**10\_RDMBS205PXC+mysql存储引擎**

**一 PXC**

**MHA必须做的配置**

配置mysql主从同步

指定vip地址

恢复服务器时必须手动同步数据和手动配置主从结构,从新修改配置文件

1.1 **PXC概述** Percona XtraDB Cluster

1.1.1是基于Galera的mysql高可用集群解决方案

1.1.2 Galera cluster是Codershiop公司开发的一套免费开源的高可用方案

1.1.3 PXC集群主要由两部分组成: Percona Server with XtraDB 和Write Set

Replication patches(同步\多主复制插件)

1.1.4 官网:http://galeracluster.com

**1.2 PXC 特点**

1.2.1 数据强一致性,无同步延迟

1.2.2 没有主从切换操作,无需使用虚拟VIP #不用配置数据库主从和设置vip

1.2.3 仅支持InnoDB存储引擎,默认为InnoDB

1.2.4 多线程复制

进程: 系统进行资源分配和调度的基本单位;是多个线程的组合,是线程的容器;单独处理一个任务,独享资源

线程: 操作系统能够进行运算调度的最小单位,进程中一个单一顺序的控制流，一个进程中可以并发多个线程;一起处理一个任务,相互独立,可同时工作,共享资源

1.2.5 部署使用简单

1.2.6 支持节点自动加入,无需手动拷贝数据

**1.3 相应端口**

端口 说明

3306 数据服务端口

4444 SST 端口 全量同步时才可见此端口

4567 集群通信端口

4568 IST 端口 增量同步时才可见此端口

SST state snapshot transfer 全量同步

IST increment state transfer 增量同步

**1.4 服务器角色**: 71 72 73 全为数据库服务器

编辑每台服务器的/etc/hosts文件,进行服务器名映射

pxenode71\72\73 ~]# vim /etc/hosts

192.168.4.71 pxcnode71

192.168.4.72 pxcnode72

192.168.4.73 pxcnode73



映射验证:相互ping ping -c 4 -i 0.1 pxcnode71\72\73

**二 部署PXC**

**2.1 安装软件**

**2.1.1 软件介绍**

percona-xtrabackup-24-2.4.13-1.el7.x86\_64.rpm 在线热备程序

qpress-1.1-14.11.x86\_64.rpm 递归压缩程序

percona-xtradb-cluster-server-57-5.7.25-31.35.1.el7.x86\_64.rpm

集群服务程序

**2.1.2 71 72 73安装依赖包\主软件包**

room9pc01 ~]$ scp -r /linux-soft/03/PXC

[root@192.168.4.71\72\73:/root/](mailto:root@192.168.4.71/72/73:/root/)

pxenode71 ~]# cd PXC

pxenode71 PXC]# rpm -ivh libev-4.15-1.el6.rf.x86\_64.rpm

pxenode71 PXC]# yum -y install

percona-xtrabackup-24-2.4.13-1.el7.x86\_64.rpm

pxenode71 PXC]# rpm -ivh qpress-1.1-14.11.x86\_64.rpm

pxenode71 PXC]# tar -xvf

Percona-XtraDB-Cluster-5.7.25-31.35-r463-el7-x86\_64-bundle.tar

pxenode71 PXC]# yum -y install Percona-XtraDB-Cluster-\*.rpm

**2.2 配置服务(71 72 73都修改)**

pxenode71 ~]# cd **/etc/percona-xtradb-cluster.conf.d/**

**2.2.1** 修改**数据库服务运行参数配置文件mysqld.cnf**

pxenode71 percona-xtradb-cluster.conf.d]#

71 percona-xtradb-cluster.conf.d]# cp mysqld.cnf{,.bak} #备份

71 percona-xtradb-cluster.conf.d]# vim mysqld.cnf

[mysqld]

server-id=**71\72\73 #修改server-id号**

**2.2.2** 修改**mysql服务运行进程配置文件mysqld\_safe.cnf**

该文件仅查看

**2.2.3** 修改**主配置文件wsrap.cnf(PXC集群配置文件)**

pxenode71 percona-xtradb-cluster.conf.d]# vim **wsrep.cnf**

**2.2.3.1 定义集群成员列表**

8 wsrep\_cluster\_address=gcomm://

**192.168.4.72,192.168.4.73,192.168.4.71** #本机IP写最后

**2.2.3.2 定义本机IP地址**

25 wsrep\_node\_address=**192.168.4.71** #解除注释,写本机IP

**2.2.3.3 定义集群名称.3台服务器必须设置相同**

27 wsrep\_cluster\_name=**pxc-cluster** #默认即可

**2.2.3.4 定义本机服务器名**

30 wsrep\_node\_name=**pxcnode71**

**2.2.3.5 定义SST数据同步授权用户及密码**

39 wsrep\_sst\_auth="sstuser:**123qqq...A**" #解除注释,修改密码

**2.3 启动集群服务**

在1台服务器上执行即可(192.168.4.71)

**2.3.1 71启动集群服务**

pxenode71 ~]# systemctl start mysql.service

**2.3.2 查看数据库管理员初始登录密码**

pxenode71 ~]# grep pass /var/log/mysqld.log

**2.3.3 使用初始密码登录**

pxenode71 ~]# mysql -uroot -p"初始密码"

**2.3.4 修改登录密码(无密码策略要求)**

mysql> alter user user() identified by "123456"

**2.3.5 添加授权用户**

mysql> grant reload,lock tables,replication client,process on \*.\* to sstuser@"localhost" identified by "123qqq...A";

**2.3.6 72\73启动数据库服务mysql(第一次启动会做全量同步)[无d]**

pxcnode72\73 ~]# systemctl restart mysql #无d

**2.3.7 72\73登录数据库查看用户授权**

pxcnode72\73 ~]# mysql -uroot -p123456

72\73 mysql> select user,host from mysql.user;

有数据: sstuser | localhost

72\73 mysql> show grants for sstuser@"localhost";

有在71授予的权限

**2.3.8 在71\72\73查看端口**

pxenode71 ~]# ss -antulp | grep :3306

tcp LISTEN 0 80 :::3306 :::\* users:(("mysqld",pid=1862,fd=32))

pxenode71 ~]# ss -antulp | grep :4567

tcp LISTEN 0 128 \*:4567 \*:\* users:(("mysqld",pid=1862,fd=11))

**2.4 测试配置**

**2.4.1 任意一台服务器上查看集群信息**

71\72\73任意mysql> **show status like "%wsrep%";**

wsrep\_incoming\_addresses

192.168.4.73:3306,192.168.4.71:3306,192.168.4.72:3306

#成员列表

wsrep\_cluster\_size 3 #集群服务器台数 wsrep\_cluster\_status Primary #本服务器在集群中的状态 wsrep\_connected ON #相互之间的连接状态

wsrep\_ready ON #本服务器的服务状态

**2.4.2 测试集群功能**

**在任意一台服务器上的数据库内添加访问数据的授权访问用户**

73 mysql> grant all on gamedb.\* to admin@"%" identified by "123456";

71\72 mysql> show grants for admin@"%";

**在客户端使用授权访问用户连接任意数据库服务器都可以写入数据,查看到同样数据**

**50连接71的数据库**

mysql50 ~]# mysql -h192.168.4.71 -uadmin -p123456

50 mysql> show databases; #无gamedb库,创建

50 mysql> create database gamedb;

72\73 mysql> show databases;

**建表时,必须要有主键字段**

71 mysql> create table gamedb.stuinfo(

-> id int **primary key auto\_increment**, #主键并自增长

-> name char(15),

-> age tinyint unsigned,

-> class char(9));

**72\73查看库.表 gamedb.stuinfo**

72\73 mysql> show databases;

mysql> use gamedb;

mysql> show tables;

**50连接71的数据库写入数据**

50 mysql> insert into gamedb.stuinfo(name,age,class)

values("tom",21,"nsd1906");

50 mysql> insert into gamedb.stuinfo(name,age,class)

values("jerry",22,"nsd1906");

50 mysql> insert into gamedb.stuinfo(name,age,class)

values("harry",23,"nsd1906");

**71\72\73查看数据**

71\72\73 mysql> select \* from gamedb.stuinfo;

**数据的id按服务器数量自加间隔增长,避免id值重复**

**2.4.3 测试故障自动恢复**

任何一台数据库服务器宕机都不影响用户存取数据

**2.4.3.1 停止72的mysql服务并查看72的端口**

pxcnode72 ~]# systemctl stop mysql

pxcnode72 ~]# ss -antulp | grep :3306

pxcnode72 ~]# ss -antulp | grep :4567

**2.4.3.2 71\73上数据库内查看%wsrep%状态**

71 mysql> show status like "%wsrep%";

wsrep\_incoming\_addresses

192.168.4.73:3306,192.168.4.71:3306 #成员列表没有72了

wsrep\_cluster\_size 2 #集群服务器台数变成2 wsrep\_cluster\_status Primary #本服务器在集群中的状态 wsrep\_connected ON #相互之间的连接状态

wsrep\_ready ON #本服务器的服务状态

**2.4.3.3 50连接71数据库并写入game.db数据,在71\73上看写入的数据**

mysql50 ~]# mysql -h192.168.4.71 -uadmin -p123456

50 mysql> insert into gamedb.stuinfo(name,age,class)

values("xyy",24,"nsd1906");

50 mysql> insert into gamedb.stuinfo(name,age,class)

values("xyy1",25,"nsd1906");

50 mysql> insert into gamedb.stuinfo(name,age,class)

values("xyy2",26,"nsd1906");

71\73上数据库内查看新写入的数据[此时id按2间隔增长]

71\73 mysql> select \* from gamedb.stuinfo;

**2.4.3.4 宕机服务器继续运行后,自动同步宕机期间的数据**

**启动72的musql服务并查看72的端口**

pxcnode72 ~]# systemctl restart mysql

pxcnode72 ~]# ss -antulp | grep :3306

pxcnode72 ~]# ss -antulp | grep :4567

**72查看数据库内的数据是否自动同步**

72 mysql> select \* from gamedb.stuinfo;

71\72\73查看%wsrep%状态

71\72\73 mysql> show status like "%wsrep%";

wsrep\_incoming\_addresses

192.168.4.73:3306,192.168.4.72,192.168.4.71:3306

#成员列表恢复

wsrep\_cluster\_size 3 #集群服务器台数恢复为3 wsrep\_cluster\_status Primary #本服务器在集群中的状态 wsrep\_connected ON #相互之间的连接状态

wsrep\_ready ON #本服务器的服务状态

**2.4.3.5 50连接72数据库写入数据,并在71\72\73上查看写入的数据**

mysql50 ~]# mysql -h192.168.4.72 -uadmin -p123456

50 mysql> insert into gamedb.stuinfo(name,age,class)

values("natasha",27,"nsd1096");

71\72\73 mysql> select \* from gamedb.stuinfo; #正常显示新写入的值

**三 mysql存储引擎**

**3.1 存储引擎概述**

存储引擎,作为可插拔式的组件提供

mysql服务软件自带的功能程序,处理表的处理器

不同的存储引擎有不同的功能和数据存储方式

mysql 5.0\5.1 MyISAM

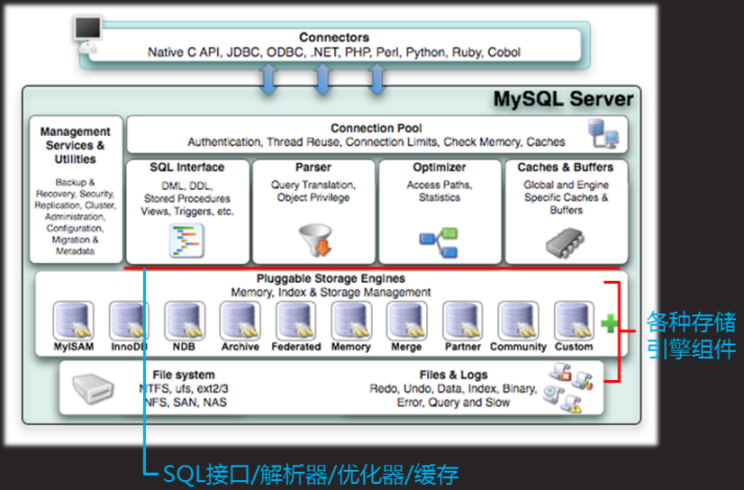
mysql 5.5\5.6 InnoDB

列出可用的存储引擎类型: show engines;

查看软件版本: show variables like "%version%";

查看表的当前引擎及详细属性: show create table stuinfo;

**3.2 mysql体系结构**



**管理工具,连接池,接口,解析器,优化器,缓存,存储引擎,文件系统**

**四 配置存储引擎**

**4.1 修改表存储引擎**

4.1.1 建表时手动指定[未指定则使用默认存储引擎]

**create table 表名 engine=存储引擎名;**

4.1.2 建表后手动修改表的存储引擎

**alter table 表名 engine=存储引擎名;**

**4.2 设置数据库服务使用的存储引擎为MyISAM[修改/etc/my.cnf配置文件]**

vim /etc/my.cnf

default-storage-engine=存储引擎名 #[mysqld]下添加此行

修改完成后重启mysqld服务

shutting down mysql ... [确定]

starting mysql [确定]

mysql50 ~]# mysql -uroot -p123456 -e "show engines;"

#列出可用引擎列表,修改后表中的默认引擎为MyISAM

**4.3 存储引擎特点**

**4.3.1 MyISAM存储引擎**

主要特点

支持表级锁定;不支持事务\事务回滚\外键

表文件

表名.frm #表结构文件[存储 desc 表名 出现的内容]

表名.MYI #表索引文件

表名.MYD #表数据文件

**4.3.2 InnoDB存储引擎**

主要特点

支持行级锁定;支持事务\事务回滚\外键

表文件

表名.frm 表结构文件 #.frm文件是所有存储引擎都有的文件

表名.ibd 表索引+数据文件

事务日志文件

ibdata1

ib\_logfile0

ib\_logfile1

**4.3.3 mysql锁机制**

4.3.3.1 锁粒度

表级锁:对整张表加锁

行级锁:仅对被访问的行分别加锁

4.3.3.2 锁类型

读锁(共享锁):支持并发读,可同时多个客户端读

写锁(互斥锁\排他锁):是独占锁,上锁期间其他线程不能读表或写表

4.3.3.3 查看当前锁状态

show status like “table\_lock%”;

**4.3.4 InnoDB的事务特性(ACID)**

**事务: 从连接到操作到断开连接的过程**

**事务回滚:当事务过程出现错误时,表内容恢复到正确的状态**

4.3.4.1 **Atomic[əˈtɒmɪk]:原子性**

事务的整个操作是一个整体,不可分隔,要求全部成立,要么全部失败

4.3.4.2 **Consistency[kənˈsɪstənsi]:一致性**

事务操作的前后,所有人看到的表记录都一样

4.3.4.3 **Isolation[ˌaɪsəˈleɪʃn]:隔离性**

事务操作是相互隔离不受影响的,多个用户对同一表操作时,相互不知道对方的存

在及对方在对表进行事务操作,只有事务操作提交前后才知道

4.3.4.4 **Durability[ˌdjʊərəˈbɪlɪti]:持久性**

数据一旦提高,不可改变,永久改变表数据

**4.3.4.5 相关命令**

**mysql> show variables like “autocommit”; #查看提交状态**

**mysql> set autocommit=off; #关闭自动提交**

**mysql> rollback; #数据回滚**

**mysql> commit; #提交数据**

**命令行关闭自动提交只对当前终端有效,关闭后自动还原**

**4.4 事务特性测试**

4.4.1 开1个终端登录50的数据库

mysql50 ~]# mysql -uroot -p123456

4.4.2 查看自动提交状态

50 mysql> show variables like "autocommit";

Variable\_name Value

autocommit ON

4.4.3 关闭自动提交

50 mysql> set autocommit=off;

4.4.4 创建一个表,并写入数据,不提交

50 mysql> create table db10.e(id int) engine=innodb;

50 mysql> insert into db10.e values(11);

4.4.5 新开一个终端登录50数据库,查看db10.e中的数据

50 mysql> select \* from db10.e; #无数据

4.4.6 原终端提交

50 mysql> commit;

4.4.7 新终端查看db10.e中的数据

50 mysql> select \* from db10.e; #有数据11

4.4.8 原终端删除表e记录不提交

mysql> delete from table db10.e;

4.4.9 新终端再查看db10.e中的数据

50 mysql> select \* from db10.e; #有数据11

4.4.10 原终端回滚+提交+查看

50 mysql> rollback;

50 mysql> commit;

50 mysql> select \* from db10.e; #有数据11

4.4.11 新终端再查看db10.e中的数据

50 mysql> select \* from db10.e; #有数据11

4.4.12 原终端删除db10.e记录并提交并查看

50 mysql> delete from db10.e;

50 mysql> commit;

mysql> select \* from db10.e; #无数据

4.4.13 新终端再查看db10.e中的数据

mysql> select \* from db10.e; #无数据