**nginx优化:**

* 1. **调整启用进程数，一般为cpu核心数量倍数 ：. worker\_processes 8;**
  2. **优化nginx的并发量 可用压力测试工具ab工具: 看情况再调试**

events {

worker\_connections 65536; #每个worker\_能满足多少并发量,默认为1024

use epoll; #epoll： nginx的查询方式,

}

* 1. **优化linux内核参数，最大文件数量（ulimit –a 查看属性，）永久设置如下**

[root@proxy ~]# ulimit -Sn 10000 #设置软限制(临时规则)

[root@proxy ~]# ulimit -Hn 10000 #设置硬限制(临时规则)

[root@proxy ~]# vim /etc/security/limits.conf #永久设置,内核参数,须重启生效

\* soft nofile 65535

\* hard nofile 65535

#分4列分别为：用户或组(\*) 硬限制(hard)或软限制(soft) 需要限制的项目(nofile) 限制的值

worker\_rlimit\_nofile 65535; #linux内核下开启文件打开数为65535，就相应应该填写65535。

* 1. **优化Nginx数据包头缓存**

修改Nginx配置文件，增加数据包头部缓存大小,解决414报错（请求的URL过长错误）

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

http {

client\_header\_buffer\_size 1k; //手打上去,默认请求包头信息的缓存

large\_client\_header\_buffers 4 4k; //手打上去,加大请求包头部信息的缓存个数与容量

.. ..

}

[root@proxy ~]# nginx -s reload

* 1. **设置浏览器本地缓存静态数据**

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

server {

listen 80;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

location ~\* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ { #在server中添加location

expires 30d; //定义客户端缓存时间为30天

}

location ~\* \.(mp4)$ { #可添加多个

expires 30d; //定义客户端缓存时间为30天

}

说明:location ~\* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ ,定义缓存类型,

* 1. **对页面进行压缩处理**
  2. **服务器内存缓存**

http {

open\_file\_cache max=65535 inactive=60s; #max缓存数量,建议和打开文件数一致，缓存60s

open\_file\_cache\_valid 60s; #多久检查一次缓存有效信息

open\_file\_cache\_min\_uses 5; #一文件在inactive时间内被打开5次，描述符在缓存中一直打开

open\_file\_cache\_errors off;

* 1. **选择合理够用的模块**

**Mysql 安装：**

1. Yum install –y mysql-server #DNS不对是无法直接yum 安装
2. 第一次连接需要查看随机生成的密码（mariadb不需要），查看mysqld.log日志文件：
3. 修改密码策略：
4. 配置选项：my.conf开启binlog日志重新规划日志存放路径，日志格式，

bind-address=0.0.0.0 #监听地址，有些版本差异，添上不会出问题

server-id= {{id.stdout}} #用ansible批量传送，传参方式获取

binlog-format=mixed #binglog日志格式

log\_bin=/var/log/mysql/mysql-bin #binlog日志存放路径和命名

relay-log=/var/log/mysql/relay-log #relay-log中继日志

relay-log-index=/var/log/mysql/relay-log.info #relay-log中继日志

relay-log-info-file=/var/log/mysql/relay-log.info

relay\_log\_purge=0 #不删除日志，MHA用得上

#以下是主从半同步复制的配置（semi sync master/slave）

plugin-load=rpl\_semi\_sync\_master=semisync\_master.so #加载主半同步复制模块

plugin-load=rpl\_semi\_sync\_slave=semisync\_slave.so #加载从半同步复制模块

rpl\_semi\_sync\_slave\_enabled=1 #启用从半同步复制模式

rpl\_semi\_sync\_master\_enabled=1 #启用主半同步复制模式

rpl\_semi\_sync\_master\_timeout=3000

**binlog日志详解**：记录了对数据库除了查询的所有操作。可以用来数据恢复、主从复制和判断数据库是否被攻击。

binlog\_format参数：该参数可以设置binlog日志存储方式，有statement、row、mixed。

　　1、 statement 格式的日志，只记录sql语句，如果读写量大，日志大小会爆炸性增长

　　2、 row格式的日志，不是记录SQL语句，而是记录记录表的行更改情况。

　　3、 mixed格式的日志，根据情况决定使用statement 还是row格式记录日志。

**数据导入：**（将/etc/passwd文件导入数据库中）

1、将文件拷贝到数据库指定的检索目录中：secure\_file\_priv="/mydatadir" 配置文件参数指定的路径

2、在数据库中创建相对应的表结构

命令格式:load date infile “目录名/文件名” Into table 表名 Fields terminated by “分隔符” lines terminated by “\n”;

mysql> load data infile "/mydatadir/passwd" into table user fields terminated by ":" lines terminated by "\n";

**数据导出：**导出作用:把表记录导出到文件

命令格式:sql 查询结果 into outfile “目录名/文件名”Fields terminated by “分隔符” lines terminated by “\n”

mysql> select \* from user into outfile "/mydatadir/a1.txt";

mysql> select name,uid,shell from user where id <= 5 into outfile "/mydatadir/a3.txt";

mysql> select name,uid,shell from user where id <= 5 into outfile "/mydatadir/a4.txt" fields terminated by "#" lines \ terminated by "!!";

**复制表（快速建表）：备份表与快速建表**

**1.1:将源表xxx复制为新表yyy 格式:crate tabe yyy select \* from xxx**

mysql> create table userbak select \* from db3.user; #将db3库的user表,复制到db4库userbak表

**1.2:将指定查询结果复制为新表zzz**

mysql> create table user3 select name,uid,shell from db3.user order by uid desc limit 5;

mysql> select \* from user3;

**1.3:复制源表xxx的结构到新表vvv**

例如:将db3库的user表结构,复制到db4库的user1表,主键/索引等无法复制过来

mysql> create table user1 select \* from db3.user where 1=2;

mysql> select \* from user1;

例如:复制部分字段到新表

mysql> create table user4 select name,uid,shell from db3.user where 1=2;

**1.4:将源表vvv 的名称改为www**

Alter table vvv rename to www;

**mysql数据库调优：**

1. 提高mysql系统的硬件性能,响应速度
2. 替换有问题的硬件(cpu/磁盘/内存等)
3. 服务程序的运行参数调整
4. 对sql查询进行优化
5. 网络带宽,网络架构
6. 查看数据库相关配置
7. 连接数/并发量
8. 连接超时
9. 缓存相关参数
10. 线程参数
11. 存储引擎优化
12. 文件打开数(open\_files)：发现当Open\_files大于open\_files\_limit值时，MySQL数据库就会发生卡住的现象，导致Nginx服务器打不开相应页面。
13. 慢查询日志：超过指定时间的SQL语句查询称为“慢查询”

开启慢查询日志，将慢查询语句写入文件中。再针对文件中的慢查询语句进行优化。

[mysqld]

slow-query-log=1 #启用慢查询，1为开启，0为关闭

slow-query-log-file=/var/log/\*\* #指定慢查询日志路径和名字

long-query-time=5 #设置慢查询超时时间，超过这个时间的查询将被记录

log-queries-not-using-indexes #记录未使用索引的查询

1. 索引优化

为经常用来做搜索的字段建立索引吧。建了索引，没有建索引，性能差了4倍左右。

从上图你可以看到那个搜索字串 “last\_name LIKE ‘a%'”，

1. 优化程序元访问数据的sql查询命令，可用explan查询 >
   1. explain select \* from mysql.user
   2. 尽量避免select \*
2. 设置合理的表结构，选择合适的字符类型
3. 合理垂直分隔与水平分隔分库分表（mycat）：
   1. 表越小响应越快，
   2. 分片规则有10几种，如枚举法，固定分片，求模等等
4. 读写分离（maxscale）：减少服务器负载
5. 安装maxscale软件，提供管理命令
6. 修改maxscale配置文件，添加主从库信息，定义监控的数据库信息，添加监控用户，定义读写分离器
7. 选择合理的存储引擎（engine） ：查看库默认使用的存储引擎show variables like '%storage\_engine%';
   1. Mysiam:表级锁，适合读比较多的，不支持事务
   2. Innodb :行级锁，适合写比较多的，支持事务
8. 监控是否有大量无效查询与访问IP，例如有大量爬虫，

**一、按数据库运行状态分：**

**(1)热备：**在数据库运行时，直接进行备份，对运行的数据库没有影响

**(2)冷备：**在数据库停止运行的时候进行备份，这种备份方式最为简单，只需要拷贝数据库物理文件即可。

**(3)温备：**同样是在数据库运行的时候进行备份的，但对当前数据库的操作会产生影响。

热备份的优点：1.可在表空间或数据文件级备份，备份时间短。

2.备份时数据库依然可以使用。

3.可达到秒级恢复，能够恢复到某一个时间点上。

4.恢复的速度很快，在大多数情况下在数据库工作时就可恢复。

5.几乎所有的数据库实体都可以进行恢复。

热备份的缺点：1.尽量不要出错，否则后果会很严重。

2.如果热备份不成功，所得结果不可用于时间点的数据恢复。

3.维护的工作比较困难。

冷备份的优点：1.是非常快速的备份方法，因为只需要拷贝文件即可

2.容易归档，容易恢复到某个时间点上(只需将文件再拷贝回去即可)

3.能与归档方法相结合，作数据库（最新状态）的恢复。

4.容易维护，且比较安全。

冷备份的缺点：1.单独使用时，只能提供到"某一时间点的上"的恢复。

2.在实施备份的全过程中，数据库必须是关闭状态。

3.不能按表或按用户恢复。

**注意：**冷备份必须在数据库关闭的情况下进行，当数据库处于打开状态时，执行数据库文件系统备份是无效的。而且在恢复后一定要把数据库文件的属组和属主改为mysql。

**二、按照备份方式分类：**

**物理备份**：直接复制数据文件进行的备份

优点：不需要其他的工具，直接复制就好，恢复直接复制备份文件即可

缺点：与存储引擎有关，跨平台能力较弱

**逻辑备份：**从数据库中导出数据另存而进行的备份

优点：能使用编辑器处理，恢复简单，能基于网络恢复，有助于避免数据损坏

缺点：备份文件较大，备份较慢，无法保证浮点数的精度，使用逻辑备份数据恢复后，还需要手动重建索引，十分消耗cpu资源。

**三、按照业务划分**

**(1)完全备份：**即对整个数据库的数据和数据结构进行备份。

优点：很直观容易被人理解。而且当发生数据丢失的灾难时，只要用灾难之前的备份文件，就可恢复丢失的数据。

缺点：首先由于每天都对系统进行完全备份,因此在 备份数据中有大量是重复的。这些重复的数据占用了大量的空间， 这对用户来说就意味着增加成本；其次，由于需要备份的数据量相当大，因此备份所需时间较长。对于那些业务繁忙，备份窗口时间有限的单位来说，选择这种备份策略无疑是不明智的。

**(2)增量备份：**每次备份的数据只是相当于上一次备份后增加的和修改过的数据。

优点：很明显：没有重复的备份数据，即节省空间，又缩短了备份时间。

缺点：在于当发生灾难时，恢复数据比较麻烦。

举例来说：如果系统在星期四的早晨发生故障，丢失大批数据，那么现在就需要将系统恢复到星期三晚上的状态。 这时管理员需要首先找出星期一的那份完全备份数据进行系统恢复，然后，再找出星期二的数据来恢复星期二的数据，然后在找出星期三的数据来恢复星期三的数据。很明显这比第一种策略要麻烦得多。另外这种备份可靠性也差。在这种备份下，各备份数据间的关系就象链子一样，一环套一环其中任何一个备份数据出了问题都会导致整条链子脱节。

**(3)差异备份：**每次备份的数据是相对于上一次全备份之后新增加的和修改过的数据。管理员先在星期一进行一次系统完全备份；然后在接下来的几天里，管理员再将当天所有与星期一不同的数据（新的或经改动的）备份到磁带上。

举例来说，在星期一，网络管理员按惯例进行系统完全备份；在星期二，假设系统内只多了一个资产清单，于是管理员只需将 这份资产清单一并备份下来即可；在星期三，系统内又多了一份产品目录，于是管理员 不仅要将这份目录还要连同星期二的那份资产清单一 并备份下来。如果在星期四系统内有多了一张工资表那么星期四需要备份的内容就是：工资表+产 品目录+资产清单。

**数据库备份与恢复**

**Mysqldump工具完全备份：**

1、mysqldump -u用户 -p密码 数据库名 >备份文件名.sql #备份单个数据库

2、mysqldump -u用户 -p密码 all >备份文件名.sql #备份所有的数据库

[root@dbsvr1 ~]# mysqldump -u root -p --all-databases > /root/alldb.sql

3、mysqldump -u用户 -p密码 数据库名 表名 >备份文件名.sql #备份单张表，数据库名表名不带点、

[root@sql51 mysqlbak]# mysqldump -uroot -p654321 db3 user >/mysqlbak/db3\_user.sql

4、mysqldump -u用户 -p密码 -B 数据库名1 数据库名2 表名 >备份文件名 #备份多个数据库，加-B

[root@sql51 mysqlbak]# mysqldump -uroot -p654321 -B db2 db4 >/mysqlbak/twodb.sql

**Mysqldump工具完全恢复：**恢复数据表需要加库名

恢复数据库之前,要先创建新库,备份数据库时用 -B 备份的多个数据库的,恢复时可以省略数据库名

1、mysql -u用户 -p密码 < 备份文件名 #用户必须有登录备份权限

[root@sql53 ~]# mysql -uroot -p654321 </root/twodb.sql #备份时用-B备份的，恢复时省略了数据库名

2、mysql -u用户 -p密码 库名 <备份文件名 #用户必须有登录备份权限,库名可以为新库名

[root@sql53 ~]# mysql -uroot -p654321 db3 < /root/db3\_user.sql #将db3\_user.sql恢复到db3库中

**Binlog日志增量备份:**

[root@sql51 ~]# mysqlbinlog /logdir/plj.000002 #查看日志文件内容

[root@sql51 ~]# mysqlbinlog /logdir/plj.000001 | grep insert #查看日志文件内容并筛选

[root@sql52 ~]# mysqlbinlog /root/plj.000001 | mysql -uroot -p654321 #恢复日志内所有语句

[root@sql52 ~]# mysqlbinlog --start-position=560 --stop-position=924 /root/plj.000001 |mysql -uroot -p654321 #恢复指定部分范围内的数据，根据偏移量

\*\*选择范围必须包含日志文件的commit \*\*不指定开头,默认从最前面开始 \*\*不指定结束,默认执行到最后

**Innobackupex备份：（需要安装xtrabackup工具）**

[root@sql51 ~]# yum -y install libev-4.15-1.el6.rf.x86\_64.rpm

[root@sql51 ~]# yum install -y percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm

**数据完全备份与恢复**

Innobackupex恢复数据要求数据库清空mysql目录,因此那怕你只需要备份多个库中其中一个库,也要将数据库三个系统库(mysql sys performance\_schema)一起备份

完全备份案例:将所有库完全备份到/backup(目录不需事先创建,备份程序自动创建)

格式:Innobackupex --user root --password 123456 /allbak -no-timestamp

Innobackupex 选项 --host 主机名 --user 用户名 --port 端口号(默认3306可不用写) --databases 数据库名(不写将备份所有数据库) --no-timestamp 不用时间戳,不用日期命名备份文件存储的子目录名

Innobackupex 备份之后有两种文件1是数据文件,2是备份信息文件

1数据文件,用来恢复数据

2备份配置文件:有备份信息,用来做增量备份或者差异备份等的参考

**主从配置：**从库的io线程读取主库binlog日志写入到中继日志，sql线程读取中继日志并执行sql语句，达到主从

**MHA集群：**MHA由perl语言编写而来.所以所有主机都要安装

必须先装数据节点安装包:mha4mysql-node-0.56-0.el6.noarch.rpm

再安装管理包:mha4mysql-manager-0.56.tar.gz

[root@sql56 mha4mysql-manager-0.56]# perl Makefile.PL #检查环境

[root@sql56 mha4mysql-manager-0.56]# make && make install

拷贝配置模板到/etc/mha\_manager 再配置，

配置1：指定工作目录，指定日志所在文件，指定加载的故障切换脚本

配置2：添加主从同步用户、监控数据用户、设置免密认证用户和端口、指定主从以及设置候选master

部署故障切换脚本、在主库上配置VIP，添加管理用户，管理主机测试免密连接

管理机查看服务状态：56 ~]# masterha\_check\_status --conf=/etc/mha\_manager/app1.cnf

管理机监控]#masterha\_manager --conf=/etc/mha\_manager/app1.cnf --remove\_dead\_master\_conf --ignoer\_last\_failover

恢复：将宕机主机恢复数据，并加入新的主从，mha配置文件从新添加主机信息，管理主机从新监控：