1. MySQL性能优化
2. 优化的风险

优化不是对一个环境进行的，而是对一个复杂的已投产系统进行的，存在一定的风险，当优化了一个问题后，另一个问题可能就会出现，因此要控制调优带来的问题在可接受的范围内。

1. 优化的需求

由于优化存在一定的风险，因此不能单纯为了优化而优化，要根据业务的需求来优化，重点是保证稳定性和业务的可持续性，这两点比性能更加重要。

1. 优化的四个维度
2. 优化的方向

数据库优化有两个主要方向：安全与性能。从安全上看是要保障数据的安全性，从性能上看，是要保证数据的高性能访问。

1. 优化的维度

数据库的优化可以从四个维度进行：

硬件 ： CPU，内存，存储，网络设备

系统配置： 服务器系统，数据库服务参数

数据库表结构： 高可用，分库分表，读写分离，存储引擎，表设计

Sql及索引： sql语句，索引的使用

从优化的成本来分析： 硬件 > 系统配置 > 数据库表结构 > SQL及索引；

从优化的效果来分析： 硬件 < 系统配置 < 数据库表结构 < SQL即索引；

1. 优化的工具
2. mysqladmin

mysql客户端，可进行管理操作

1. mysqlshow

查看shell命令

1. show [SESSION | GLOBAL] variables

查看数据库参数信息

1. show [SESSION | GLOBAL] STATUS

查看数据库的状态信息

1. SHOW ENGINE INNODB STATUS Innodb

查看引擎的所有状态

1. information schema

获取元数据的方法

1. SHOW PROCESSLIST

查看当前所有连接session的状态

1. explain

查查看查询语句的执行计划

1. how index

查看表的索引信息

1. slow-log

记录慢查询语句

1. mysqldumpslow

分析slowlog文件

1. Zabbix

监控主机，系统，数据库（部署zabbix监控平台）

1. mysqlslap

分析慢日志

1. sysbench

压力测试工具

1. workbench

管理，备份，监控，分析，优化工具

1. mysql profiling

统计数据库整体状态工具

17， Performance Shema mysql

性能状态统计的数据

1. 数据库优化思路

数据库调优时不需要进行全面，大范围的调优，一般只需要在数据库层面优化即可。

1. 应急调优的思路

当业务突然卡顿，无法正常运行时，需要立刻解决，思路如下：

1. show processlist(查看链接session状态)
2. explain(分析查询计划)，show index from table(分析索引)
3. 通过执行计划判断，索引问题（有没有，是否合理）或者语句本身问题
4. show status like “%lock%” ，查询锁状态
5. SESSION\_ID，杀掉有问题的session
6. 常规调优的思路

对于业务周期性的卡顿，但是不影响正常的使用，用以下的思路解决：

1. 查看slowlog, 分析slowlog, 分析出查询慢的语句
2. 按照一定优先级，排查所有的慢语句
3. 分析top sql ，进行explain调试，查看语句执行时间
4. 调整索引或语句本身
5. 查询优化
6. MySQL查询流程

由于数据库的读写操作比例是10:1，因此sql优化主要是解决查询的优化问题。下面是查询的执行路径：

1. 客户端将查询sql语句发送到服务器中；
2. 服务器检查查询缓存中是否有该sql语句，如果有则直接返回结果，否则进入下一步；
3. 服务器通过关键字解析sql语句，通过语法解析生成解析树，解析器使用Mysql语法规则验证和解析查询，再经过预处理器生成新的解析树；
4. 解析树被校验合法后，经过查询优化器优化转成查询计划，一条语句有很多种执行方式，最后都会返回相同的结果。优化器的作用就是找到这其中最好的执行计划；
5. 查询执行引擎根据执行计划完成整个查询操作，最常使用的是MyISAM引擎和InnoDB引擎。
6. 服务器将查询结果发送回客户端；
7. 查询优化

sql是开发者与数据库的交互手段，因此SQL调优是花费最多时间的，常见的分析手段有慢查询日志，EXPLAIN分析查询，通过定位分析性能的瓶颈，才能够更好地优化数据库系统的性能。

1. 慢查询
2. 开启慢查询日志

在配置文件my.cnf或者my.ini中在【mysqld】一行下面加入两个配置参数：

log-slow-queries=/data/mysqldata/slow-query.log

long\_query\_time=5

上面是慢查询日志的存放位置，一般这个目录要有mysql的运行账号的可写权限，该目录设置为mysql的数据存放位置。下面是慢查询的阈值，只有超过这个时间才是慢查询，可以记录在慢查询日志中。

1. 慢查询分析

通过打开log文件查看哪些SQL语句执行效率低，如果日志内容过多，可以使用mysqldumpslow工具对慢查询日志进行分类汇总，显示汇总后的摘要结果。

进入log日志的存放目录，运行：

mysqldumpslow slow-query.log

该命令有一些参数，如下所示：

-s : 表示按照什么方式排序，c，t，l，r分别表示按照记录次数，时间，查询时间，返回的记录数排序；ac, at, al, ar, 表示相应的倒序。

-t n： 表示返回前面n条数据

-g: 后面可写正则匹配模式，大小写不敏感

由于开启慢查询日志会占用CPU资源，因此我们可以阶段性地开启慢查询日志来定位性能瓶颈。

1. EXPLAIN
2. 概述

EXPLAIN可以帮助开发人员分析SQL问题，它显示了SQL语句的执行计划，我们可以从执行计划中找出可以优化的地方。

1. 使用

在SQL语句前加上EXPLAIN即可：

EXPLAIN SELECT \* FROM sts

1. 输出结果

EXPLAIN的输出结果由多个列组成：

id : select查询序列号；

select\_type : select语句的类型

simple: 简单select(不使用union或子查询)

primary: 最外面的select语句

union: union中的第二个或者后面的select语句

dependent union: union中的第二个或后面的select语句

union result: union的结果

subquery: 子查询中的第一个select

dependent subquery: 子查询中的第一个select，取决于外面的查询

derived: 导出表的select

table: 显示从哪张表中查询

type: 区间索引，最重要的列，显示连接使用了何种类型

效果的排行如下所示：

system > const > eq\_ref > ref > ref\_or\_null > index\_merge > unique\_subquery > index\_subquery > range > index > ALL

possible\_keys: Mysql能够使用哪个索引在该表中找到行，如果是空的，则没有相关的索引，需要提高性能。

key: 实际使用的索引，如果为null表示没有使用索引，如果是primary表示使用了主键。

key\_len: 最长的索引宽度，长度越短越好

ref: 显示使用哪个列或者常数与key一起从表中选择行；

rows: 显示mysql执行查询时必须检查的行数

Extra: 执行状态说明，包含解决查询的详细信息