intel® SSD DC P3700 Series
3D XPOINT™ TECHNOLOGY
Early SSD Prototype
5.11X
IOPS
PERFORMANCE
IOPS
IOPS
70/30 READ/WRITE MIX
QUEUE DEPTH
8
IOPS
78,600
IOPS
401,400


1992 12MB
2017 >10TB
1,000,000x
the capacity while
shrinking the
form factor
First Intel® SSD for
Commercial Usage
U.2 SSD
2014
2017
2018
2019
>6TB
>10TB
>30TB
1xxTB

人还是比机器贵的



未来发展

- 软硬件结合
- 软件优化

定制

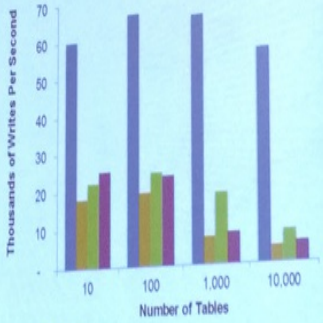
- Amazon Aurora
- Compatible with the open source MySQL
- Most of the smarts are in the storage
- **A data insert in MySQL requires six writes , Aurora requires only two**
- 软硬件结合
- 最重要的地方就是可用性的提升，性能是其次

Amazon Aurora

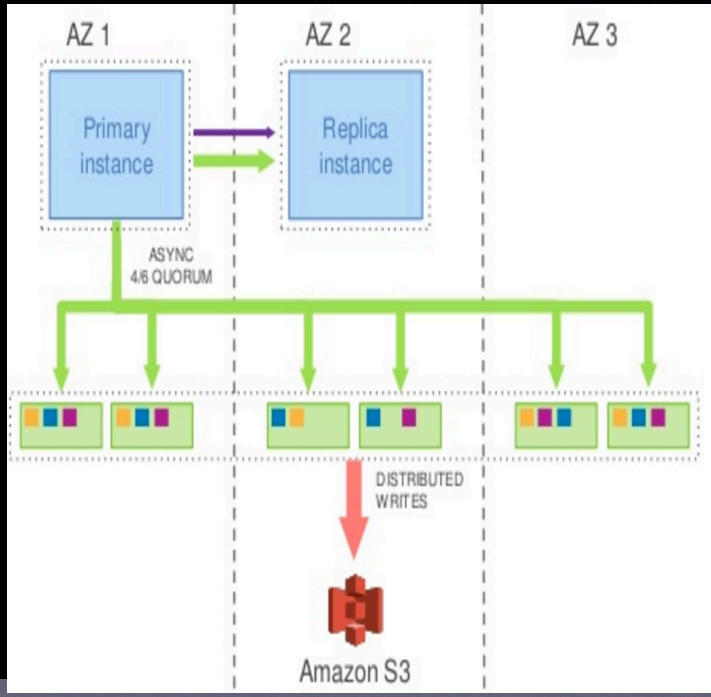
Amazon Aurora's writes scale with table count

Write Performance & Table Count

Tables	Aurora	MySQL I2.8XL Local SSD	MySQL I2.8XL RAM Disk	RDS MySQL 30K IOPS (Single AZ)
10	60,000	18,000	22,000	25,000
100	66,000	19,000	24,000	23,000
1,000	64,000	7,000	18,000	8,000
10,000	54,000	4,000	8,000	5,000



Write only workload
1,000 Connections
Query Cache (default – On for Aurora, Off for MySQL)

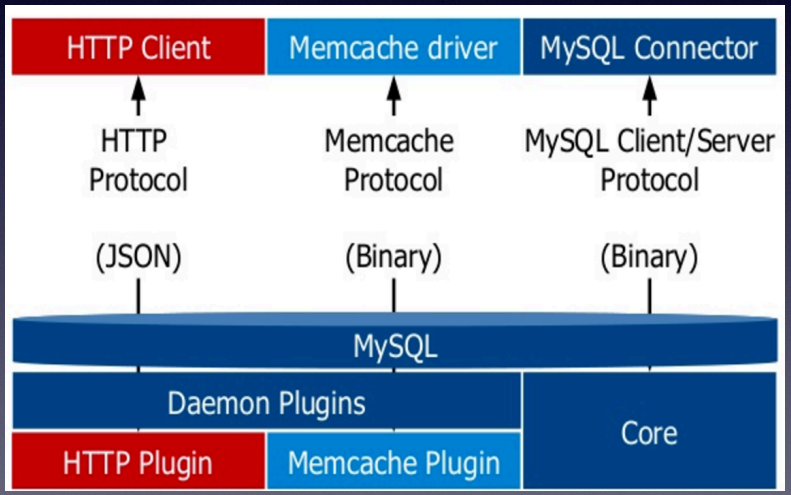


分布式存储

- 按照提供接口主要分成两类：基于文件接口和基于块设备
- 考虑到数据库场景的需求，使用分布式块设备接口是一个比较合理的选择
- 目前分布式块设备存储方案
  - Sheepdog
  - GlusterFS
  - Ceph
- 数据库的可伸缩性
- 高性能，相对而言，目前看性能在数据库上并不理想
- 提高整体资源利用率
- 简化存储管理

数据整合

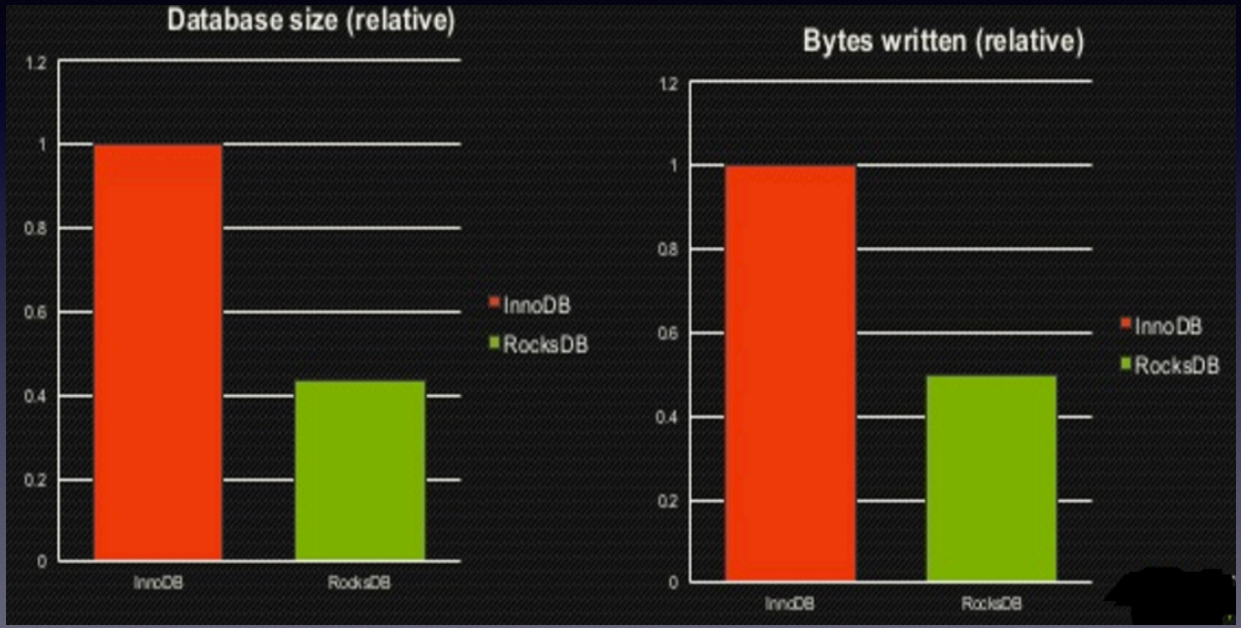
- 多引擎
- 多种数据格式
- 异构数据同步



RocksDB

- LSM Tree
- LevelDB
- SSD
- MyRocks: MySQL + RocksDB
- MongoRocks : MongoDB+RocksDB

MyRocks

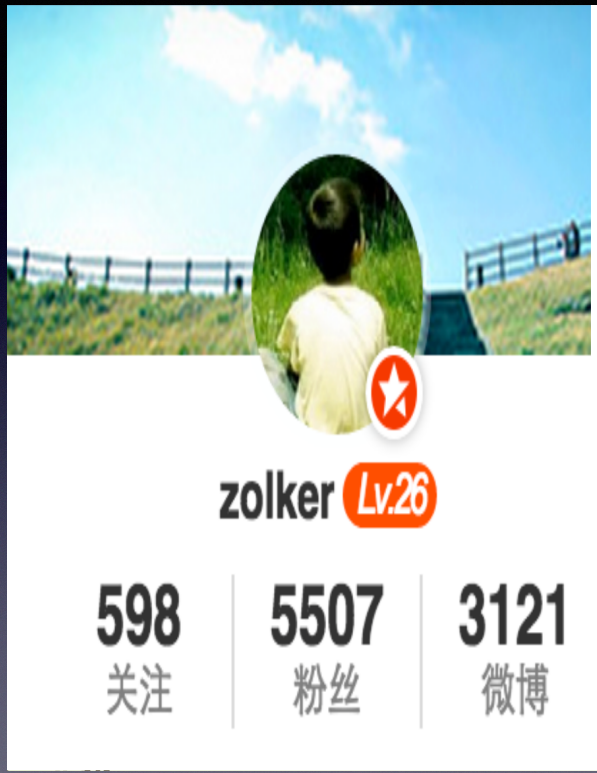

Database size (relative)
1.2
1
0.8
0.6
0.4
0.2
0
InnoDB
RocksDB
InnoDB
RocksDB
Bytes written (relative)
1.2
1
0.8
0.6
0.4
0.2
0
InnoDB
RocksDB
InnoDB
RocksDB

总结

- 优化是无止境的
- 服务可控
- 长远眼光
- One Size Doesn`t Fit All

Q & A

联系方式


zolker Lv.26
598
关注
5507
粉丝
3121
微博


zolker
北京 海淀
扫一扫上面的二维码图案，加我微信