1、 硬件层面优化
1.1 数据库物理机采购（*****）
1.2 企业案例：
1.3 服务器硬件配置调整
(1)服务器BIOS调整： （buffer和缓存调整大点。）
(2)阵列卡调整：
2 、操作系统层面优化
2.1 操作系统及MySQL实例选择
2.2 文件系统层优化（***）
2.2.1 调整磁盘Cache mode
2.2.2 采用Linux I/O scheduler算法deadline
2.2.3 采用xfs文件系统
2.2.4 mount挂载文件系统
2.2.5 Linux 内核参数优化
2.3 优化TCP协议栈
2.4 网络优化
2.5 其他优化
3、 MySQL数据库层面优化（*****）
3.1 my.cnf参数优化
3.2 关于库表的设计规范
3.3 SQL语句的优化
3.3.1 索引优化
网站打开慢之慢查询
3.3.2 大的复杂的SQL语句拆分成多个小的SQL语句
3.3.3 数据库是存储数据的地方，但不是计算数据的地方
3.3.4 搜索功能，like ‘%oldboy%’ 一般不要用MySQL数据库
4、 网站集成架构优化（*****）
网站集群架构上的优化
5、MySQL数据库管理流程（*****）
6、MySQL数据库安全优化（*****）
6.1 MySQL基础安全

MySQL数据库优化

1、 硬件层面优化

1.1 数据库物理机采购（*****）

- CPU（运算）：64位CPU，一台机器2-16颗CPU，至少2-4颗，L2越大越好。
- 内存：96G-256G（百度），3-4个实例。32-64G，跑1-2个实例（新浪）。
- disk(磁盘IO)：机械盘：选SAS，数量越多越好。

性能：SSD（高并发） > SAS（普通业务线上） >SATA（线下）
选SSD：使用SSD或者PCIe SSD设备，可提升上千倍的IOPS效率。
随机IO：SAS单盘能力300IOPS SSD随机IO：单盘能力可达35000IOPS Flashcache HBA卡

raid磁盘阵列： 4快盘：RAID0>RAID1(推荐)>RAID5(少用)>RAID1
主库选择raid10，从库可选raid5/raid0/raid10，从库配置等于或大于主库
网卡：使用多块网卡bond，以及buffer，tcp优化
千兆网卡及千兆、万兆交换机
提示：
数据库属于IO密集型服务，硬件尽量不要使用虚拟化。
Slave硬件要等于或大于Master的性能

1.2 企业案例：

百度：某部门IBM服务器为48核CPU，内存96GB，一台服务器跑3~4个实例：
sina：服务器是DELL R510居多，CPU是E5210，48GB内存，硬盘12*300G SAS，做RAID10

1.3 服务器硬件配置调整

(1)服务器BIOS调整： (buffer和缓存调整大点。)
提升CPU效率参考设置：
a.打开Perfirmance Per Watt Optimeized（DAPC）模式，发挥CPU最大性能，数据库通常需要高运算量
b.打开CIE和C States等选项，目的也是为了提升CPU效率
c. MemoryFrequency（内存频率）选择MaximumPerformance（最佳性能）
d.内存设置菜单中，启动Node Interleaving，避免NUMA问题
(2)阵列卡调整：
a.购置阵列卡同时配备CACHE及BBU模块(机械盘)
b.设置阵列写策略为WEB，甚至OFRCE WB （对数据安全要求高）（wb指raid卡的写策略：会写（write back））
c.严禁使用WT策略，并且关闭阵列预读策略

2 、操作系统层面优化

2.1 操作系统及MySQL实例选择

1.一定要选择x86_64系统，推荐使用CentOS6.8 linux，关闭NUMA特性
2.将操作系统和数据分开，不仅仅是逻辑上，还包括物理上
3.避免使用Swap交换分区
4.避免使用软件磁盘阵列
5.避免使用LVM逻辑卷
6.删除服务器上未使用的安装包和守护进程

2.2 文件系统层优化（***）

2.2.1 调整磁盘Cache mode

启用WCE=1(Write Cache Enable),RCD=0(Read Cache Disable)模式
命令：sdparm -s WCE=1,RCD=0 -S /dev/sdb

2.2.2 采用Linux I/O scheduler算法deadline

deadline调度参数

对于Centos Linux建议 read_expire = 1/2 write_expire
echo 500 >/sys/block/sdb/queue/iosched/read_expire
echo 1000 >/sys/block/sdb/queue/iosched/write_expire
Linux I/O调度方法 Linux deadline io 调度算法。

2.2.3 采用xfs文件系统

业务量不是很大也可采用ext4，业务量很大，推荐使用xfs：调整XFS文件系统日志和缓冲变量
XFS高性能设置。

2.2.4 mount挂载文件系统

增加：async,noatime,nodiratime,nobarrier等 noatime 访问文件时不更新inode的时间戳，高并发环境下，推荐显示应用该选项，可以提高系统I/O性能 async 写入时数据会先写到内存缓冲区，只到硬盘有空档才会写入磁盘，这样可以提升写入效率！风险为若服务器宕机或不正常，会损失缓冲区中未写入磁盘的数据 解决办法：服务器主板电池或加UPS不间断电源 nodiratime 不更新系统上的directory inode时间戳，高并发环境，推荐显示该应用，可以提高系统I/O性能 nobarrier 不使用raid卡上电池

2.2.5 Linux 内核参数优化

1.将vm，swappiness设置为0-10
2.将vm，dirty_background_ratio设置为5-10，将vm，dirty_ratio设置为它的两倍左右，以确保能持续将脏数据刷新到磁盘，避免瞬间I/O写，产生严重等待

2.3 优化TCP协议栈

#减少TIME_WAIT，提高TCP效率 net.ipv4.tcp_tw_recyle=1 net.ipv4.tcp_tw_reuse=1 #减少处于FIN-WAIT-2连接状态的时间，使系统可以处理更多的连接 net.ipv4.tcp_fin_timeout=2 #减少TCP KeepAlived连接侦测的时间，使系统可以处理更多的连接。net.ipv4.tcp_keepalived_time=600 #提高系统支持的最大SYN半连接数（默认1024） net.ipv4.tcp_max_syn_backlog = 16384 #减少系统SYN连接重试次数(默认5) net,ipv4.tcp_synack_retries = 1 net.ipv4.tcp_sync_retries = 1 #在内核放弃建立的连接之前发送SYN包的数量 net.ipv4.ip_local_prot_range = 450065535 #允许系统打开的端口范围

2.4 网络优化

#优化系统套接字缓冲区 #Increase TCP max buffer size net.core.rmem_max=16777216 #最大socket读buffer net.core.wmem_max=16777216 #最大socket写buffer net.core.wmem_default = 8388608 #该文件指定了接收套接字缓冲区大小的缺省值(以字节为单位) net.core.rmem_default = 8388608 #优化TCP接收/发送缓冲区 # Increase Linux autotuning TCPbuffer limits net.ipv4.tcp_rmem=4096 87380 16777216 net.ipv4.tcp_wmem=4096 65536 16777216 net.ipv4.tcp_mem = 94500000 915000000927000000 #优化网络设备接收队列 net.core.netdev_max_backlog=3000

2.5 其他优化

net.ipv4.tcp_timestamps = 0 net.ipv4.tcp_max_orphans = 3276800 net.ipv4.tcp_max_tw_buckets = 360000

3、 MySQL数据库层面优化（*****）

3.1 my.cnf参数优化

此优化主要针对innodb引擎
如果采用MyISAM引擎，需要key_buffer_size加大。

```
key_buffer_size = 256M
#指定用于索引的缓冲区大小，增加它可得到更好的索引处理性能。对于内存在4GB左右的服务器来说，该参数可设置为
```

强烈推荐采用innodb引擎，default-storage-engine=Innodb
调整innodb_buffer_pool_size大小，考虑设置为物理内存的50%-60%左右。

innodb_buffer_pool_size = 64M

#InnoDB使用一个**缓冲池来保存索引和原始数据**，设置越大，在存取表里面数据时所需要的磁盘I/O越少。强烈建议不要武断地将InnoDB的Buffer Pool值配置为物理内存的50%~80%，应根据具体环境而定。

根据实际需要设置inno_flush_log_at_trx_commit，sync_binlog的值。如果要需要数据不能丢失，那么两个都设为1.如果允许丢失大一点数据，则可分别设为2和0，在slave上可设为0

innodb_flush_log_at_trx_commit （这个很管用）

抱怨Innodb比MyISAM慢 100倍？那么你大概是忘了调整这个值。默认值1的意思是每一次事务提交或事务外的指令都需要把日志写入（flush）硬盘，这是很费时的。特别是使用电 池供电缓存（Battery backed up cache）时。设成2对于很多运用，特别是从MyISAM表转过来的是可以的，它的意思是不写入硬盘而是写入系统缓存。日志仍然会每秒flush到硬 盘，所以你一般不会丢失超过1-2秒的更新。设成0会更快一点，但安全方面比较差，即使MySQL挂了也可能会丢失事务的数据。而值2只会在整个操作系统 挂了时才可能丢数据。

设置innodb_file_per_table = 1，使用独立表空间 设置innodb_data_file_path = ibdata1:1G:autoextend,不要使用默认的10% 设置innodb_log_file_size=256M,设置innodb_log_files_in_group=2，基本可满足90%以上的场景； 不要将innodb_log_file_size参数设置太大，这样可以更快同时又更多的磁盘空间，丢掉多的日志通常是好的，在数据库崩溃后可以降低恢复数据库的事件 设置long_query_time = 1记录那些执行较慢的SQL，用于后续的分析排查； 根据业务实际需要，适当调整max_connection（最大连接数）max_connection_error（最大错误数，建议设置为10万以上，而open_files_limit、innodb_open_files、table_open_cache、table_definition_cache这几个参数则可设为约10倍于max_connection的大小；）不要设置太大，会将数据库撑爆 建议关闭query cache功能或降低设置不要超过512M query_cache_size = 64M #指定MySQL查询缓冲区的大小。可以通过在MySQL控制台观察，如果Qcache_lowmem_prunes的值非常大，则表明经常出现缓冲不够的情况;如果Qcache_hits的值非常大，则表明查询缓冲使用得非常频繁。另外如果改值较小反而会影响效率，那么可以考虑不用查询缓冲。对于Qcache_free_blocks，如果该值非常大，则表明缓冲区中碎片很多。 tmp_table_size = 64M #设置内存临时表最大值。如果超过该值，则会将临时表写入磁盘，其范围1KB到4GB。 max_heap_table_size = 64M #独立的内存表所允许的最大容量。 table_cache = 614 #给经常访问的表分配的内存，物理内存越大，设置就越大。调大这个值，一般情况下可以降低磁盘IO，但相应的会占用更多的内存,这里设置为614。

3.2 关于库表的设计规范

1.推荐utf-8字符集，虽然有人说谈没有latin1快

2.固定字符串的列尽可能多用定长char，少用varchar

存储可变长度的字符串使用VARCHAR而不是CAHR---节省空间，因为固定长度的CHAR，而VARCHAR长度不固定（UTF8不愁此影响）

3.所有的InnoDB表都设计一个无业务的用途的自增列做主键

4.字段长度满足需求前提下，尽可能选择长度小的

5.字段属性尽量都加NOT NULL约束（空的字段不能走索引，查询速度慢）

对于某些文本字段，例如“省份”或者“性别”我们可以将他们定义为ENUM类型

6.尽可能不使用TEXT/BLOB类型，确实需要的话，建议拆分到子表中，不要和主表放在一起，避免SELECT*的时候读性能太差。

7.读取数据时，只选取所需要的列，不要每次都SELECT * 避免产生严重的随机读问题，尤其是读到一些TEXT/BLOB类型，确实需要的话，建议拆分到子表中，不要和主表放在一起，避免SELECT*的时候读性能太差

8.对一个VARCHAR（N）列创建索引时，通常取其50%（甚至更小）左右长度创建前缀索引就足以满足80%以上的查询需求了，没必要创建整列的全长度索引。

9.多用符合索引，少用多个独立索引，尤其是一些基础（Cardinality）太小（如果说：该列的唯一值总数少于255）的列就不要创建独立索引了。

3.3 SQL语句的优化

3.3.1 索引优化

1）白名单机制一百度，项目开发，DBA参与，减少上线后的慢SQL数据

抓出慢SQL,配置my.cnf

long_query_time = 2

log-slow-queries=/data/3306/slow-log.log

log_queries_not_using_indexs

按天轮询：slow-log.log

2）慢查询的日志分析工具——mysqlsla或pt-query-digest（推荐）

pt-quey-diges,mysqldumpslow,mysqlsla,myprofi,mysql-explain-slow-log,mysqllogfileter

3)每天晚上0点定时分析慢查询，发到核心开发，DBA分析，及高级运维，CTO的邮箱里

DBA分析给出优化建议-->核心开发确认更新-->DBA线上操作处理

4)定期使用pt-duplicate-key-checker检查并删除重复的索引
定期使用pt-index-usage工具检查并删除使用频率很低的索引

5)使用pt-online-schema-change来完成大表的ONLINE DDL需求
6)有时候MySQL会使用错误的索引，对于这种情况使用USE INDEX
7)使用explain及set profile优化SQL语句
网站打开慢之慢查询
3.3.2 大的复杂的SQL语句拆分成多个小的SQL语句
子查询，JOIN连表查询，某个表4000万条记录
3.3.3 数据库是存储数据的地方，但不是计算数据的地方
对数据计算，应用类处理，都要拿到前端应用解决。禁止在数据库上处理
3.3.4 搜索功能，like ‘%oldboy%’ 一般不要用MySQL数据库
使用连接(JOIN)来代替子查询（Sub_Queries）
避免在整个表上使用cout(*)，它可能锁住整张表
多表联接查询时，关联字段类型尽量一致，并且都要有索引。
在WHERE子句中使用UNION代替子查询
多表连接查询时，把结果集小的表（注意，这里是指过滤后的结果集，不一样是全表数据量小的）作为驱动表
4、 网站集成架构优化（*****）
网站集群架构上的优化
1.服务器上跑多实例，2-4个（具体需要看服务器的硬件信息）
2.主从复制一主五从，采用mixed模式（混合或行模式），尽量不要跨机房同步（进程远程读本地写），（数据要一致，拉光纤，没有网络延迟）
3.定期使用pt-table-checksum、pt-table-sync来检查并修复mysql主从复制的数据差异（重构）
4.业务拆分：搜索功能，like '%oldboy% ' 一般不要用MySQL数据库
5.业务拆分：某些业务应用使用nosql持久化存储，例如：memcached、redis、ttserver
例如粉丝关注，好友关系等
6.数据库前端必须要加cache，例如：memcached，用户登录，商品查询
7.动态的数据库静态化，整个文件静态化，页面片段静态化
8.数据库集群与读写分离。一主多从，通过程序或dbproxy进行集群读写分离
9.单表超过800万，拆库拆表。人工拆表拆库（登录、商品、订单）
10.百度、阿里国内前三公司，会选择从库进行备份，对数据库进行分库分表
5、MySQL数据库管理流程（*****）
任何一次人为数据库记录的更新，都要走一个流程：
a.人的流程：开发-->核心开发-->运维或DBA
b.测试流程：内网测试-->IDC测试-->线上执行
c.客户端管理，phpmyadmin
6、MySQL数据库安全优化（*****）
6.1 MySQL基础安全
1.启动程序700，属主和用户组为MySQL。
2.为MySQL超级用户root设置密码。
3.如果要求严格可以删除root用户，创建其他管理用户，例如admin。
4.**登录时尽量不要在命令行暴露密码**，备份脚本中如果有密码，给设置700，属主和密码组为mysql或root。
5.删除默认存在的test库。
6.初始删除无用的用户，只保留。
| root | 127.0.0.1 |
| root | localhost |
7.**不要一个用户管理所有的库，尽量专库专用户**(少量库)
8.清理mysql操作日志文件~/.mysql_history（权限600，可以不删）

9.禁止开发获得到web连接的密码，禁止开发连接操作生产对外的库
10.phpmyadmin安全
11.服务器禁止设置外网IP
12.防SQL注入（WEB）php.ini或web开发插件监控，waf控制