因为负载均衡的前提就是，各个服务器的数据库是数据同步的。在业务量不大的时候，我们会使用主从复制的方法实现服务器数据同步，一主多从或者是双主等，但是虽然进行了读写分离，但是对于读的方法限制还是比较大，所以解决数据同步的问题就是数据库集群的意义。我这里使用mysql官网提供的mysql-cluster实现集群。

mysql cluster中的几个概念解释

为了简单，我后面简称mysql-cluster为mc。

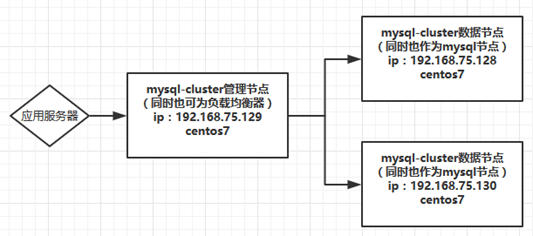
1、mc已经包含了mysql，我下载的最新的mc7.5，官方说明包含的是mysql版本是5.7。所以不需要使用别的msyql的安装包安装数据库。同时注意mysql5.7的版本在安装的命令和配置上面和之前的版本有很大的不同，所以网上有很多mc7.5之前的版本，所包含的mysql版本不同，所以安装方法不同。

2、管理节点，mc管理节点负责管理、配置、监控整个集群。

3、数据节点，使用内存存放数据，保存进数据节点的数据都会自动复制并存储到其他数据节点。

4、mysql节点，也叫数据库节点，和我们平时使用的mysql相同，作为数据库使用。被数据节点访问。

架构图及说明



我实验中的配置就是如图所示，因为虚拟机占用内存较大，只使用了3台服务器，在实际情况中最好将数据节点和mysql节点分开。在实际中负载均衡服务还需要做备份，因为万一负载均衡服务器宕机将会导致所有数据节点都无法访问，所以需要对负载均衡服务器备份，有条件的话，分开管理节点和负载均衡器。实验只实现整个数据库集群，负载均衡请参考之前的博客配置即可。

1. 前期准备：保证服务器间可以互相ping通，有固定的ip地址。保证其使用的3306等端口没有其他程序占用。
2. 群集管理节点的默认端口是1186; 数据节点的默认端口是2202.但是，群集可以自动为已经空闲的数据节点分配数据节点的端口。
3. 下载mysql cluster

首先使用该镜像<http://mirrors.sohu.com/mysql/MySQL-Cluster-7.5/> 下载mysql cluster。

我下载的版本是mysql-cluster-gpl-7.5.5-linux-glibc2.5-x86\_64.tar.gz

Wget <http://mirrors.sohu.com/mysql/MySQL-Cluster-7.5/> mysql-cluster-gpl-7.5.5-linux-glibc2.5-x86\_64.tar.gz

1. 解压

# cd /usr/local

# tar xvf mysql-cluster-gpl-7.5.5-linux-glibc2.5-x86\_64.tar.gz

1. 将需要的文件取出

# cd mysql-cluster-gpl-7.5.5-linux-glibc2.5-x86\_64

# cp bin/ndb\_mgm\* /usr/local/bin

# cd /usr/local/bin

# chmod +x ndb\_mgm\*

1. 新建配置文件并且初始化管理节点

# mkdir /var/lib/mysql-cluster

# mkdir /usr/local/mysql

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

这里修改为：

[ndbd default]

NoOfReplicas=2

DataMemory=512M

IndexMemory=18M

[ndb\_mgmd]

HostName=10.1.138.116

DataDir=/var/lib/mysql-cluster

[ndbd]

HostName=10.1.138.97

DataDir=/var/lib/mysql-cluster

[mysqld]

[mysqld]

1. 使用配置文件初始化管理节点

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

出现MySQL Cluster Management Server mysql-5.7.16 ndb-7.5.4

1. 然后就能使用ndbd进去管理了(如果ndbd命令不行，就使用在/usr/local/bin目录下使用ndb\_mgm命令)

# ndbd

ndb\_mgm>show（使用show命令查看管理情况，当数据节点配置完毕之后，我们再用这个命令查看和管理）

到此为止管理节点配置完毕，接下去配置数据和sql节点

1. 安装配置数据和mysql节点

以下的所有操作需要在所有的集群节点都要进行相同的操作

新增用户组mysql和用户msyql

# groupadd mysql

useradd -g mysql -s /bin/false mysql

mkdir /var/lib/mysql-cluster

chown root:mysql /var/lib/mysql-cluster

1. 解压

#cd /usr/local

# tar xvf mysql-cluster-gpl-7.5.4-linux-glibc2.5-x86\_64.tar.gz

1. 创建链接方便访问

# ln -s /usr/local/mysql-cluster-gpl-7.5.4-linux-glibc2.5-x86\_64 /usr/local/mysql

1. 初始化数据库

(这里要注意，如果你安装的版本和我的不同，数据库初始化的命令使不同的，很多之前的版本会使用：scripts/mysql\_install\_db --user=mysql来初始化，这个已经被mysql在新的版本中废弃了，所以需要使用下面的命令安装，如果你需要安装别的版本请参考mysql官网的对应版本的安装命令。)

# cd /usr/local/mysql

如果下方这个命令无法使用，那么就进入bin目录下使用./mysqld --initialize进行初始化，总之正常安装mysql如何初始化就如何进行安装就可以了，这里还可以设置安装数据库的data目录等参数这里就不多解释了，网上安装mysql5.7的教程很多。

#./mysqld --user=mysql --basedir=/usr/local/mysql --datadir=/usr/local/mysql/data/ --initialize

# mysqld –initialize 就是你害得我一直报错！！！！

如果初始化成功之后，系统会提示一个随机生成的数据库密码，此时需要记住这个密码，之后登录数据库需要使用这个密码！！！

1. 修改权限

chown -R root .

chown -R mysql data

chgrp -R mysql .

cp support-files/mysql.server /etc/rc.d/init.d/

chmod +x /etc/rc.d/init.d/mysql.server

chkconfig --add mysql.server

1. 配置数据节点

# vi /etc/my.cnf

[mysqld]

ndbcluster

ndb-connectstring=10.1.138.116

[mysql\_cluster]

ndb-connectstring=10.1.138.116

其中的IP为管理节点的IP

1. 启动集群节点上面的服务启动mysql（成功会有success）# /etc/init.d/mysql.server start

启动mysql成功之后请自己登录进mysql内然后进行密码修改等操作，就和正常安装完成mysql的操作一样。需要注意的是，集群数据库的密码需要相同哦！

启动ndbd# /etc/init.d/ndbd --initial如果上述不行使用绝对路径的这个：# cd 如果出现下述现象就成功了

2017-03-06 14:04:07 [ndbd] INFO -- Angel connected to '192.168.75.129:1186'

2017-03-06 14:04:07 [ndbd] INFO -- Angel allocated nodeid: 2

1. 最后当所有的节点配置完成，回到管理节点，使用上述说过的show查看，如下的类似显示，证明已经连接完成

ndb\_mgm> show

Cluster Configuration

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

[ndbd(NDB)] 2 node(s)

id=2 (not connected, accepting connect from 192.168.75.128)

id=3 @192.168.75.130 (mysql-5.1.63 ndb-7.1.23, starting, Nodegroup: 0)

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

id=1 @192.168.75.129 (mysql-5.7.16 ndb-7.5.4)

[mysqld(API)] 2 node(s)

id=4 (not connected, accepting connect from any host)

id=5 (not connected, accepting connect from any host)

1. 集群完成后我们需要配置集群性能测试工具。我们使用flexAsynch。它是一种非常好的NDB测试工具。由于Oracle并不提供flexAsynch的二进制文件下载，仅仅在MySQL Cluster的源码包中提供这个测试工具的源码，所以必须手动编译MySQL Cluster才能获得flexAsynch的执行文件。以下命令都是在RHEL 6.4中以root账户权限运行。
2. 安装libaio-devel

#yum install libaio-devel

1. 安装cmake
2. yum install -y gcc gcc-c++ make automake
3. wget http://www.cmake.org/files/v2.8/cmake-2.8.10.2.tar.gz
4. tar -zxvf cmake-2.8.10.2.tar.gz
5. cd cmake-2.8.10.2
6. ./bootstrap
7. gmake
8. gmake install

<http://www.cnblogs.com/mophee/archive/2013/03/19/2969456.html>

1. Cmake配置

#cmake . -DBUILD\_CONFIG=mysql\_release -DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql -DWITH\_NDB\_TEST=ON -DWITH\_BUNDLED\_LIBEVENT=ON -DWITH\_BUNDLED\_MEMCACHED=ON -DWITH\_NDBMTD=ON -DWITH\_NDB\_TEST=ON

1. 编译

#make VERBOSE=1

1. 安装

#make install

1. 拷贝到MySQL Cluster管理结点目录下。

编译安装完成之后，flexAsynch位于/usr/local/mysql/bin目录中，将其拷贝至MySQL Cluster管理结点的/usr/local/mysql/bin目录或者其他合适的目录中，即可对MySQL Cluster进行性能测试。

如果运行flexAsynch时报告找不到某些lib库的错误，那么需要添加以下内容到/etc/profile文件中：

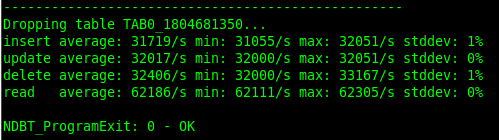
#LD\_LIBRARY\_PATH=/usr/local/mysql/lib

#export LD\_LIBRARY\_PATH

1. 可以执行以下性能测试：

#/usr/local/mysql/bin/flexAsynch -ndbrecord -temp -con 1 -t 4 -p 10 -a 2 -l 3 -r 2

1. 得到如下结果：

[](javascript:;)

# 二、参数说明

-t ：想要启动的线程总数，默认值为1

-p ：每个线程的并行事务总数，默认值为1

-o ：每个循环的事务总数，默认值为500

-l ：想要运行的循环总数，默认值为1，0表示无限循环

-load\_factor ：索引中的负载因数，以百分比表示（40 ~ 99）

-a ：属性（数据库表的列）总数，默认值为25

-c ：每次事务的操作总数

-s ：每项属性的大小，默认为1字节（主键大小总是为1字节，与这项参数值无关）

-simple ：使用简单读模式来读取数据库

-dirty ：使用脏读模式来读取数据库

-write ：在插入和更新时，使用writeTuple模式

-n ：使用标准表名

-no\_table\_create ：不要在数据库中创建表

-temp ：创建表，但不记录日志

-no\_hint ：不给出事务处理协调器执行到哪儿的提示

-adaptive ：使用自适应的发送算法（默认选项）

-force ：当通信时强制发送

-non\_adaptive ：以10毫秒的间隔时间发送

-local ：1表示每个线程都有自己的结点执行；2表示多个结点循环执行每个并行事务；3表示多个结点随机执行每个并行事务

-ndbrecord ：使用NDB记录

-r ：额外循环的总数

-insert ：对标准表仅执行插入操作

-read ：对标准表仅执行读操作

-update ：对标准表仅执行更新操作

-delete ：对标准表仅执行删除操作

-create\_table ：对标准表仅执行创建表操作

-drop\_table ：对标准表仅执行删除表操作

-warmup\_time ：测量开始之前的预热时间

-execution\_time ：测量完成时的执行时间

-cooldown\_time ：测量完成之后的冷却时间

-table ：标准表的总数，默认值为0

常用的参数有-l（测试循环次数）、-t（线程数量）、-p（并行事务数量）、-o（每次循环的事务数量）、-c（每次事务的操作数量），所以性能测试的总操作数量为l\*t\*p\*o\*c。

例如，可以执行以下性能测试：

/usr/local/mysql/bin/flexAsynch -ndbrecord -temp -con 1 -t 4 -p 10 -a 2 -l 3 -r 2

上述性能测试基于以下文章搭建的MySQL集群：

<http://blog.csdn.net/yang1982_0907/article/details/20716845>

如果出现以下错误：

Configuration error: Error: Could not alloc node id at localhost port 1186: Connection done from wrong host ip 127.0.0.1.

那么就说明没有空闲的“slot”可以接受性能测试建立的连接，这时候需要在管理结点的config.ini配置文件中添加更多的“[MYSQLD]”项，可以理解为将管理结点自身也配置为SQL结点。可以参考以下资料：

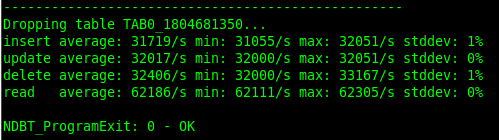
<http://serverfault.com/questions/548651/benchmark-mysql-cluster-using-flexasynch-no-free-node-id-found-for-mysqldapi>

在config.ini配置文件中添加以下内容，然后重启MySQL集群：

[MYSQLD]

NodeId=6

Hostname=192.168.124.141

上述示例性能测试得到的结果为：  
[](javascript:;)

总结

1、在实际中需要分开数据节点和sql节点。

2、安装最好参考官网的文档进行配置安装，网上的博客安装的可能为旧版本，命令可能不同，连我自己现在写的不知道你看的时候是不是官网又出了新版本，可能在某些地方进行了改动。

3、配置文件过于简单，还有很多配置会在实际中被使用到，需要参考文档进行添加。（my.cnf文件的配置需要根据你的实际进行修改）

4、架构简单，实验足够，现实远不足。（需要使用备份数据库，要备份管理节点等）

5、还未测试各种节点断开、服务器停止、数据库服务停止、是否会出现数据异常等等特殊情况，还需要进行考虑。因为一旦数据不同步，想要恢复数据会比较麻烦。（之后经过测试，数据库服务停止之后，或者服务器停止之后，只要管理节点正常，数据会之后在开启之后同步过去，但是为了防止万一还是需要做好数据备份工作）

## sysbench语法

执行sysbench –help，可以看到sysbench的详细使用方法。

sysbench的基本语法如下：

**sysbench [options]... [testname] [command]**

下面说明实际使用中，常用的参数和命令。

### （1）command

command是sysbench要执行的命令，包括prepare、run和cleanup，顾名思义，prepare是为测试提前准备数据，run是执行正式的测试，cleanup是在测试完成后对数据库进行清理。

### （2）testname

testname指定了要进行的测试，在老版本的sysbench中，可以通过--test参数指定测试的脚本；而在新版本中，--test参数已经声明为废弃，可以不使用--test，而是直接指定脚本。

例如，如下两种方法效果是一样的：

|  |  |
| --- | --- |
| 1  2 | sysbench --test=./tests/include/oltp\_legacy/oltp.lua  sysbench ./tests/include/oltp\_legacy/oltp.lua |

测试时使用的脚本为lua脚本，可以使用sysbench自带脚本，也可以自己开发。对于大多数应用，使用sysbench自带的脚本就足够了。不同版本的sysbench中，lua脚本的位置可能不同，可以自己在sysbench路径下使用find命令搜索oltp.lua。P.S.：大多数数据服务都是oltp类型的，如果你不了解什么是oltp，那么大概率你的数据服务就是oltp类型的。

### （3）options

sysbench的参数有很多，其中比较常用的包括：

**MySQL连接信息参数**

* --mysql-host：MySQL服务器主机名，默认localhost；如果在本机上使用localhost报错，提示无法连接MySQL服务器，改成本机的IP地址应该就可以了。
* --mysql-port：MySQL服务器端口，默认3306
* --mysql-user：用户名
* --mysql-password：密码

**MySQL执行参数**

* --oltp-test-mode：执行模式，包括simple、nontrx和complex，默认是complex。simple模式下只测试简单的查询；nontrx不仅测试查询，还测试插入更新等，但是不使用事务；complex模式下测试最全面，会测试增删改查，而且会使用事务。可以根据自己的需要选择测试模式。
* --oltp-tables-count：测试的表数量，根据实际情况选择
* --oltp-table-size：测试的表的大小，根据实际情况选择
* --threads：客户端的并发连接数
* --time：测试执行的时间，单位是秒，该值不要太短，可以选择120
* --report-interval：生成报告的时间间隔，单位是秒，如10

root@db2:~# sysbench --test=oltp help

sysbench 0.4.12: multi-threaded system evaluation benchmark

oltp options:

--oltp-test-mode=STRING 执行模式{simple,complex(advanced transactional),nontrx(non-transactional),sp}。默认是complex

--oltp-reconnect-mode=STRING 重新连接模式{session(不使用重新连接。每个线程断开只在测试结束),transaction(在每次事务结束后重新连接),query(在每个SQL语句执行完重新连接),random(对于每个事务随机选择以上重新连接模式)}。默认是session

--oltp-sp-name=STRING 存储过程的名称。默认为空

--oltp-read-only=[on|off] 只读模式。Update，delete，insert语句不可执行。默认是off

--oltp-skip-trx=[on|off] 省略begin/commit语句。默认是off

--oltp-range-size=N 查询范围。默认是100

--oltp-point-selects=N number of point selects [10]

--oltp-simple-ranges=N number of simple ranges [1]

--oltp-sum-ranges=N number of sum ranges [1]

--oltp-order-ranges=N number of ordered ranges [1]

--oltp-distinct-ranges=N number of distinct ranges [1]

--oltp-index-updates=N number of index update [1]

--oltp-non-index-updates=N number of non-index updates [1]

--oltp-nontrx-mode=STRING 查询类型对于非事务执行模式{select, update\_key, update\_nokey, insert, delete} [select]

--oltp-auto-inc=[on|off] AUTO\_INCREMENT是否开启。默认是on

--oltp-connect-delay=N 在多少微秒后连接数据库。默认是10000

--oltp-user-delay-min=N 每个请求最短等待时间。单位是ms。默认是0

--oltp-user-delay-max=N 每个请求最长等待时间。单位是ms。默认是0

--oltp-table-name=STRING 测试时使用到的表名。默认是sbtest

--oltp-table-size=N 测试表的记录数。默认是10000

--oltp-dist-type=STRING 分布的随机数{uniform(均匀分布),Gaussian(高斯分布),special(空间分布)}。默认是special

--oltp-dist-iter=N 产生数的迭代次数。默认是12

--oltp-dist-pct=N 值的百分比被视为'special' (for special distribution)。默认是1

--oltp-dist-res=N ‘special’的百分比值。默认是75

General database options:

--db-driver=STRING 数据库类型，指定数据库驱动程序('help' to get list of available drivers)

--db-ps-mode=STRING 数据库预处理模式{auto, disable} [auto]

Compiled-in database drivers:

mysql - MySQL driver

mysql options:

--mysql-host=[LIST,...] MySQL server host [localhost]

--mysql-port=N MySQL server port [3306]

--mysql-socket=STRING MySQL socket

--mysql-user=STRING MySQL user [sbtest]

--mysql-password=STRING MySQL password []

--mysql-db=STRING MySQL database name [sbtest]

--mysql-table-engine=STRING storage engine to use for the test table {myisam,innodb,bdb,heap,ndbcluster,federated} [innodb]

--mysql-engine-trx=STRING whether storage engine used is transactional or not {yes,no,auto} [auto]

--mysql-ssl=[on|off] use SSL connections, if available in the client library [off]

--myisam-max-rows=N max-rows parameter for MyISAM tables [1000000]

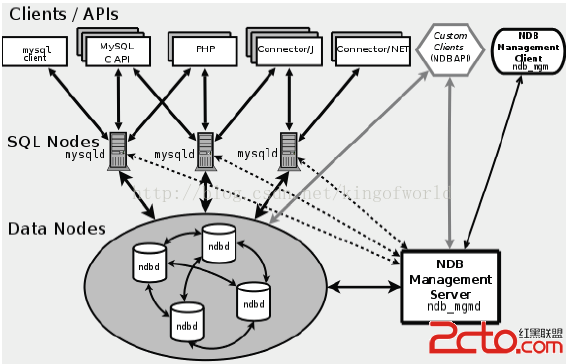
--mysql-create-options=STRING additional options passed to CREATE TABLE []

## .MySQLCluster

**简介：**

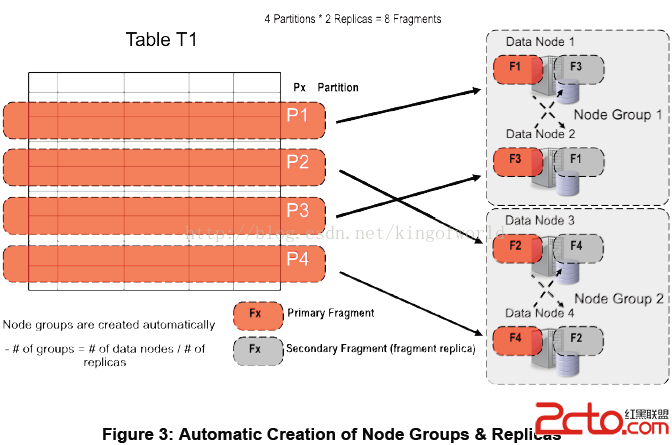
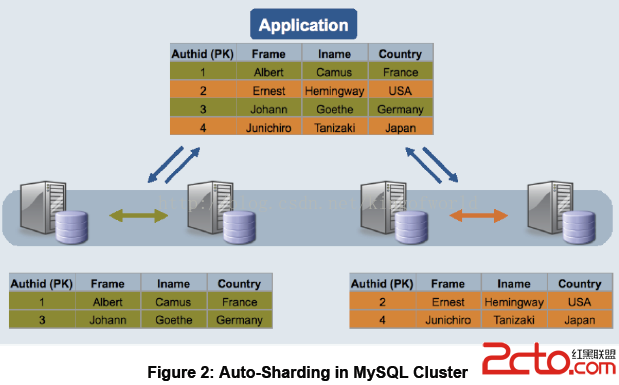
MySQL Cluster 是MySQL 官方集群部署方案，它的历史较久。支持通过自动分片支持读写扩展，通过实时备份冗余数据，是可用性最高的方案，声称可做到99.999%的可用性。

架构及实现原理：

****

MySQL cluster主要由三种类型的服务组成：

NDB Management Server：管理服务器主要用于管理cluster中的其他类型节点（Data Node和SQL Node），通过它可以配置Node信息，启动和停止Node。 SQL Node：在MySQL Cluster中，一个SQL Node就是一个使用NDB引擎的mysql server进程，用于供外部应用提供集群数据的访问入口。Data Node：用于存储集群数据；系统会尽量将数据放在内存中。

****

缺点及限制：

对需要进行分片的表需要修改引擎Innodb为NDB，不需要分片的可以不修改。NDB的事务隔离级别只支持Read Committed，即一个事务在提交前，查询不到在事务内所做的修改；而Innodb支持所有的事务隔离级别，默认使用Repeatable Read，不存在这个问题。外键支持：虽然最新的Cluster版本已经支持外键，但性能有问题（因为外键所关联的记录可能在别的分片节点中），所以建议去掉所有外键。Data Node节点数据会被尽量放在内存中，对内存要求大。