[Cassandra教程]（五）Cassandra自带工具介绍

时间 2016-09-06 13:58:02 FlyML
原文
主题 Cassandra

Cassandra 自带了多个集群或数据管理工具，本文接下来简要介绍一下，欲知详情还是推荐大家自己啃一遍英文文档。下面是本文将会介绍到的工具：

- nodetool utility
- CQL shell
- cassandra utility
- cassandra-stress tool
- SSTable utilities

一、Nodetool

nodetool 是一个进行集群管理的利器，功能强大繁杂，通过命令行方式操作，标准使用方式如下：

```
$ nodetool [options] command [args]
```

下面是一个常用option list

Short	Long	Description
-h	--host	Hostname or IP address
-p	--port	Port number
-pwf	--password-file	Password file path
-pw	--password	Password
-u	--username	User name

如果是tar包方式安装，nodetool工具位于安装目录的./bin 目录内。

通常情况需要通过 -h+ip 指定nodetool操作对应的集群中node节点，但是如果你要操作的都是当前节点，也可以省略掉这个选项。

nodetool操作内容炒鸡炒鸡的多，保准看花了你的眼，所以我们需要知道他的 help 命令 ，如下：

```
# 列出nodetool所有可用的命令
$ nodetoolhelp
```

```
# 列出指定command 的帮助内容
$ nodetoolhelpcommand-name
# 例如：查看status 命令的详细帮助内容
$ nodetoolhelpstatus
```

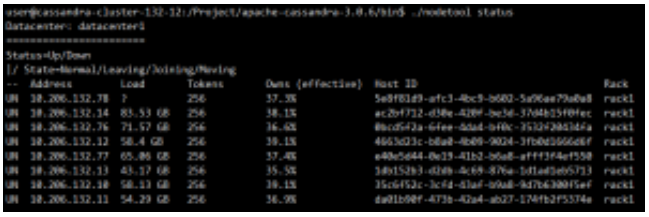
请大家务必记住上面个两个help命令，常用且实用。

下面介绍一下常用的几个nodetool command：

1、查看集群运行状态

```
$ nodetoolstatus
```

执行结果类似下图：



2、移除某个废弃节点

```
# 命令模板
$ nodetool <options> removenode -- <status> | <force> | <ID>

# 使用实例
# 移除Host ID为d0844a21-3698-4883-ab66-9e2fd5150edd的节点
$ nodetoolremovenoded0844a21-3698-4883-ab66-9e2fd5150edd
# 查看节点删除状态
$ nodetoolremovenodestatus
```

3、其他

```
# 参看某个节点负载，内存使用情况
$ nodetoolinfo

# 查看各个CF的详细统计信息，包括读写次数、响应时间、memtable信息等
$ nodetoolcfstats

# 其他
$ nodetoolhelp
```

原文链接 ： http://www.flyml.net/2016/09/06/cassandra-tutorial-tools-used

二、CQL shell

cqlsh是一个通过CQL（ Cassandra Query Language）来与Cassandra集群中的数据进行交互的命令行工具，可以在 ./bin 目录内找到。

cqlsh本身基于python 2.7，所以可能需要事先有python 2.7环境（如果 采用python 2.7.11+版本可能会报错 ‘ref() does not take keyword arguments’，可以将python降级，或者升级Cassandra版本即可解决，参见 连接 ）。

cqlsh的help类似nodetool，单help 列出所有命令，help + command列出该命令详细信息

```
# 连接本机cql shell，如果Cassandra.yaml 配置了listen 的rpc_address和rpc_port，
# 则需连接此ip-port
$ cqlsh [options] [host [port]]
```

下面通过一个实例介绍一下cqlsh的基本使用，实例参考自 此处 。

1、进入到命令行交互模式，查看当前的keyspace有哪些

```
cqlsh> desckeyspaces;

system    mykeyspace    OpsCenter    system_traces
```

2、使用SimpleStrategy策略创建一个新的keyspace，Cassandra里的keyspace对应MySQL里的database的概念，这种策略不会区分不同的数据中心和机架，数据复制份数为2，也就是说同一份数据最多存放在两台机器上

```
cqlsh> CREATEKEYSPACEtestks
 WITHreplication = {
    'class': 'SimpleStrategy',
    'replication_factor': '2'
};
```

Cassandra中之前是没有表的概念，之前叫Column Family，现在这个概念逐渐被淡化，像CQL中就直接称作Table，和传统数据库中表是一个意思。但是和传统数据库表的明显区别是必须有主键，因为一个表的数据可能是分布在多个机器上的，Cassandra使用主键来做分区，常用的分区方法有Murmur3Partioner、RandomPartitioner、ByteOrderedPartioner，一般默认使用第一个它的效率和散列性更好。还一个非常让人振奋的特性是列支持List、Set、Map三种集合类型，不仅仅是整形、字符串、日期等基本类型了，这给很多数据存储带来极大方便，比如一个用户帐号对应多个Email地址，或者一个事件对应多个属性等，就可以分别使用List和Map来表示，并且支持对集合的追加操作，这对一些追加的场景就特别方便，比如我们在做Velocity计算时，同一个Key值往往对应多条记录，比如记录一个IP过去3个月所有的登陆信息，就可以放在List中来表示，而不用拆成多条来存储了。

3、创建一个表

```
cqlsh:testks> createtablemytab (idtext,List<text>) ;
```

```
cqlsh:testks> desctablemytab;
CREATETABLEmytab (
    idtext,
    valueslist<text>,
    PRIMARYKEY (id)
) WITH
    bloom_filter_fp_chance=0.010000 AND
    caching='KEYS_ONLY' AND
    comment='' AND
    dclocal_read_repair_chance=0.000000 AND
    gc_grace_seconds=864000 AND
    index_interval=128 AND
    read_repair_chance=0.100000 AND
    replicate_on_write='true' AND
    populate_io_cache_on_flush='false' AND
    default_time_to_live=0 AND
    speculative_retry='99.0PERCENTILE' AND
    memtable_flush_period_in_ms=0 AND
    compaction={'class': 'SizeTieredCompactionStrategy'} AND
    compression={'sstable_compression': 'LZ4Compressor'};
```

bloom_filter_fp_change：在进行读请求操作时，Cassandra首先到Row Cache中查看缓存中是否有结果，如果没有会看查询的主键是否Bloom filter中，每一个SSTable都对应一个Bloom filter，以便快速确认查询结果在哪个SSTable文件中。但是共所周知，Bloom filter是有一定误差的，这个参数就是设定它的误差率。

caching：是否做Partition Key的缓存，它用来标明实际数据在SSTable中的物理位置。Cassandra的缓存包括Row Cache、Partitioin Key Cache，默认是开启Key Cache，如果内存足够并且有热点数据开启Row Cache会极大提升查询性能，相当于在前面加了一个Memcached。

memtable_flush_period_in_ms：Memtable间隔多长时间把数据刷到磁盘，实际默认情况下Memtable一般是在容量达到一定值之后会被刷到SSTable永久存储。

compaction：数据整理方式，Cassandra进行更新或删除操作时并不是立即对原有的旧数据进行替换或删除，这样会影响读写的性能，而是把这些操作顺序写入到一个新的SSTable中，而在定期在后台进行数据整理，把多个SSTable进行合并整理。合并的策略有SizeTieredCompactionStrategy和LeveledCompactionStrategy两种策略，前者比较适合写操作比较多的情况，后者适合读比较多的情况。

compression：是否对存储的数据进行压缩，一般情况下数据内容都是文本，进行压缩会节省很多磁盘空间，但会稍微消耗一些CPU时间。除了LZ4Compressor这种默认的压缩方式外，还有SnoopyCompressor等压缩方式，这种是Google发明的，号称压缩速度非常快，但压缩比一般。

4、插入一条数据到数据表中

```
cqlsh:testks> insertintomytab(id,values) values('100',['hello','world']);
cqlsh:testks> select * frommytab;

 id    | values
-----+--------------------
 100 | ['hello', 'world']

(1 rows)
```

5、追加一个值到List元素中

```
cqlsh:testks> updatemytabsetvalues=values+['My name is Brian'] whereid='100';
cqlsh:testks> select * frommytab;

 id    | values
-----+----------------------------------------
 100 | ['hello', 'world', 'My name is Brian']
```

6、对某一行数据或集合中的某一个值设置过期时间，单位为秒

```
cqlsh:testks> updatemytabusingttl 60 setvalues=values+['Who are you'] whereid='100';
cqlsh:testks> select * frommytab;

 id    | values
-----+------------------------------------------------------
 100 | ['hello', 'world', 'My name is Brian', 'Who are you']

(1 rows)

cqlsh:testks> select * frommytab;

 id    | values
-----+----------------------------------------
 100 | ['hello', 'world', 'My name is Brian']
```

原文链接 : http://www.flyml.net/2016/09/06/cassandra-tutorial-tools-used

三、cassandra utility

Cassandra utility 其实是Cassandra提供的一个启动时配置接口，也就是说通过这个方法可以配置C*启动参数，例如运行时java heap size等。

根据安装方式的不同，C*工具使用方法也略有不同，如下：

```
# 1、在 conf/cassandra-env.sh 中添加运行参数
# 适用于：tar包安装或pachage安装
JVM_OPTS="$JVM_OPTS -D[PARAMETER]"

# 2、在启动C*的命令行后直接跟运行参数
# 适用于：tar包安装
$ cassandra [PARAMETERS]

# 例如：
cassandra-env.sh: JVM_OPTS="$JVM_OPTS -Dcassandra.load_ring_state=false"
命令行: $ bin/cassandra -Dcassandra.load_ring_state=false
```

下面是一些 仅支持命令行方式 的选项：

Option	Description
-f	在前台启动Cassandra进程，默认后台进程启动
-h	Help.
-p filename	将C*进程的进程ID记录在filename文件中，方便后期根据进程ID来停止C*进程
-v	打印出版本信息然后退出

举几个使用例子：

```
#替换死亡节点
Commandline: bin/cassandra -Dcassandra.replace_address=10.91.176.160
cassandra-env.sh: JVM_OPTS="$JVM_OPTS -Dcassandra.replace_address=10.91.176.160"

#启动节点但并不加入ring
Commandline: bin/cassandra -Dcassandra.join_ring=false
cassandra-env.sh: JVM_OPTS="$JVM_OPTS -Dcassandra.join_ring=false"

#启动节点时清除gossip state
Commandline: $ bin/cassandra -Dcassandra.load_ring_state=false
cassandra-env.sh: JVM_OPTS="$JVM_OPTS -Dcassandra.load_ring_state=false"
```

更多的使用实例可以参考 这儿

原文链接 ： http://www.flyml.net/2016/09/06/cassandra-tutorial-tools-used

四、cassandra-stress tool

C* stress tool是一个针对集群进行压力测试的工具，基于java，工具本身没怎么用过，所以了解比较

浅，在这儿简单介绍一下，首先，使用：

```
# Package 安装:
$ cassandra-stresscommand [options]
# Tarball 安装:
$ cdinstall_location/tools
$ bin/cassandra-stresscommand [options]
```

一些常用option

Command	Description
counter_read	Multiple concurrent reads of counters. The cluster must first be populated by a counter_write test.
counter_write	Multiple concurrent updates of counters.
help	Display help: `cassandra-stress help` Display help for an option: `cassandra-stress help [options]` For example: `cassandra-stress help -schema` 注：每个option都支持sub-option，具体支持哪些以及怎么使用可以用 $ cassandra-stress help option 来查看
legacy	Legacy support mode.
mixed	Interleave basic commands with configurable ratio and distribution. The cluster must first be populated by a write test.
print	Inspect the output of a distribution definition.
read	Multiple concurrent reads. The cluster must first be populated by a write test.
user	Interleave user provided queries with configurable ratio and distribution.
write	Multiple concurrent writes against the cluster.

下面举一个简单的读写测试例子：

```
#写入1m行数据
$ cassandra-stresswrite n=1000000 -ratethreads=50
```

```
# 读200k行
$ cassandra-stressread n=200000 -ratethreads=50

# 以3分钟的间隔来读取数据
$ cassandra-stressreadduration=3m -ratethreads=50

# Read 200,000 rows without a warmup of 50,000 rows first.
# 无50k预热情况下读取200k行
$ cassandra-stressread n=200000 no-warmup -ratethreads=50
```

更多内容可以参考Help或 这儿

原文链接 ： http://www.flyml.net/2016/09/06/cassandra-tutorial-tools-used

五、SSTable utilities

sstable utilities 是一组操作sstable的工具，大致介绍如下：

- sstabledump
  Dump the contents of the specified SSTable in JSON format——将指定的SSTable内容以JSON格式导出
- sstableexpiredblockers
  The sstableexpiredblockers utility will reveal blocking SSTables that prevent an SSTable from dropping.——列出因为阻塞而没有即使处理丢弃的sstable 列表

- sstablekeys
  The sstablekeys utility dumps table keys.——导出table的key
- sstablelevelreset
  The sstablelevelreset utility will reset the level to 0 on a given set of SSTables.——将指定的sstable lever设为0
- sstableloader (Cassandra bulk loader)
  Provides the ability to bulk load external data into a cluster, load existing SSTables into another cluster with a different number of nodes or replication strategy, and restore snapshots.——提供了一种可以跨集群迁移数据的方案，细节可以点击链接查看
- sstablemetadata
  The sstablemetadata utility prints metadata about a specified SSTable.——输出指定sstable的metadata
- sstableofflinerelevel
  The sstableofflinerelevel utility will relevel SSTables.
- sstablerepairedset
  The sstablerepairedset utility will reset the level to 0 on a given set of SSTables.——将一组给定的sstable lever置0
- sstablescrub
  An offline version of nodetool scrub. It attempts to remove the corrupted parts while

preserving non-corrupted data.——sstable擦洗工具，试图删除损坏的数据同时保留正常数据

- sstablesplit
  Use this tool to split SSTables files into multiple SSTables of a maximum designated size.——sstable分割工具
- sstableupgrade
  Upgrade the SSTables in the specified table or snapshot to match the currently installed version of Cassandra.——针对C*版本升级，将旧版本的sstable 升级一下以兼容新版
- sstableutil
  The sstableutil utility will list the SSTable files for a provided table.——针对指定的表列出sstable文件
- sstableverify
  The sstableverify utility will verify the SSTable for a provided table.——针对指定的表来对sstable进行验证

原文链接 ： http://www.flyml.net/2016/09/06/cassandra-tutorial-tools-used

感谢你看到了最后，附赠一个比较简陋的monitor 脚本，用来监控节点状态，如果有挂了的可以邮件通知，需要自定义Cassandra目录及邮件服务器，如下：

The end

本文参考大量参考文献，参考内容均已附注与文中，如有侵权或异议请及时联系本文作者，多谢。

本文为原创文章，转载请注明 出处：http://www.flyml.net

分享

收藏 纠错