set global validate\_password\_policy=0;

set global validate\_password\_length=6

alter user user() identified by "123aaa...A";

RDBMS 关系型数据库服务软件

(需要提前搭建库表结构)

NOsqL非关系型数据库服务软件

(不需要预先定义数据存储结构)

(每条记录可以有不同的数据类型和字段个数)

Nosql软件(Memancached redis mongoDB couchDB )

Day01

1.LAMP平台,与Apache HTTP Server组成

LNMP平台,与Nginx组成

1. 软件安装后自动创建相关目录与文件

主配置文件 /etc/my.cnf

数据库目录 /var/lib/mysql

默认端口号:3306

随机密码存放在日志文件 /var/log/mysqld.log

1. 数据库基本管理

DDL数据定义语言:create alter drop

DML数据操作语言:insert update delete

DCL数据控制语言:grant rewoke

DTL数据事物语言:commit rollback savepoint

1. 表管理命令

建表:create table 库名.表名(字段名1 类型(宽度)...)

Default charset=utf8;指定中文字符

修改表记录:update 库名.表名 set 字段=值;

Delete from 表名;

1. 字符类型

定长:Char(字符个数) 最大字符个数255,不够指定字符个数在右边用空格

补全,个数超出时,无法写入数据;

变长:varchar(字符个数) 按数据实际大小分配存储空间,个数超出无法写入

大文本类型:text/blob 字符数大于65535存储时使用

1. 数值类型

**整数型**:tinyint 微小整数 0-255

Smallint 小整数 0-65535

Mediumint 中整型

Int 大整型

Bigint 极大整型

Unsigned 使用无符号存储范围

**浮点型:**float 单精度

Double 双精度

格式:1.字段名 类型; (pay float,id double)

2.字段名 类型(总宽度,小数位数);

(pay float(5,2),id double(4,2));

1. 日期时间类型:

日期时间:datatime(当未赋值时,以当前系统时间赋值)

Timestamp(当为赋值时,其值为null)

日期: data

年:year 要求用4位数赋值

时间:time

1. 枚举类型

enum单选:格式:字段名 enum(值1,值2,值N) 仅选择一个值 必须在列表里选

Day04

1. 用户授权

1.1定义:在数据库服务器上添加客户端连接时使用的用户，并设置访问权限。

1.2用户授权的原因:默认只允许数据库管理员root用户本机登陆

1.3配置用户授权的命令格式:

Grant 权限列表 on 数据库名 to 用户名@’客户端地址’identified by ‘密码’ [ with grant option ] ;

权限列表 个别权限:select , insert ,update ,delete update(name,uid)

所有权限:all

无权限:usage

数据库名 库名.表名 db3.user

某个库 db3.\*

所有库/表 \*.\*

用户名 客户端连接时使用的名字，授权时自定义即可

客户端地址 本机 localhost

某1台主机 192.168.4.51

某个网段 192.168.4.%

所有主机 %

With grant option 可选项,授权用户有执行grant命令的权限

相关命令举例:select user();

show grants;

set password=password("123aaa...A");

1.4授权库mysql:记录授权信息

user表 记录已有的授权用户及权限

db表 记录已有授权用户对数据库的访问权限

tables\_priv表 记录已有授权用户对表的访问权限

columns\_priv表 记录已有授权用户对字段的访问权限

常用命令: grant all on \*.\* to root@"%" identified by "123qqq...A" with grant option;

mysql> select host ,user from mysql.user;

mysql>select\*from mysql.user where user="root" and host="localhost"\G;

mysql> show grants from root@"%";

mysql>update mysql.tables\_priv set Table\_priv="select,insert,delete"

where user="yaya99" and host="192.168.4.51";(更改权限)

1.5撤销权限:删除已有授权用户的权限

命令格式:revoke 权限列表 on 库名.表 from用户名@"客户端地";

1.6 root密码

真机上操作vim /etc/my.cnf

[mysqld]

skip-grant-tables 重启 无密码进入mysql

修改密码:

mysql>updatemysql.userset authentication\_string=password("123qqq...A")

where user="root" and host="localhost" ;

mysql> flush privileges;

mysql> exit

注释真机上文件内容

1. 完全备份

1.备份方式:物理备份与逻辑备份

2.备份策略:完全备份 增量备份 差异备份

3.物理备份与恢复:

完全备份 命令格式

]#mysqldump -uroot -p密码 库名 > 目录/xxx.sql

备份时库名表示方式：

--all-databases 或 -A //所有库

数据库名 //单个库

数据库名 表名 //单张表

-B 数据库1 数据库2 //多个库

完全恢复命令格式

]#mysql -uroot -p密码 [ 库名 ] < 目录/xxx.sql

1. 增量备份:启用mysql服务的binlog日志文件实现对数据的 时时增量 备份

四.binlog日志

1.Vim /etc/my.cnf 添加server\_id=本机号

Log\_bin=/mylog/文件名

添加mysql权限 重启 ls /mylog/文件名.\*

2.进入数据库查看正在使用的日志: show master status;

3.在系统命令行操作数据库里命令: 进入数据库命令 -e 数据库里命令

4.手动创建新的日志文件的四种方式:

systemctl restart mysqld

Mysql>flush logs;

Mysql -uroot -p密码 -e ‘flush logs’

Mysqldump -uroot -p密码 --flush-logs 库名 > /dbbak/库名.sql

1. 删除的两种方式:当前所有的mysql>reset master;

删除指定的mysql> purge master logs to ‘文件名’;

1. 修改日志格式:vim /etc/my.cnf 添加binlog\_format=””

日志记录的三种方式:**报表模式**statement**行模式**row **混合模式** mixed

查看日志记录内容:mysqlbinlog /mylog/ly.000002 | grep -i update

1. 日志文件如何区分记录多条sql命令(日志记录sql命令的方式)

偏移量 --start-position=数字 --stop-position=数字

时间点 --start-datetime=”年与日时分秒” --stop-datetime=””

1. 读取日志指定范围的sql命令恢复数据

scp /mylog/ly.000003 [root@192.168.4.51:/opt/](mailto:root@192.168.4.51:/opt/)#从服务机传到客户

恢复:mysqlbinlog --start....../opt/文件 | 进入数据库

Day05 DBA基础

1. 数据备份与恢复
2. 软件介绍
3. 软件安装:percona.... libev....
4. 命令格式:innobackupex <选项>
5. 常用选项:--incremental 目录名 增量备份
6. 备份命令格式:

**完全备份与恢复**

**增量备份与恢复**

**恢复单张表**

5.1完全备份:innobackupex --user root --password 123aaa...A /allback --no-timestamp #自动备份到并创建/allback目录

▲数据完全恢复步骤:(在客户端另一个机器上)

停止数据库服务ssystemctl stop mysql

清空数据库目录rm -rf /var/lib/mysql/\*

准备恢复数据ls /opt/allback

cat /opt/allback/xtrabackup\_checkpoint中type=full\_backup

拷贝本分数据到数据库目录下innobackupex --apply-log /opt/allback/

cat /opt/allback/xtrabackup\_checkpoint 中type=full\_prepared

修改目录的所有者/组用户为mysql chown -R mysql:mysql /var/lib/mysql

启动服务 systemctl restart mysqld

管理员登录查看服务器

5.3使用完全备份文件,恢复单张表的所有数据

备份文件innobackupex --user root --password 123aaa...A --databases="db5" /db5bak --no-timestamp

▲恢复单张表的所有数据步骤:(在数据库本机操作)

删除表空间文件alter table 库.表 discard tablespace;

在备份文件里导出表信息 innobackupex --apply-log --export /db5bak

把表信息文件拷贝到数据库目录下 cp /db5bak/db5/b.{cfg,exp,ibd} ` /var/lib/mysql/db5/

导入表信息alter table db5.b import tablespace;

删除表信息文件rm -rf /var/lib/mysql/db5/b.cfg 和b.exp

查看表记录select \* from b;

5.4 增量备份与恢复(备份50 恢复51)

**增量备份**定义:备份上次备份后,所有新产生的数据

首次备份innobackupex --user root --password 123aaa...A /fullbak --no-timestamp

增量备份:innobackupex --user root --password 123aaa...A

-incremental /new1dir --incremental-basedir=/fullbak

--no-timestamp (插入信息记录)

再次增量备份:innobackupex --user root --password 123aaa...A

-incremental /new2dir --incremental-basedir=/new1dir

--no-timestamp (插入信息记录)

xtrabackup\_checkpoints 存放备份信息的文件

**增量恢复的步骤**

停止数据库服务

清空数据库目录 rm -rf /var/lib/mysql/\*

准备恢复数据

innobackupex --apply-log --redo-only /opt/fullbak/

Innobackupex --apply-log --redo-only/opt/fullbak/

--incremental-dir=/opt/new1dir

Innobackupex --apply-log --redo-only/opt/fullbak/

--incremental-dir=/opt/new2dir

合并数据

拷贝数据到数据库目录下innobackupex --copy-back /opt/fullbak/

修改数据库目录为MySQLchown -R mysql:mysql /var/lib/mysql

启动服务

查看数据

**DAY01**

50 客户端

51 主服务器 启用binlog日志 授权用户 查看binlog日志信息

52 从服务器 设置server\_id 指定主库信息 启动slave程序 查看状态信息

**配置主服务器**

修改配置文件 vim /etc/my.cnf

Server\_id=51

Log\_bin=master51 #启用binlog日志

重启服务 systemctl restart mysqld

查看信息 ls /var/lib/mysql/master51.\*

进入数据库授权用户 grant replication slave on \*.\* to repluser@’%’

Identified by ‘123qqq...A’;

查看日志信息 show master status\G;

**配置从服务器**

修改配置文件 vim /etc/my.cnf

Server\_id=52

重启配置文件 systemctl restart mysqld

主库备份数据

mysqldump -uroot -p123qqq...A --master-data db5 > /root/db5.sql

传给从服务器 scp /root/db5.sql [root@192.168.4.52:/opt/](mailto:root@192.168.4.52:/opt/)

从服务器恢复数据 mysql -uroot -p123qqq...A db5 < /opt/db5.sql

**二.主从同步模式**

**1.配置mysql一主多从结构(51主 52从 53从)**

要求:把数据库服务器53 也配置

修改配置文件vim /etc/my.cnf

Server\_id=53

重启配置文件 systemctl restart mysql

主库备份文件:传给从服务器53

mysqldump -uroot -p123qqq...A --master-data bbsdb > /root/bbsdb.sql

从服务器恢复数据 mysql -uroot -p123qqq...A -e 'create database bbsdb'

导入数据到库 mysql -uroot -p123qqq...A bbsdb < /root/bbsdb.sql

(主)查偏移量grep master51 /root/bbsdb.sql

(从)进入数据库操作change master to master\_host=’192.168.4.51’,

Master\_user=’repluser’,

Mater\_password=’123qqq...A’,

Master\_log\_file=master51.000001’,

Master\_log=偏移量;

启动slave

显示slave状态: show slave status\G;

**◆附加操作:将其他结构中的从服务器还原**

关闭mysql服务 cd /var/lib/mysql

删除host53-relay-bin.\*

relay-log.info|master

Master.info 重启服务

进入到数据库删除多余的数据

**2.配置mysql主从从结构(53做54的主 54做53的从做55的主 55做54的从 )**

**(53主服务器)**修改配置文件 vim /etc/my.cnf

Server\_id=53 log-bin=master53

重启服务 systemctl restart mysql

授权用户 grant replication slave on \*.\* to repluser@"%" identified by

"123qqq...A";

查看状态

**(54主从服务器)**修改配置文件vim /etc/my.cnf

Server\_id=54

log-bin=master54

log\_slave\_updates #允许级联复制

重启服务

进入数据库授权:grant replication slave on \*.\* to repluser@’%’

identified by ‘123qqq...A’;

设置:change master to master\_host=”主服务器地址”,

master\_user=’repluser’,

Master\_password=’密码’,

Master\_log\_file=master53 #binlog日志文件名

Master\_log\_pos=偏移量

启动slave服务:start slave;

检查状况: show slave status\G;

**(55从服务器)**

1. 配置mysql主主结构(两台数据库服务器互为主从)

要求:把数据库服务器192.168.4.56和192.168.4.57

1. **主从同步复制模式**

配置半同步模式 192.168.4.54

命令行配置 1.查看是否允许有动态加载模块

Show variables like “have\_dynamic\_loading”; (显示on为正确)

2.命令行加载插件

Install plugin rpl\_semi\_sync\_master SONAME “semisync\_master.so”;

Install plugin rpl\_semi\_sync\_slave SONAME “semisync\_salve.so”;

3.查看模块是否加载

select plugin\_name, plugin\_status from

information\_schema.plugins

where plugin\_name like '%semi%';

4.启用半同步复制

set global rpl\_semi\_sync\_master\_enables=1 执行主服务器

set global rpl\_semi\_sync\_slave\_enables=1 执行从服务器

5.查看是否启用半同步复制

Show variables liek ‘rpl\_semi\_sync\_%\_enavled’;

6.写入永久配置文件 /etc/my.cnf

在[mysql]下面添加

select user.host from mysql.user;

1.4排错

last\_IO\_Error:IO线程报错信息

last\_SQL\_Error:SQL线程报错信息

排错方法:改之前停止stop slave;根据提示解决问题

终极解决办法:删除相关的四种文件后重启数据库服务,重新配置主服务器信息

**###################DAY02#################**

**一.构建读写分离思路**

1. 部署mysql 主从结构(主 从服务器)
2. 部署MySQL代理服务器(装包 修改配置文件 启动服务)

a.在代理机上安装max.....软件部署maxscale服务

b.查看服务安装位置及信息

c.修改配置文件 /etc/maxscale.cnf

server部分 修改server部分address=主机ip地址

复制server部分 修改server标题部分address=slave主机IP地址

修改[mysql monitor]部分 servers=主从数据库的主机名

user=maxscalemon #监控用户

Password=123qqq...A

注释掉[read-only service]部分

修改[read-write service]部分

password=密码

server=主从数据库的主机名

user=授权进入数据库的用户名

注释掉[read-only listener]部分

修改[myxadimin listener]部分

加入行port=端口号(没有分配出去的端口)

◆ **sed -i '/#/d' /etc/maxscale.cnf (去除文件里带有注释的行)**

d.配置数据库服务器(创建授权用户)

grant replication slave ,replication client on \*.\* to

maxscalemon@'%' identified by '123qqq...A'

grant select on mysql.\* to maxscalerouter@'%'

identified by '123qqq...A';

启动服务 maxscale -f /etc/maxscale.cnf

查看服务信息 netstat -utnlp | grep maxscale

查看监测信息 maxadmin -uadmin -pmariadb -P4016

List servers(进入程序后操作)

停止服务 kill -9 pid号

3.测试配置(客户端连接代理服务访问数据)

在主服务器51上建库表 并且添加客户端访问数据的连接用户

52查看授权用户信息

客户端链接代理服务器57 访问数据

mysql -h192.168.4.57 -P4006 -uliuliu -p123qqq...A

查看权限 插入数据 查看表

**数据库进阶DAY03 数据分片**

1. 相关概念

对于mycat

1. 一个彻底开源的，面向企业应用开发的大数据库集群
2. 支持事务、ACID、可以替代MySQL的加强版数据库
3. 一个可以视为MySQL集群的企业级数据库，用来替代昂贵的Oracle集群
4. 一个融合内存缓存技术、NoSQL技术、HDFS大数据的新型SQL Server
5. 结合传统数据库和新型分布式数据仓库的新一代企业级数据库产品
6. 一个新颖的数据库中间件产品
7. Mycat软件介绍

为什么要用Mycat? 如今随着互联网的发展，数据的量级也是撑指数的增长，从GB到TB到PB。对数据的各种操作也是愈加的困难，传统的关系性数据库已经无法满足快速查询与插入数据的需求。这个时候NoSQL的出现暂时解决了这一危机。它通过降低数据的安全性，减少对事务的支持，减少对复杂查询的支持，来获取性能上的提升。但是，在有些场合NoSQL一些折衷是无法满足使用场景的，就比如有些使用场景是绝对要有事务与安全指标的。这个时候NoSQL肯定是无法满足的，所以还是需要使用关系性数据库。如何使用关系型数据库解决海量存储的问题呢？此时就需要做数据库集群，为了提高查询性能将一个数据库的数据分散到不同的数据库中存储，为应对此问题就出现了——MyCat

#### Mysql作用 : 能满足数据库数据大量存储；提高了查询性能

#### Mycat支持提供10种分片规则:枚举法 固定分片 求模法 日期列分区法

1. 配置数据分片
2. 安装软件(需要有Java命令 将软件的配置移到/use/local/目录下)

bin #mycat命令

Catlet #扩展功能

Conf #配置文件

Lib #mycat使用的jav包

Logs #mycat启动日志和运行日志

Wrapper.log #mycat服务启动日志

Mycat.log #记录sql脚本执行后的保错内容

1. 配置文件说明

Server.xml #设置链接账号及逻辑库

Schema.xml #配置数据分片

Rule.xml #分片规则

其他文件 #函数调用文件

3.修改配置文件

配置客户端连接时使用的用户和密码 vim /conf/server.xml (使用默认配置)

配置数据分片<schema>....</schema> 定义分片信息

<table>....</table> 定义表

Name 逻辑库名或逻辑表

Datanode 指定数据节点名

Rule 指定使用的分片规则

type=global 数据不分片存储

修改文件 vim /conf/schema.xml

sed -i '56,77d'schema.xml sed -i '39,42d' schema.xml

sed -i '16,18d' schema.xml sed -i '45,47d' schema.xml

修改table name=部分 添加dn3

修改datanode name 部分修改datahost主机名

修改datahost name部分 修改writehost里的url user

Password #需要在数据里授权

复制三部分datahost部分 进行修改:

配置数据库服务器53.4.5.(创建存储库 用户授权)

(53)Create database db1

Grant all on \*.\* to admin@’%’identified by ‘密码’

(54)Create database db2

Grant all on \*.\* to admin@’%’identified by ‘密码’

(55)Create database db3

Grant all on \*.\* to admin@’%’identified by ‘密码’

1. 启动服务

/usr/local/mycat/bin/mycat start

1. 查看服务信息

查看进程 ps -C java

查看端口 netstat -ntulp | grep java

日志文件 ls /usr/local/mycat/logs

1. 排错时间
2. 分片规则的使用(**端口8066**)

**枚举法**

在rule.xml文件里找到<function name="hash-int"class=

"io.mycat.route.function.PartitionByFileMap">

<property name="mapFile">

partition-hash-int.txt</property>部分

修改蓝色字文件 vim /usr/local/mycat/conf/partition-hash-int.txt

添加10020=2 有多台依次添加

创建表create table employee (id int primary key auto\_increment,

sharding\_id int, name char(15),age tinyint,sex,

enum("man","womam"));#代理机器上

插入数据 insert into employee(sharding\_id,name,age,sex)

values(10000,"lucy","19","womam");#对应sharding号存在对应数据机

**Mod-long 求模法:根据字段与设定的数字求模结果存储数据**

Vim schema.xml 里删除tablename=hostnews部分的id 和icrenment

重启服务

**Type=global数据不分离存储**

八.建新库新表

**添加新库** 在server.xml文件里添加库名

**添加新表** 在schema.xml文件里复制schem部分 添加表名(复制下面的)

#库名表名不能一样

**重启服务**

**测试配置**

##############DBA day04############################

1. **相关概念**

集群分类:LB（负载均衡集群）：LVS Nginx HAproxy

　　　　　　HA（高可用集群 ）:keepalived

HPC():用在专业领域

1. **MHA介绍(mysql高可用集群)**

MHA Manger(管理节点):单独部署在一台独立的机器上

MHA Node（数据节点）:运行在每台mysql服务器上

1. **环境准备**
2. 准备集群环境

a.安装依赖包

b.配置所有数据库之服务器之间彼此ssh免密登录 ssh-keygen

Ssh-copy-id root@地址

c.配置一主多从结构

配置主库51

配置 从库

1. 配置管理节点(进入到mha....)

▲ yum -y install perl-DBD-mysql perl-DBI perl-ExtUtils-\* perl-CPAN\*

rpm -ivh mha4mysql-node-0.56-0.el6.noarch.rpm

tar -zxvf mha4mysql-manager-0.56.tar.gz

perl Makefile.PL

Make

Make install

附加知识点:ls /root/perl5/bin/masterha\_\*

Masterha\_check\_ssh 检查MHA的SSH配置状况

Masterha\_check\_repl 检查mysql复制状况

Masterha\_manager 启动MHA

Masterha\_check\_status 启动MHA运行状态

Masterha\_stop 停止MHA

▲编写主配置文件

mkdir /etc/mha

cp samples/conf/app1.cnf /etc/mha

Vim /etc/mha/app1.cnf

修改第一部分的workdir=自己创建的目录 #工作目录

Log=自己创建目录下的manager.log #日志文件

加master\_ip\_failover\_script=/etc/nha/master\_ip\_failover #故障切换脚本

添加行repl\_user=repluser

repl\_password=123qqq...A

ssh\_user=root

ssh\_port=22

user=root

password=123qqq...A

修改server部分

▲创建故障切换脚本

[root@host57 mha-soft-student]# cp master\_ip\_failover /etc/mha/

chmod +x /etc/mha/master\_ip\_failover

修改文件35行 的vip地址

1. 配置数据节点

安装mha\_node软件(3台数据库)

用户授权

启用半同步复制模式

配置主服务器,修改数据库服务运行参数

启用master模块 slave模块 禁止自动删除中继日志

配置从服务器,修改数据库服务运行参数

1. 创建MAH
2. 测试配置
3. 测试集群功能

#################################DAY05###########################

进程:独占资源,独立去完成工作

线程:是进程的最小单元,多个线程可以同时工作,共享资源.

1. PXC概述

galera产品是以galera cluster方式为mysql提高高可用集群解决方案的。galera cluster就是集成了galera插件的mysql集群。galera replication是codership提供的mysql数据同步方案，具有高可用性，方便扩展，并且可以实现多个mysql节点间的数据同步复制与读写，可保障数据库的服务高可用及数据强一致性。

PXC属于一套近乎完美的mysql高可用集群解决方案，相比那些比较传统的基于主从复制模式的集群架构MHA和MM+keepalived，galera cluster最突出特点就是解决了诟病已久的数据复制延迟问题，基本上可以达到实时同步。而且节点与节点之间，他们相互的关系是对等的。

PXC特点:数据强一致性,无同步延迟,没有主从切换操作,无需使用虚拟ip

支持innoDB存储引擎,支持多借点自动接入,无需手动拷贝数据.

相应端口:3306数据库服务端口 4444 sst端口(全量同步)

4567集群通信端口 4568IST端口(增量同步)

1. 部署PXC

1.主机名绑定

2.安装软件

3.配置服务 ls /etc/percona-xtradb-cluster.conf.d/ # 所在目录

Msyqld.cnf #服务运行参数配置文件 (仅修id号)

Wsrep.cnf #集群配置文件

修改第8行所有成员的地址 本机地址放在最后

修改第25行 主机地址

修改第27行集群名

修改 30行主机名

修改39行密码和用户

4.启动服务(在任意一台即可)(此操作中的mysql没有d)

Systemctl restart mysql@bootstrap.service

使用初始密码登录 修改登录密码

添加配置文件里的授权用户

查看端口3306(数据库) 4567(集群通讯端口)

其他机器直接启动mysql查看端口就可以

1. mysql存储引擎

表文件 表名.frm(表结构) 表名.MYI(索引) 表名.MYD(数据)

1. 存储引擎介绍

show variables like "%version%"; #版本变量查看

八部分:sql接口 解析器 优化器 缓存器

2.使用存储引擎:

修改数据库默认引擎: vim/etc/my.cnf文件中添加

default-storage-engine=myisam(存储引擎)

重启进入数据库 show engines;

进入数据建表指定存储引擎create table db10.c(age int)engine=inoodb;

修改表的存储引擎 alter table db10.e engine=myisam;(此时表文件变为3个)

查看表使用的存储引擎show create table e;

1. Myisam存储引擎特点:支持表级锁

不支持事务 事务回滚 外键

1. 支持事务表的事务特性:原子性 一致性 隔离性 持久性

############################**NoSQL\_day01#**#####################

**概述：**

搭建redis服务器　redis服务日常常用配置项目　　LNMP+Redis

搭建redis集群和管理集群　redis主从复制　哨兵服务　AOF RDB

Redis数据类型

1. **搭建redis服务器(端口号6379)**

1.相关概念

2.redis介绍:高性能的分布式内存数据库,支持数据持久化(定期把内存里

数据存储到硬盘),支持多种数据类型hash...

支持master-slave模式数据备份.

3.搭建数据库服务器redis服务

安装软件

启动服务之前要初始化配置

在redis目录下执行 ./utils/install\_server.sh

初始化完自动启动服务

Redis-cli进入命令行(输入ping 出现pong)

命令行输入:keys \* #查看当前的数据

Set name bob /get name #存放在当前主机的物理内存

/etc/init.d/redis\_6379 stop #停止服务

ls /var/lib/redis/6379/ #停止服务后内存数据所

存在的位置

4.数据的常用管理命令

exists 名 #测试key名是否存在,返回值为1是存在

0是不存在

ttl 名 #查看key生存时间,返回值为-1是永久存在

-2是已过期 不存在

Type 名 #查看key类型,返回值为string是数值类型

expire 名 数字 #设置key有效时间 以秒为单位

flushall #删除内存里所有key

Flushdb #删除所有库的所有key

Keys ? #显示当前库中名为一个字符的库.依次.

5.配置文件常用配置 /etc/redis/6379.conf

修改501行:密码

修改70行:本机ip

修改93行:端口(没有被分配到应用的随机端口)

6.重启 进入命令行:redis-cli -h IP地址 -p 配置文件里的端口 -a 密码

或者进入命令行输入:auth 密码

停止服务:redis-cli -h ip地址 -p 6350 -a 密码 shutdown

修改配置文件可以用脚本停止 vim /etc/init.d/redis\_6379

$CLIEXEC -h 192.168.4.50 -p6350 -a 123456 shutdown

1. **部署LNMP+redis**(mysql nginx phph-fpm)
2. 构建环境

测试文件是否正确 /usr/local/nginx/sbin/nginx -t

测试能否连接redis php -m | grep -i redis

2.配置PHP支持redis

a)安装软件(执行phpize生成配置文件) 安装php扩展

b)修改配置文件 vim /etc/php.ini

728行 /usr/lib64/php/modules/

730行 删掉\_dir 改成redis.so

c)编写测试脚本

#####################**NoSQL\_day02#############################**

**一.创建集群**

1.3 启用集群功能:需要配置文件 /etc/redis/6379.conf

815行 去掉注释

823行 去掉注释

829行 改成5000

重启

在建集群前需清空目录下所有 Rm -rf /var/lib/redis/6379/\*

Cluster info 查看集群状态(在任意一台redis机器)

1.4 查看集群

Redis-trib.rb info ip地址:6354

Redis-trib.rb check ip地址:6354

1.5访问集群存取数据

Redis-cli -c -h 192.168.4.52 -9 6352

1.6

1. **管理集群**

2.1 向集群中添加新主机(在57集群管理主机上操作)

添加master角色主机

要求把新的机器的redis服务器添加到集群里做主服务器

1.运行redis服务并且支持集群功能

2.添加master角色的主机

添加主机到集群add-node

分配hash槽 reshard

3.查看集群信息

4.访问集群存取信息

添加slave角色主机

要求把新机器的redis服务器添加到集群里做从服务器

1.运行 redis服务器并且支持集群功能

2.添加slave角色的主机

Add-node --slave ]

3.查看集群状态

2.2把主机移除集群

1)移除slave服务器

redis-trib.rb del-node 主服务器iP:端口 被移除的slave机器ip

2)移除master主机

指定移除slots 个数

redis-trib.rb reshard 192.168.4.58:6358(问题中输入info所

显示的个数 4096)

指定接受slots 主机id

Receiving node id (除了被移除的master主机之外的主机ip)

指定移除slots 主机id

Node1(被移除的id)

Node2(输入done)

最后操作 删除master主机IP (del-node)

2.3 把移除的主机再次添加到master/slave中

将被移除的主机进入到命令行 cluster reset进行重置清空,再进入到

管理主机添加

查看集群信息 info

2.4 让一台主服务器有多台从服务器

Redis-trib.rb add-node 所做的角色 --master id 主服务器的id

所要添加作为从服务器的ip:端口 主服务器的IP:端口 #在集群管理主机操作

2.5还原为独立的机器

停止服务 /etc/init.d/redis\_6379 stop

删除数据库目录下所有rm -rf /var/lib/redis/6379/\*

注释配置文件里的三行vim /etc/redis/6379.conf

重启服务/etc/init.d/redis\_6379 start

查看端口netstat -utnlp | grep redis-server

进入命令行 keys \* / cluster info

####################**nosql day03**########################

**一.主从复制**

1.一主多从 (查看复制信息以及主机角色:info replication)

命令行配置:salveof 被指定的主机ip 端口号

永久配置:/etc/redis/6379.conf 修改282行:

反客为主:1.临时修改:slaveof no one

2.永久修改:注释配置文件282行

2.一主多从

同上

3.配置带验证的主从复制

修改配置文件815行:去掉注释密码(此时数据不同步)

修改289行加入主服务器的密码 (一定要重启)

在命令行下配置文件config set masterauth 123456

Config rewrite (永久生效)

1. 哨兵服务

安装源码的编译软件gcc

创建并编写配置文件

启动哨兵服务

1. **持久化**
2. RDB(默认) dump.rdb (220行)

写入数据再save ,将用来备份的主机停止服务,将数据库目录下的dump.rdb传给要用来恢复的主机的数据库目录下.清除数据库目录下的所有,再重启.

2.AOF:追加方式记录写操作的文件,追加在文件的末尾.

开启AOF功能:配置文件的673行 改成yes

Appendonly.aof 配置文件

1. AOF文件记录写操作的方式

Appendfsync always 时时记录,并完成磁盘同步

Appendfsync everysec 每秒记录一次,并完成磁盘同步

Appendfsync no 写入aof 不执行磁盘同步

1. 修复AOF文件

Redis-check0aof --fix appendonly.aof

1. **数据类型**

1.**字符类型**(string)

Set key 值 #设置key及值

Setrange key 排序位数 数值 #从某位数之后开始复写值

Strlen key #统计字串长度

Append key 值 #存在则追加数据

Setbit key offset 值 #设置或清楚特定偏移量上的位

Bitcount key #统计字串中被设置为1的比特位数

Decr key #将key中的值减1

Decrby key 值 #将key中的值,减去值

Getrange key 头位数 尾位数 #截取key中的这个范围

Incr key #将key的值加1

Incrby key 值 #将key的值加值

Incrbyfloat key 值 #为key中存储的值加上小数数值

Mget key #获取多个key的值,空格分割

Mset key 值 #设置多个key以及数值

2.**列表类型**(字符列队 先进后出)

Lpush key 值 #将多个数据插入列表key

Lrange key 起始位 终止位 #读取key值

Lpop key #移除并返回列表头的数据

Llen key #查看列表key的长度

Lindex key 某位数(从0开始)#显示列表中的某位数的值

Lset key 某位数 值 #将这个位置的值修改为其他值

Rpush key 值 #将值插入到key列表的末尾

Rpop key #删除并输出末尾的值

3.**hash表(一个key对应多个field,一个filed对应一个值)**

hset key 列表名 值 #给hash表中的列赋值

Hget key 列表名 #获取hash表中的列值

Hmget key 列表名 值 #同时给多个列表赋值

Hmget key 列表名 值 #获取多个列表值

Hkey key #hash表中的所有列名

Hgetall key #显示所有的列表和值

Hvals key #列出有列表中的值

Hdel key 列表名 #删除

####################**安全DAY01#**#############################

**●监控概述**

1. 概述

　　　监控的目的：

　　　监控的资源类别：公开数据包括web ftp ssh等应用服务 tcp端口等

私有数据包括cpu内存磁盘等使用和进程等运行信息

2.监控软件

　　　系统监控命令：traceroute 　到达目标主机所经过的路由ip

iostat

Ping

Netstat或ss

ifcofig

df -h

Swapon -s

Free

Uptime

Ps

　　　自动化监控系统：Cacti　　基于SNMP协议的监控

Nagios　　基于Agent监控，检查与报警机制

Zabbix　　基于多种监控机制

**一．Zabbix基础**

1. **基本概念**

　　　Zabbix简介：高度集成的监控解决方案，通过C/S模式采集监控数据

　　　　　　　　　通过B/S模式实现Web管理．

　　　监控拓扑图：在LNMP环境下运行（监控服务器）

　　　　　　　　　需安装Agent,支持SNMP（被监控主机）

**2.部署LNMP**

1. **安装源码zxbbix软件**

　　　　安装的软件net-snmp-devel curl-devel libevent-devel

　　操作./configure --enable-server --enable-proxy --enable-agent

　--with-mysql=/usr/bin/mysql\_config --with-net-snmp --with-libcurl

1. 创建存储数据的库表及连接用户

1. 做初始化准备

　　　修改配置文件　／usr/local/nginx/conf/nginx.conf

安装软件　php-gd php-xml php-ldap php-bcmath php-mbstring

修改php配置文件　/etc/php.ini

878行672行384行394行

1. 初始化
   1. <http://192.168.2.5/index.php>

欢迎页面

检测页面

连接数据库配置

配置监控服务器（端口号默认10051）

配置总结

保存配置信息的文件：/usr/local/nginx/html/conf/zabbix.conf.php

1. 登录管理页面
   1. 账号:admin 密码：zabbix
   2. 修改登录密码：
   3. 修改页面语言为中文
2. 启动zabbix监控服务
   1. 修改配置文件　host name user password log\_file
   2. 创建zabbix用户　启动服务：zabbix\_server
   3. 查看端口，默认监控的端口10051

启动前保证启动检查端口3306　80　9000

1. **Zabbix监控服务**
   1. 监控远端主机

配置客户端（被监控主机）：

安装软件gcc pcre-devel仅加载模块agent

Ls /usr/local/etc 存放配置文件

Ls /usr/local/bin 存放命令

Ls /usr/local/sbin

修改配置文件:zabbix\_agentd.conf

90行　134行　30行

启动服务：启动前创建zabbix用户

Zabbix\_agentd　＃启动命令

服务端口为:10050

配置监控服务器2.5:

管理员登录页面

添加主机:1配置-2主机-右上角创建主机---

选择模板:显示的主机名-模板-点击链接指示器

右选择-更新

查看数据:刷新-检测中-最新数据-各种选择-应用

* 1. 自定义监控项(在客户端定义监控命令,给监控服务器使用)

配置客户端(192.168.2.100)

允许自定义监控命令

配置文件去注释264行 280行=1

定义监控命令

/local/etc/zabbix\_agentd.conf.d/任意

写入内容UserParameter=任意,

wc -l /etc/passwd | awk '{print $1}'

重启服务:先杀再启

测试定义的监控命令

zabbix\_get -s 127.0.0.1 -k 任意

配置服务器(192.168.2.5)(管理员登录做配置)

创建新模板: ATMP 配置-模板-创建模板

创建新应用级:mon\_user 模板名-右上创建应用集-ok

创建新监控项 mon\_web100\_user 对应命令为所谓任意

监控项-右上创建监控项-编辑内容即可

` 监控主机使用新创建的模板:配置-主机-模板名-模板-

添加-更新

查看监控数据:检测中-最新数据

取消模板链接-删除模板-

#######################安全DAY02############################

◆**Zabbix报警机制**

基本概念:自定义的监控器默认不会自动报警,也不会提示错误

触发器(tigger):会执行某一动作

动作(action):被触发后的行为,可以使发送邮件或者重启服务

**■监控报警机制()**

1.创建触发器:配置-模块-触发器-创建-编辑到表达式-添加

2.创建动作的前提要有邮件服务器(端口25在监控服务器本身配置)

安装postfix

绑定用户 echo "127.0.0.1 zabbixserver" >> /etc/hosts

安装mailx

mail -s "xxx" zabbix < /etc/hosts

切换到zabbix用户 直接输入mail

创建动作:配置-动作-创建动作-默认信息用默认的选项

创建用户:管理-用户-报警媒介-只修改收件人

测试配置:在监控的100主机上添加新用户,触发器被出发后执行动作,

查看用户是否收到邮件’

1. **zabbix进阶操作**

●自动发现的配置过程:创建自动发现规则

创建action动作

通过动作,执行添加主机,链接模板到主机等操作

1.自动发现

创建自动发现规则:配置-自动发现-创建发现规则

创建动作:事件源为自动发现-创建动作-操作

1. 主被动监控(修改为主动监控)

配置客户端:安装软件

修改配置文件 注释93行 134 118 183 145

启动服务 查看京城

配置监控服务器:添加监控主机

克隆模板

1. 拓扑图与聚合图形
2. 监控案例

1.监控nginx

运行Nginx服务器 并支持查看状态信息

编写脚本,并给脚本加X权限

把脚本定义为命令

重启动zabbxi\_agentd服务

测试命令

在监控页面作如下配置

创建新的监控模板 STMP2

创建新的应用集 nginx\_status

创建新的监控项 名称和对应的命令

Now\_link\_num get\_ngixn\_status

His\_sum\_num

调用新创建的模板监控主机100

查看监控数据

2.监控网络链接状态