集群安装

# MySQL集群简介

## 什么是MySQL集群

MySQL集群是一个无共享的(shared-nothing)、分布式节点架构的存储方案，其目的是提供容错性和高性能。

数据更新使用读已提交隔离级别（read-committedisolation)来保证所有节点数据的一致性，使用两阶段提交机制（two-phasedcommit)保证所有节点都有相同的数据(如果任何一个写操作失败，则更新失败）。

无共享的对等节点使得某台服务器上的更新操作在其他服务器上立即可见。传播更新使用一种复杂的通信机制，这一机制专用来提供跨网络的高吞吐量。

通过多个MySQL服务器分配负载，从而最大程序地达到高性能，通过在不同位置存储数据保证高可用性和冗余。

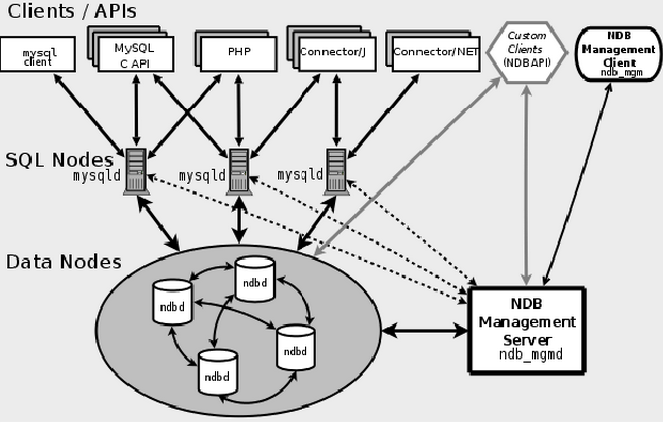
## 集群 MySQL中名称概念

1) Sql 结点（SQL node—下图对应为 mysqld）：分布式数据库。包括自身数据和查询中心结点数据

2) 数据结点（Data node – ndbd）：集群共享数据（内存中）

3) 管理服务器（Management Server – ndb\_mgmd）：集群管理 SQL node,Data node

详情见下图：



# 环境配置

hosts：

|  |
| --- |
| 192.168.8.11 master  192.168.8.12 slave1  192.168.8.13 slave2 |

# 安装前要

不管是Management Server，还是Data node、SQL node，都需要先安装MySQL集群版本，然后根据不同的配置来决定当前服务器有哪几个角色。

# 创建mysql用户和组（n台）

安装之前准备好mysql用户和mysql用户组，相关命令：

# groupadd mysql

# useradd mysql -g mysql

推荐使用这种方式：

# useradd mysql 默认创建同名组

# 关闭防火墙（n台）

chkconfig iptables off

service iptables stop

setenforce 0

# 安装MySQL集群版本

## 主节点

### 上传

su - root

cd /usr/local

rz –y

上传安装包

mysql-cluster-gpl-7.4.6-linux-glibc2.5-x86\_64.tar.gz

### 解压

tar –zxvf mysql-cluster-gpl-7.4.6-linux-glibc2.5-x86\_64.tar.gz

### 重命名

mv mysql-cluster-gpl-7.4.6-linux-glibc2.5-x86\_64 mysql

### 授权

chown –R mysql:mysql mysql

### 下发

su - root

scp -r /usr/local/mysql root@slave1:/usr/local

scp -r /usr/local/mysql root@slave2:/usr/local

### 安装

/usr/local/mysql/scripts/mysql\_install\_db --user=mysql --basedir=/usr/local/mysql --datadir=/usr/local/mysql/data

### 配置Management Server

su - root

mkdir /var/lib/mysql-cluster

vi /var/lib/mysql-cluster/config.ini

|  |
| --- |
| [ndbd default]  NoOfReplicas=2  定义在Cluster环境中相同数据的分数，通俗一点来说就是每一份数据存放NoOfReplicas份。如果希望能够冗余，那么至少设置为2（一般情况来说此参数值设置为2就够了），最大只能设置为4。另外，NoOfReplicas值得大小，实际上也就是node group大小的定义。NoOfReplicas参数没有系统默认值，所以必须设定，而且只能设置在[NDBD DEFAULT]中，因为此数值在整个Cluster集群中一个node group中所有的NDBD节点都需要一样。另外NoOfReplicas的数目对整个Cluster环境中NDB节点数量有较大的影响，因为NDB节点总数量是NoOfReplicas \* 2 \* node\_group\_num；  DataMemory=80M  设定用于存放数据和主键索引的内存段的大小。这个大小限制了能存放的数据的大小，因为ndb存储引擎需属于内存数据库引擎，需要将所有的数据（包括索引）都load到内存中。这个参数并不是一定需要设定的，但是默认值非常小（80M），也就是说如果使用默认值，将只能存放很小的数据。参数设置需要带上单位，如512M，2G等。另外，DataMemory里面还会存放UNDO相关的信息，所以，事务的大小和事务并发量也决定了DataMemory的使用量，建议尽量使用小事务；  IndexMemory=18M  设定用于存放索引（非主键）数据的内存段大小。和DataMemory类似，这个参数值的大小同样也会限制该节点能存放的数据的大小，因为索引的大小是随着数据量增长而增长的。参数设置也如DataMemory一样需要单位。IndexMemory默认大小为18M；  实际上，一个NDB节点能存放的数据量是会受到DataMemory和IndexMemory两个参数设置的约束，两者任何一个达到限制数量后，都无法再增加能存储的数据量。如果继续存入数据系统会报错“table is full”。  [ndb\_mgmd]  Id=1集群中结点的唯一标识，取值 1~63。  Hostname=192.168.8.11 主节点  datadir=/usr/local/mysql/logs  [ndbd]  Id=2  Hostname=192.168.8.12 从节点  datadir=/usr/local/mysql/data/  [ndbd]  Id=3  Hostname=192.168.8.13 从节点  datadir=/usr/local/mysql/data/  [MYSQLD]  [MYSQLD] |
| 完整配置，无注解：  [ndbd default]  NoOfReplicas=2  DataMemory=80M  IndexMemory=18M  [ndb\_mgmd]  Id=1  Hostname=192.168.8.11  datadir=/usr/local/mysql/logs  [ndbd]  Id=2  Hostname=192.168.8.12  datadir=/usr/local/mysql/data/  [ndbd]  Id=3  Hostname=192.168.8.13  datadir=/usr/local/mysql/data/  [MYSQLD]  [MYSQLD] |

## datanode(n台)

### 授权

su - root

chown -R mysql:mysql /usr/local/mysql

### 安装

/usr/local/mysql/scripts/mysql\_install\_db --user=mysql --basedir=/usr/local/mysql --datadir=/usr/local/mysql/data

### 配置Data Node

su – root

cp /usr/local/mysql/support-files/my-default.cnf /etc/my.cnf

cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld

vi /etc/my.cnf

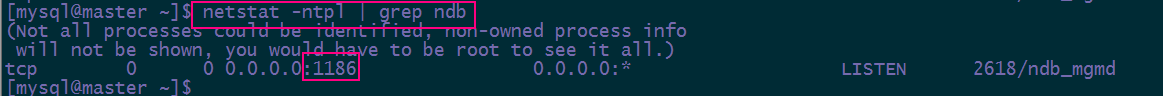
|  |
| --- |
| 追加以下内容：  datadir=/usr/local/mysql/data/  ndbcluster  #SQL节点是使用命令“mysqld –ndbcluster”启动的，或将“ndbcluster”添加到“my.cnf”后使用“mysqld”启动。  ndb-connectstring=192.168.8.11  [ndbd]  connect-string=192.168.8.11  [mysql\_cluster]  ndb-connectstring=192.168.8.11  [ndb\_mgm]  connect-string=192.168.8.11  [ndb\_mgmd]  config-file=/var/lib/mysql-cluster/config.ini |

### 在管理节点启动相关服务:

注意：如果提示错误，让创建文件夹，创建个 # mkdir /usr/local/mysql/logs。

  #/usr/local/mysql/bin/ndb\_mgmd -f /var/lib/mysql-cluster/config.ini

  #netstat –ntpl | grep ndb

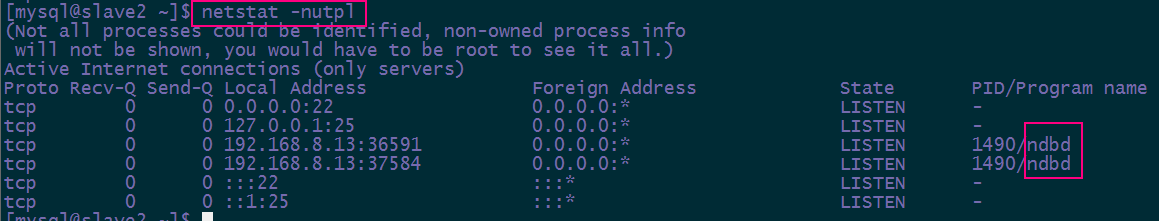


  看到1186端口开放了说明启动是正常的.

### 在数据节点启动相关服务:

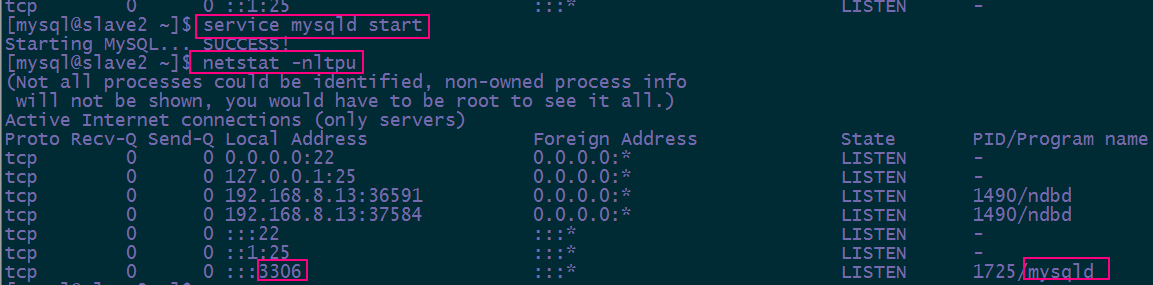
  #/usr/local/mysql/bin/ndbd --initial

  #netstat –lntpu



# service mysqld start

#netstat –lntpu



可以看到相关的ndbd服务以及mysql已经启动ok了.

# 功能测试

  到管理节点查看下相关服务状态

 # /usr/local/mysql/bin/ndb\_mgm

ndb\_mgm> show

Connected to Management Server at: localhost:1186

Cluster Configuration

---------------------

[ndbd(NDB)] 2 node(s)

id=2 @192.168.8.12 (mysql-5.6.24 ndb-7.4.6, Nodegroup: 0, \*)

id=3 @192.168.8.13 (mysql-5.6.24 ndb-7.4.6, Nodegroup: 0)

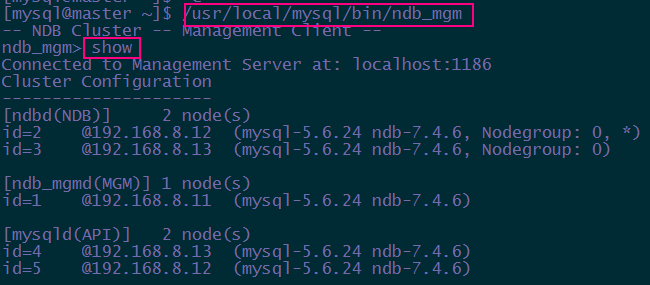
[ndb\_mgmd(MGM)] 1 node(s)

id=1 @192.168.8.11 (mysql-5.6.24 ndb-7.4.6)

[mysqld(API)] 2 node(s)

id=4 @192.168.8.13 (mysql-5.6.24 ndb-7.4.6)

id=5 @192.168.8.12 (mysql-5.6.24 ndb-7.4.6)

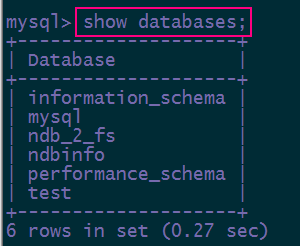


可以看到这里的数据节点、管理节点、sql节点都是正常的.

现在我们在其中一个数据节点上进行相关数据库的创建,然后到另外一个数据节点上看看数据是否同步

# /usr/local/mysql/bin/mysql -u root –p

登录时，没有设置密码，直接回车即可。

  
mysql> create database aa;  
mysql> use aa  
mysql> CREATE TABLE ctest2 (i INT) ENGINE=NDB; //这里必须指定数据库表的引擎为NDB,否则同步失败，只有这样子，这两台机器才能共享数据库的数据结构  
mysql> INSERT INTO ctest2 () VALUES (1);  
mysql> SELECT \* FROM ctest2;  
+------+  
| i    |  
+------+  
|    1 |   
+------+

现在到另外一个数据节点查看下aa数据库是否同步过来了.

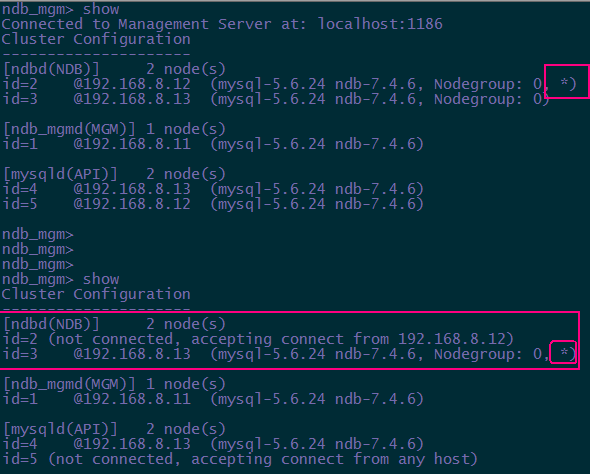
#/usr/local/mysql/bin/mysql -u root -p

mysql> show databases;  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |   
| aa                 |   
| bb                 |   
| mysql              |   
| ndb\_3\_fs           |   
| test               |   
+--------------------+  
mysql> use aa  
mysql> select \* from ctest2;  
+------+  
| i    |  
+------+  
|    1 |   
+------+

从上面可以看到数据已经同步了,mysql集群环境已经搭建完成.

# 破坏性测试

  大家在上面可以看到192.168.8.12作为主的数据节点,现在把192.168.8.12这台机器关闭,看下有什么结果



从上面可以发现现在192.168.8.12这台机器的数据节点和sql节点已经连接不上了, 192.168.8.13成为了主数据节点,我们现在在192.168.8.13数据节点上创建一个表,然后恢复192.168.8.12的数据节点,看下它是否把数据同步过来了.

先在192.168.8.13数据节点做如下操作:

  mysql> create table ctest3(id int(11)) engine=NDB;  
  mysql> show tables;  
+--------------+  
| Tables\_in\_aa |  
+--------------+  
| ctest2       |   
| ctest3       |   
+--------------+  
mysql> insert into ctest3 values(1);  
mysql> select \* from ctest3;  
+------+  
| id   |  
+------+  
|    1 |   
+------+

然后我们恢复192.168.8.12数据节点,查看下ctest3数据是否同步过来了.

mysql> show databases;  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |   
| aa                 |   
| bb                 |   
| mysql              |   
| ndb\_2\_fs           |   
| test               |   
+--------------------+  
mysql> use aa  
mysql> show tables;   
+--------------+  
| Tables\_in\_aa |  
+--------------+  
| ctest        |   
| ctest2       |   
| ctest3       |   
+--------------+  
mysql> select \* from ctest3;  
+------+  
| id   |  
+------+  
|    1 |   
+------+

可以看到192.168.8.12数据节点已经把192.168.8.13数据节点的数据同步过来了,说明mysql集群是没有问题的。