# cassandra集群搭建实战 - ch648966459的专栏 - 博客频道

分类：

Cassandra（1）

目录(?)[+]1. 基础配置与安装1.1 基础环境

node110.202.20.191 (seed1)

node210.202.20.192

node310.202.20.193

node410.202.20.194

node510.202.20.195

node610.202.20.196

node710.202.20.197

node810.202.20.198 (seed2)

以上服务器基于现在的云平台(32核，96G内存),OS:centos6.6

1.1.1 安装JAVA

yum -y installjdk-8u51-Linux-x64.rpm

rm /usr/bin/Java

ln -s /usr/java/jdk1.8.0_51/bin/java/usr/bin/java

## 1.1.2 OS环境配置

在/etc/sysctl.conf中添加以下三行：

vm.zone_reclaim_mode=0

vm.max_map_count = 262144

vm.swappiness = 1

执行sysctl -p让配置生效

在/etc/security/limits.conf 中添加以下四行：

* soft nofile 65536

```
*               hard       nofile  65536

*               soft       nproc    65536

*               hard       nproc    65536
```

在/etc/hosts加入以下信息：

```
10.202.20.191 ras1.novalocal

10.202.20.192 ras2.novalocal

10.202.20.193 ras3.novalocal

10.202.20.194 ras4.novalocal

10.202.20.195 ras5.novalocal

10.202.20.196 ras6.novalocal

10.202.20.197 ras7.novalocal

10.202.20.198 ras8.novalocal
```

### 1.1.3 添加用户

```
useradd cassandra
```

在/home/cassandra/.bash_profile中添加

```
export JAVA_HOME=/usr/java/jdk1.8.0_51
```

修改path

```
PATH=$PATH:$HOME/bin:/opt/cassandra/bin
```

### 1.1.4 配置存储

```
mkdir /data1

mkdir /data2

mount /dev/vdb1 /data1

mount /dev/vdc1 /data2

chmod 777 /data1

chmod 777 /data2
```

## 1.2 cassandra安装

1.2.1 安装

http://cassandra.apache.org/download/下载对应的版本

```
tar xvf apache-cassandra-2.1.12-bin.tar.gz -C /opt

mv /opt/apache-cassandra-2.1.12 /opt/cassandra

chown -R cassandra.cassandra /opt/cassandra
```

1.2.2 配置

```
su - cassandra
```

准备几个目录，其中commitlog用户存放交易日志，建议使用单独的盘存放，dataNfile用于存放数据文件，建议使用SSD，如果是硬盘，建议用raid0，saved_casches用于存放查询缓存，可以与数据文件放在同一个盘上

```
mkdir /data1/commitlog

mkdir /data1/data1file

mkdir /data2/data2file

mkdir /data2/saved_caches
```

配置参数文件，在配置前，做个备份

```
cp cassandra.yaml cassandra.yaml.bak
```

修改参数文件

集群名称：

```
cluster_name: 'FVP-RAS Cluster'
```

数据文件：

如果存放位置多个地方，写多行

```
data_file_directories:

  - /data1/data1file

  - /data2/data2file
```

交易日志：

```
commitlog_directory: /data1/commitlog
```

查询缓存：

```
saved_caches_directory: /data2/saved_caches
```

种子服务器IP：
多个以逗号分开

```
- seeds: "10.202.20.191,10.202.20.198"
```

监听地址：
为空，会使用主机名对应的IP地址

```
listen_address:
```

为空，会使用主机名对应的IP地址

rpc通讯地址：
为空，会使用主机名对应的IP地址

配置远程JMX访问
修改cassandra-env.sh

```
LOCAL_JMX=no
```

以下操作需要root权限

```
mkdir -p /etc/cassandra
```

```
cp/usr/java/jdk1.8.0_51/jre/lib/management/jmxremote.password.template
/etc/cassandra/jmxremote.password
```

```
chown cassandra.cassandra /etc/cassandra/jmxremote.password
```

```
chmod 400 /etc/cassandra/jmxremote.password
```

```
echo "cassandra cassandrapassword">>/etc/cassandra/jmxremote.password
```

```
sed -i '/controlRole/icassandra
readwrite'/usr/java/jdk1.8.0_51/jre/lib/management/jmxremote.access
```

配置用户与权限
修改cassandra.yaml中的以下两行

```
authenticator: PasswordAuthenticator
```

```
authorizer: CassandraAuthorizer
```

在cassandra启动后，在第一个seed节点执行以下命令：

```
echo "ALTER KEYSPACE system_auth WITHREPLICATION=
{'class':'org.apache.cassandra.locator.SimpleStrategy','replication_factor':'8'};"|cqlsh
10.202.20.191   -ucassandra -pcassandra
```

注：上面8为整个集群节点数，因为权限认证表需要全集群同步

```
echo "create user admin with password '123456' superuser;"|cqlsh 10.202.20.191  -ucassandra -pcassandra

echo "drop user cassandra;"|cqlsh 10.202.20.191   -uadmin -p123456
```

2. 操作2.1 启动

/opt/cassandra/bin/cassandra 默认是在后台运行,如果想在前台运行,请使bin/cassandra -f ,另外在启动时,请先启动一个seed的服务器实例,再启动其它节点的实例

启动后，可以通地/opt/cassandra/logs/ system.log来查看相关信息，动态查看命令如下：

```
tail -f /opt/cassandra/logs/system.log
```

## 2.2 CQL使用

```
cqlsh 10.202.20.191 -uadmin -p123456
```

可在cqlsh命令提示符下输入help或？获得帮助，输入exit或quit退出cqlsh，命令默认以“;”结束。

查看keyspace

```
DESC keyspaces;
```

创建keyspace

```
CREATE KEYSPACE ks_test WITHREPLICATION = { 'class' :'NetworkTopologyStrategy', 'DC1' : 3,'DC2' : 3};
```

注：这里3为复制因子，我们这里统一定义为3，系统认证表空间除外

切换keyspace

```
USE ks_test;
```

创建表

```
CREATE TABLE tt_table1(
   id int PRIMARY KEY,
   c1 text
);
```

查看表

```
DESC TABLES;
```

插入数据

```
INSERT INTO tt_table1(id,   c1)
   VALUES (1, 'test1');
INSERT INTO tt_table1(id,   c1)
   VALUES (2, 'test2');
INSERT INTO tt_table1(id,   c1)
   VALUES (3, 'test3');
```

查询数据

```
SELECT * FROM tt_table1;
```

建立索引后使用WHERE从句查找

```
CREATE INDEX ON tt_table1(c1);
SELECT * FROM tt_table1 WHEREc1 = 'test2';
```

删除表

```
DROP TABLE tt_table1;
```

2.3 监控2. 2.1 2.2 2.3 2.3.1 nodetool utility
Cassandra发行版附带的命令行工具，用于监控和常规数据库操作。一些常用命令如status、cfstats、cfhistograms、netstats、tpstats等。

2.3.2 DataStax OpsCenter
图形用户界面工具，从中央控制台监控和管理集群中所有节点,分为企业版与普通版，企业版收费

2.3.3 JConsole
JMX兼容工具用以监控java应用程序，提供Overview、Memory、Thread、Classes、VM summary、Mbeans方面的信息

2.4 开启数据缓存
两类cache：partitionkey cache和row cache

partition key cache：Cassandra表partition index的cache

row cache：一个row首次被访问后整个row(合并自多个对应的SSTable及memtable)被放在row cache中以便于后续对改row的访问能直接由内存获取数据。 对于很少访问的archive表当禁用缓存。

开启

CREATE TABLE users (  useridtext PRIMARY KEY, first_name text, last_name text,)

WITH caching ='all';

配置

在cassandra.yaml中，调整下列参数；

key_cache_keys_to_save

key_cache_save_period

key_cache_size_in_mb

row_cache_keys_to_save

row_cache_size_in_mb

row_cache_save_period

row_cache_provider

工作原理：

第一个read操作直接命中rowcache，从内存取出数据。第二个read操作为命中row cache，但命中partition key cache，并由此整合所有相关的SSTable及memtable中的片段为请求的row，返回row并写进row cache。

优化建议:将很少请求的数据或row很长的数据放在cache较小或不使用cache的表中

Ø 用较多的节点部署各节点负载较轻的集群

Ø 逻辑上将read稠密的数据分开在离散的表中

## 2.5 配置memtable吞吐量

可改善write性能。Cassandra在commit logspace threshold超出时将memtables内容刷进磁盘创建SSTable。在cassandra.ymal中配置commit log space threshold来调整memtable吞吐量。配置的值依赖于数据和write负载。下列两种情况可考虑增加吞吐量：

Ø write负载包含大量在小数据集上的更新操作

Ø 稳定持续的写操作

### 2.6 修复 node

使用nodetool的repair命令，修复与给定的数据范围相关的replica间的不一致性。在下属情形运行修

复：

Ø 规律的、计划的集群维护操作期间（除非没有执行过delete）

Ø 节点恢复后

Ø 在包含较少被访问的数据的节点上

Ø 在down掉的节点更新数据

运行节点修复的方针：

Ø 常规修复操作执行次数由gc_grace值硬性规定。需在该时间内在每个节点至少运行一次修复操作。

Ø 同时在多于一个的节点运行常规修复时需谨慎，最好在低峰期规律运行修复。

Ø 在很少delete或overwrite数据的系统中，可增加gc_grace的值。

修复操作消耗磁盘I/O，可通过下述途径减轻：

Ø 使用nodetool的compactionthrottling选项。

Ø 使用nodetool snapshot之后从snapshot执行修复。

修复操作会导致overstreaming（问题源于Cassandra内部Merkletrees数据结构）。比如只有一个损坏的partition但却需发送很多的stream，且若节点中存在的partition越多问题越严重。这会引起磁盘空间浪费和不必要的compaction。可使用subrange 修复来减轻overstreaming。subrange repair只修复属于节点的部分数据。

## 2.7 添加/移除节点或数据中心

2. 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.7.1. 在既存集群中添加节点
以使用vnodes的节点为例:

Ø 在新节点安装Cassandra，关闭cassandra，清除数据。

Ø 在cassandra.yaml和cassandra-topology.properties中配置如下属性：

cluster_name

listern_address/broadcast_address

endpoint_snitch

num_tokens

seed_provider

Ø 逐个启动新节点中的cassandra，不同节点的初始化之间保持两分钟的间隔。可用nodetool stats监控启动和数据流处理待所有新节点运行起来后逐个在每个之前已存在的节点执行nodetoolcleanup来移除不再属于这些节点的key。在下一个节点执行nodetool cleanup前必须等上一个节点中的nodetool cleanup结束。另外cleanup操作可考虑在低峰期进行。

2.7.2. 向既存集群中添加数据中心
以使用vnodes的节点构成的集群为例

Ø 确保keyspace使用NetworkTopologyStrategy复制策略

Ø 对于每个新节点在cassandra.yaml中设置如下属性

auto_bootstrap:false.

Ø 设置其他与所属集群匹配的属性（参见上一部分：在既存集群中添加节点）

Ø 根据设置的snitch类型在各新节点配置对应的指明网络拓扑的配置文件（无需重启）

Ø 确保使用的客户端不会自动探测新的节点。

Ø 若使用QUORUM一致性级别，需检查LOCAL_QUORUM或EACH_QUORUM一致性级别是否满足多数据中心的要求

Ø 逐个在每个节点开启cassandra。注意初始化时间间隔。

2.7.3. 替换死掉的节点
以使用vnodes的节点构成的集群为例

Ø 用nodetool status命令确认环中死掉的节点，从输出中记录该节点的HOSTID。

Ø 添加并启动新的替代节点

Ø 使用nodetool removenode命令根据记录的死亡节点的HOSTID从集群中移除死掉的节点。

2.7.4. 移除数据中心
Ø 确认没有client正在向数据中心内的任何节点写数据

Ø 在数据中心内的各节点中执行nodetoolrepair以确保数据从该中心得到正确的传播。更改所有的keyspace属性确保不再引用即将移除的数据中心。

Ø 在数据中心内的各节点分别运行nodetooldecommission

2.8 备份恢复
Cassandra通过为磁盘中的数据文件（SSTable文件）创建快照来备份数据。可为单个表、单个keyspace、所有keyspace创建快照。可用并行SSH工具为整个集群创建快照。创建时不保证所有replica一致，但在恢复快照时Cassandra利用内建的一致性机制保持一致性。创建了系统范围的快照后可开启增量备份只备份自上次快照以来变化了的数据（每当一个SSTable被flush后，一个对应的硬链接被拷贝至

与/snapshot同级的/backups子目录），快照通过硬链接创建。否则会因将文件拷贝至不同的位置而增加磁盘I/O。

2.8.1 创建快照

在每个节点执行nodetoolsnapshot命令为节点创建快照。也可通过并行SSH工具（如pssh）运行nodetool snapshot创建全局的快照。

```
nodetool -hlocalhost -p   snapshot demdb
```

执行命令后首先会将内存中的数据刷进磁盘，之后为每个keyspace的SSTable文件创建硬链接。快照的默认位置为/var/lib/cassandra/data/<keyspace_name>/<table_name>/snapshots。其中/var/lib/cassandra/data部分依据数据目录设置而不同。要保证空间充足，创建后可考虑移至其他位置。

2.8.2 删除快照

创建新的快照并不会自动删除旧的快照，需在创建新快照前通过nodetool clearsnapshot命令移除旧的快照。

```
nodetool -hlocalhost -p   clearsnapshot
```

同样可通过并行SSH工具（如pssh）运行nodetoolclearsnapshot一次删除所有节点的快照。

2.8.3 启用增量备份

默认不开启，可通过在各节点的cassandra.yaml配置文件中设置incremental_backups为true来开启增量备份。开启后会为每个新的被刷入的SSTable创建一个硬链接并拷贝至数据目录的/backups子目录。Cassandra不会自动删除增量备份文件，创建新的快照前需手工移除旧的增量备份文件。

2.8.4 从快照恢复数据

需所有的快照文件，若使用了增量备份还需快照创建之后所有的增量备份文件。通常，在从快照恢复数据前需先truncate表。（若备份发生在delete前而恢复发生在delete后且没truncate表时，则不能得到最原始的数据，因为直到compaction操作发生前被标记为tombstone的数据与原始数据处于不同的SSTable中，所以恢复包含原始数据的SSTable不会移除被标记被tombstone的数据，这些数据仍然显示为将被删除）。

可以用如下方式从快照恢复数据

使用sstableloader工具:

http://www.datastax.com/documentation/cassandra/0/webhelp/cassandra/tools/toolsBulkloader_t.html

先拷贝snapshot目录中的快照文件至相应数据目录。之后通过JConsole调用column family MBean 中的JMX方法loadNewSSTables()或者使用nodetool refresh命令而不调用上述方法。

使用重启节点方式:

Ø 若恢复单节点则先关闭该节点，若恢复整个集群则需先关闭所有节点

Ø 清除/var/lib/cassandra/commitlog中的所有文件

Ø 删除<data_directory_location>/<keyspace_name>/<table_name>中所有*.db文件

Ø 拷贝最新<data_directory_location>/<keyspace_name>/<table_name>/snapshots/<snapshot_name>的快照文件至<data_directory_location>/<keyspace_name>/<table_name>/snapshots/<snapshot_name>

Ø 若使用了增量备份则还需拷贝<data_directory_location>/<keyspace_name>/<table_name>/backups中的内容至<data_directory_location>/<keyspace_name>/<table_name>

Ø 重启节点

Ø 运行nodetool repair

2.9 日常运维2.9.1 repair

cassandra的删除操作，实际上并不是真的删除，它是执行的插入操作，插入的数据叫做tombstone（墓碑)，记录了被删除记录的信息和删除时间。根据条件查询的时候，如果它会把满足条件的记录查询出来，包括tombstone。然后过滤掉删除的记录，再把结果返回,这样的场景下，导至频繁删除的表的膨涨非常历害，带来性能问题

通过nodetoolrepair操作来解决。

基本语法：

nodetool -h hostrepair [keyspace] [table names]

使用-pr选项，表示只修复range落在该节点的master数据

nodetool -h host-pr repair [keyspace] [table names]

数据量足够大的时候，这个过程还是很慢，你可能焦急的不知道这个过程什么时候结束。你可以通过token段的方式，

nodetool -h host-st xxx -et xxxx repair [keyspace] [table names]

可以加一个-par选项,多线程并发repair。

nodetool -h host-pr -par repair [keyspace] [table names]

nodetool -h host-st xxx -et xxxx -par repair [keyspace] [table names]

可以加一个-inc选项,增量repair

nodetool -h host-pr -inc repair [keyspace] [table names]

nodetool -h host-st xxx -et xxxx -inc repair [keyspace] [table names]

注意-inc和-par是不能一起使用的。

为了防止数据重现，我们定义每3天一次进行增量repair（nodetool repair -pr -par -inc），对于停机超过4天服务器，需要清除此节点所有数据，重建此节点，默认的GC时间为10天（gc_grace_seconds = 864000），我们定义不修改此参数

顶
0