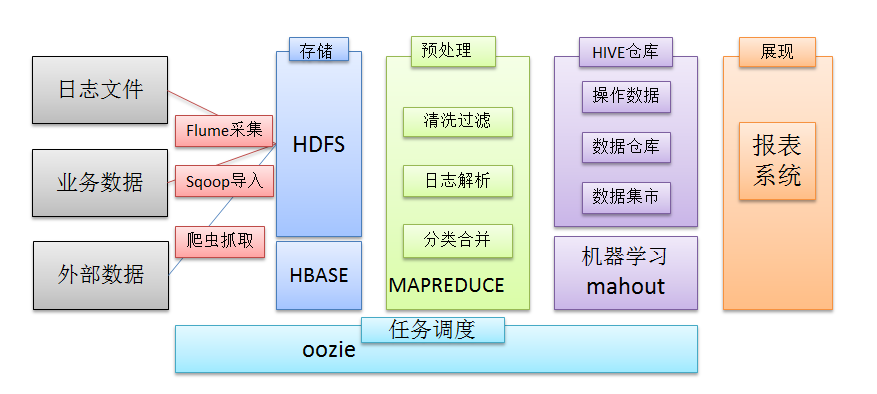
# HADOOP辅助工具、HA部署

# 前言

在一个完整的离线大数据处理系统中，除了hdfs+mapreduce+hive组成分析系统的核心之外，还需要数据采集、结果数据导出、任务调度等不可或缺的辅助系统，而这些辅助工具在hadoop生态体系中都有便捷的开源框架，如图所示：

**图：典型大规模离线数据处理平台**



# 1. Flume日志采集框架

## 1.1 Flume介绍

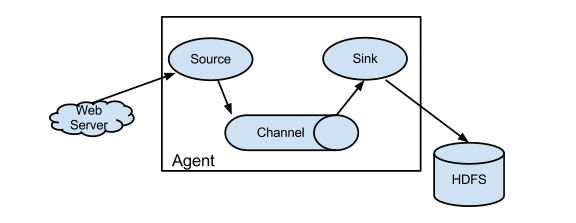
### 1.1.1 概述

* Flume是一个分布式、可靠、和高可用的海量日志采集、聚合和传输的系统。
* Flume可以采集文件，socket数据包、文件、文件夹、kafka等各种形式源数据，又可以将采集到的数据(下沉sink)输出到HDFS、hbase、hive、kafka等众多外部存储系统中
* 一般的采集需求，通过对flume的简单配置即可实现
* Flume针对特殊场景也具备良好的自定义扩展能力，

因此，flume可以适用于大部分的日常数据采集场景

### 1.1.2 运行机制

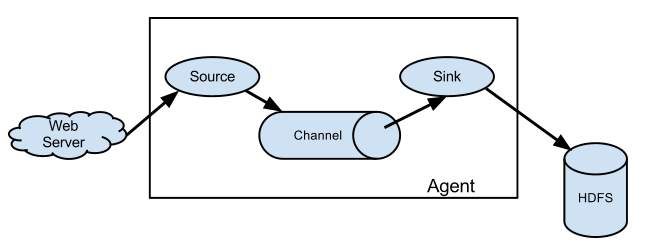
1. Flume分布式系统中最**核心的角色是agent**，flume采集系统就是由一个个agent所连接起来形成
2. **每一个agent相当于一个数据传递员，内部有三个组件：**
   * 1. Source：采集组件，用于跟数据源对接，以获取数据
     2. Sink：下沉组件，用于往下一级agent传递数据或者往最终存储系统传递数据
     3. Channel：传输通道组件，用于从source将数据传递到sink



### 1.1.4 Flume采集系统结构图

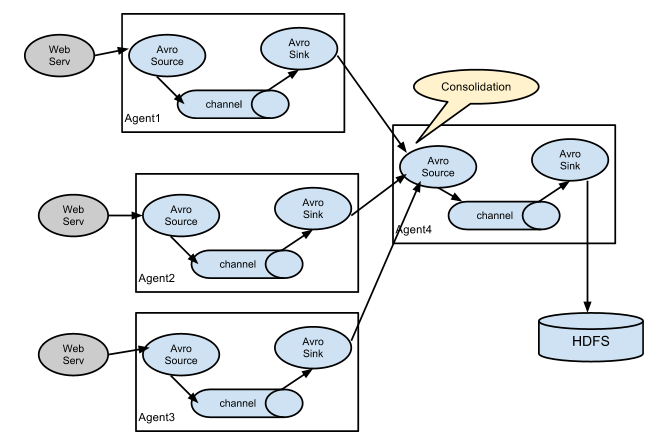
#### 1. 简单结构

单个agent采集数据



#### 2. 复杂结构

多级agent之间串联



## 1.2 Flume实战案例

#### 1.2.1 Flume的安装部署

1. Flume的安装非常简单，只需要解压即可，当然，前提是已有hadoop环境

上传安装包到数据源所在节点上

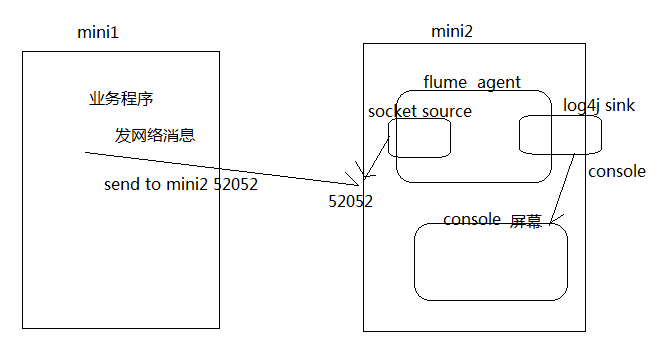
然后解压 tar -zxvf apache-flume-1.6.0-bin.tar.gz

然后进入flume的目录，修改conf下的flume-env.sh，在里面配置JAVA\_HOME

2、根据数据采集的需求**配置采集方案**，描述在配置文件中(文件名可任意自定义)

3、**指定采集方案配置文件**，在相应的节点上启动flume agent

先用一个最简单的例子来测试一下程序环境是否正常



1. 先在flume的conf目录下新建一个配置文件（采集方案）

vi netcat-logger.properties

|  |
| --- |
| # 定义这个agent中各组件的名字  a1.sources = r1  a1.sinks = k1  a1.channels = c1  # 描述和配置source组件：r1  a1.sources.r1.type = netcat  a1.sources.r1.bind = localhost  a1.sources.r1.port = 44444  # 描述和配置sink组件：k1  a1.sinks.k1.type = logger  # 描述和配置channel组件，此处使用是内存缓存的方式  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100  # 描述和配置source channel sink之间的连接关系  a1.sources.r1.channels = c1  a1.sinks.k1.channel = c1 |

1. 启动agent去采集数据

|  |
| --- |
| bin/flume-ng agent -c conf -f conf/netcat-logger.conf -n a1 -Dflume.root.logger=INFO,console |

-c conf 指定flume自身的配置文件所在目录

-f conf/netcat-logger.con 指定我们所描述的采集方案

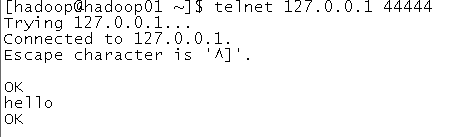
-n a1 指定我们这个agent的名字

1. 测试

先要往agent的source所监听的端口上发送数据，让agent有数据可采

随便在一个能跟agent节点联网的机器上

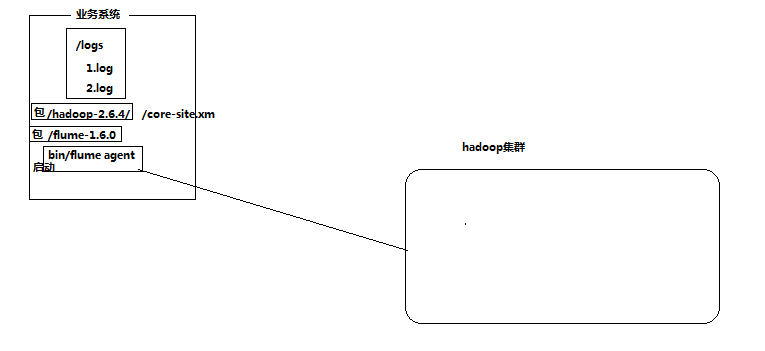
telnet anget-hostname port （telnet localhost 44444）



#### 1.2.2 采集案例

##### 1、采集日志目录中的文件到HDFS

结构示意图：



采集需求：某服务器的某特定目录下，会不断产生新的文件，每当有新文件出现，就需要把文件采集到HDFS中去

根据需求，首先定义以下3大要素

* 数据源组件，即source ——监控文件目录 : spooldir

spooldir特性：

1、监视一个目录，只要目录中出现新文件，就会采集文件中的内容

2、采集完成的文件，会被agent自动添加一个后缀：COMPLETED

3、所监视的目录中不允许重复出现相同文件名的文件

* 下沉组件，即sink——HDFS文件系统 : hdfs sink
* 通道组件，即channel——可用file channel 也可以用内存channel

配置文件编写：

|  |
| --- |
| #定义三大组件的名称  agent1.sources = source1  agent1.sinks = sink1  agent1.channels = channel1  # 配置source组件  agent1.sources.source1.type = spooldir  agent1.sources.source1.spoolDir = /home/hadoop/logs/  agent1.sources.source1.fileHeader = false  #配置拦截器  agent1.sources.source1.interceptors = i1  agent1.sources.source1.interceptors.i1.type = host  agent1.sources.source1.interceptors.i1.hostHeader = hostname  # 配置sink组件  agent1.sinks.sink1.type = hdfs  agent1.sinks.sink1.hdfs.path =hdfs://hdp-node-01:9000/weblog/flume-collection/%y-%m-%d/%H-%M  agent1.sinks.sink1.hdfs.filePrefix = access\_log  agent1.sinks.sink1.hdfs.maxOpenFiles = 5000  agent1.sinks.sink1.hdfs.batchSize= 100  agent1.sinks.sink1.hdfs.fileType = DataStream  agent1.sinks.sink1.hdfs.writeFormat =Text  agent1.sinks.sink1.hdfs.rollSize = 102400  agent1.sinks.sink1.hdfs.rollCount = 1000000  agent1.sinks.sink1.hdfs.rollInterval = 60  #agent1.sinks.sink1.hdfs.round = true  #agent1.sinks.sink1.hdfs.roundValue = 10  #agent1.sinks.sink1.hdfs.roundUnit = minute  agent1.sinks.sink1.hdfs.useLocalTimeStamp = true  # Use a channel which buffers events in memory  agent1.channels.channel1.type = memory  agent1.channels.channel1.keep-alive = 120  agent1.channels.channel1.capacity = 500000  agent1.channels.channel1.transactionCapacity = 600  # Bind the source and sink to the channel  agent1.sources.source1.channels = channel1  agent1.sinks.sink1.channel = channel1 |

Channel参数解释：

capacity：默认该通道中最大的可以存储的event数量

trasactionCapacity：每次最大可以从source中拿到或者送到sink中的event数量

keep-alive：event添加到通道中或者移出的允许时间

测试阶段，启动flume agent的命令：

|  |
| --- |
| bin/flume-ng agent -c ./conf -f ./dir-hdfs.conf -n agent1 -Dflume.root.logger=DEBUG,console |

-D后面跟的是log4j的参数，用于测试观察

生产中，启动flume，应该把flume启动在后台：

|  |
| --- |
| nohup bin/flume-ng agent -c ./conf -f ./dir-hdfs.conf -n agent1 1>/dev/null 2>&1 & |

##### 2、采集文件到HDFS

采集需求：比如业务系统使用log4j生成的日志，日志内容不断增加，需要把追加到日志文件中的数据**实时**采集到hdfs

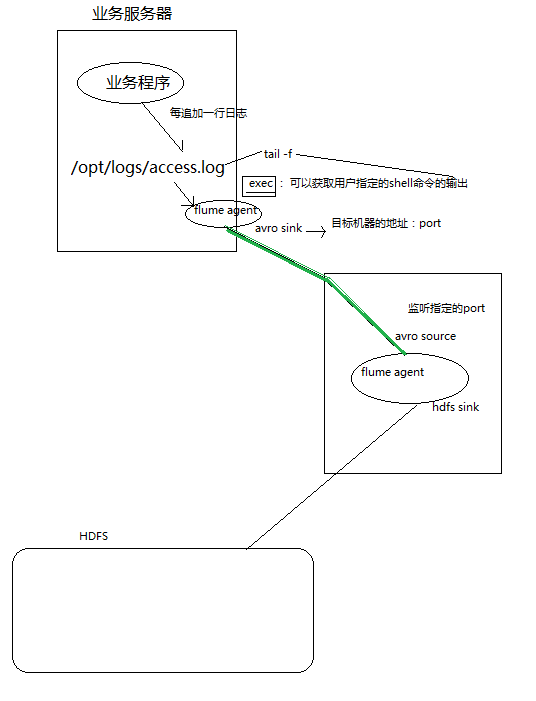
根据需求，首先定义以下3大要素

* 采集源，即source——监控文件内容更新 : exec ‘tail -F file’
* 下沉目标，即sink——HDFS文件系统 : hdfs sink
* Source和sink之间的传递通道——channel，可用file channel 也可以用 内存channel

配置文件编写：

|  |
| --- |
| agent1.sources = source1  agent1.sinks = sink1  agent1.channels = channel1  # Describe/configure tail -F source1  agent1.sources.source1.type = exec  agent1.sources.source1.command = tail -F /home/hadoop/logs/access\_log  agent1.sources.source1.channels = channel1  #configure host for source  agent1.sources.source1.interceptors = i1  agent1.sources.source1.interceptors.i1.type = host  agent1.sources.source1.interceptors.i1.hostHeader = hostname  # Describe sink1  agent1.sinks.sink1.type = hdfs  #a1.sinks.k1.channel = c1  agent1.sinks.sink1.hdfs.path =hdfs://hdp-node-01:9000/weblog/flume-collection/%y-%m-%d/%H-%M  agent1.sinks.sink1.hdfs.filePrefix = access\_log  agent1.sinks.sink1.hdfs.maxOpenFiles = 5000  agent1.sinks.sink1.hdfs.batchSize= 100  agent1.sinks.sink1.hdfs.fileType = DataStream  agent1.sinks.sink1.hdfs.writeFormat =Text  agent1.sinks.sink1.hdfs.rollSize = 102400  agent1.sinks.sink1.hdfs.rollCount = 1000000  agent1.sinks.sink1.hdfs.rollInterval = 60  agent1.sinks.sink1.hdfs.round = true  agent1.sinks.sink1.hdfs.roundValue = 10  agent1.sinks.sink1.hdfs.roundUnit = minute  agent1.sinks.sink1.hdfs.useLocalTimeStamp = true  # Use a channel which buffers events in memory  agent1.channels.channel1.type = memory  agent1.channels.channel1.keep-alive = 120  agent1.channels.channel1.capacity = 500000  agent1.channels.channel1.transactionCapacity = 600  # Bind the source and sink to the channel  agent1.sources.source1.channels = channel1  agent1.sinks.sink1.channel = channel1 |

3、两个agent级联



## 1.3 更多source和sink组件

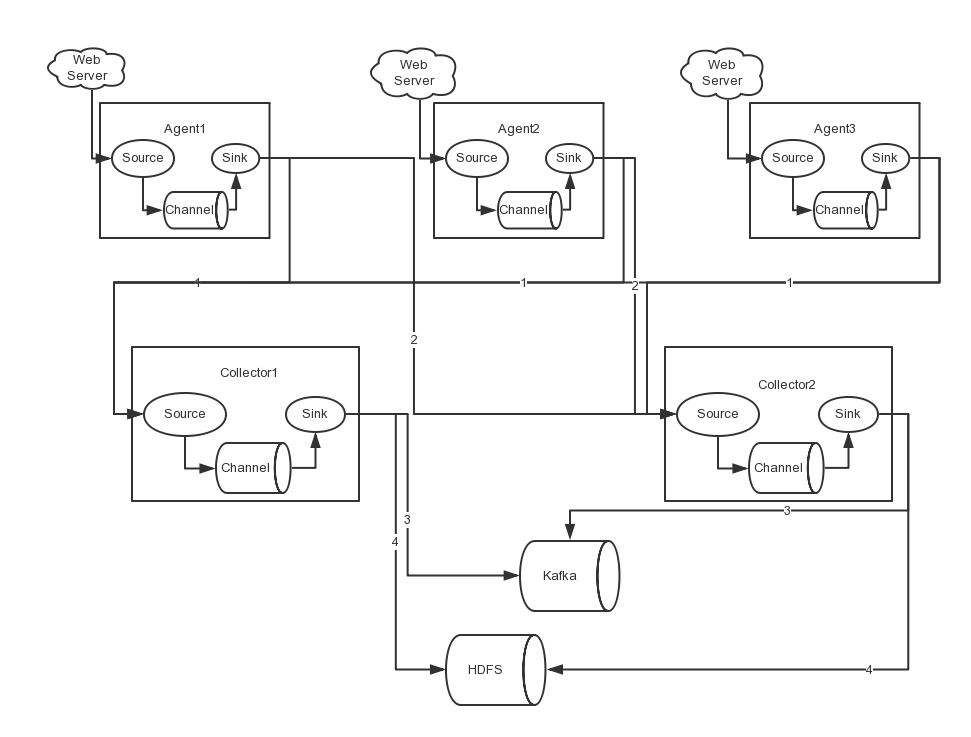
Flume支持众多的source和sink类型，详细手册可参考官方文档

<http://flume.apache.org/FlumeUserGuide.html>

# 2. 扩展：FLUME高级应用

## 1.4 案例：高可用Flum-NG配置案例

在完成单点的Flume NG搭建后，下面我们搭建一个高可用的Flume NG集群，架构图如下所示：



　　图中，我们可以看出，Flume的存储可以支持多种，这里只列举了HDFS和Kafka（如：存储最新的一周日志，并给Storm系统提供实时日志流）。

### 1.4.1 角色分配

Flume的Agent和Collector分布如下表所示：

|  |  |  |
| --- | --- | --- |
| 名称 | HOST | 角色 |
| Agent1 | mini1 | Web Server |
| Agent2 | mini2 | Web Server |
| Agent3 | mini3 | Web Server |
| Collector1 | mini4 | AgentMstr1 |
| Collector2 | mini5 | AgentMstr2 |

　　图中所示，Agent1，Agent2，Agent3数据分别流入到Collector1和Collector2，Flume NG本身提供了Failover机制，可以自动切换和恢复。在上图中，有3个产生日志服务器分布在不同的机房，要把所有的日志都收集到一个集群中存储。下 面我们开发配置Flume NG集群

### 1.4.2 配置

在下面单点Flume中，基本配置都完成了，我们只需要新添加两个配置文件，它们是agent.properties和collector.properties，其配置内容如下所示：

1、agent配置

|  |
| --- |
| [root@mini1 apache-flume-1.6.0-bin]# vi conf/agent.properties  #agent1 name  agent1.channels = c1  agent1.sources = r1  agent1.sinks = k1 k2  #set gruop  agent1.sinkgroups = g1  #set channel  agent1.channels.c1.type = memory  agent1.channels.c1.capacity = 1000  agent1.channels.c1.transactionCapacity = 100  agent1.sources.r1.channels = c1  agent1.sources.r1.type = exec  agent1.sources.r1.command = tail -F /root/log/test.log  agent1.sources.r1.interceptors = i1 i2  agent1.sources.r1.interceptors.i1.type = static  agent1.sources.r1.interceptors.i1.key = Type  agent1.sources.r1.interceptors.i1.value = LOGIN  agent1.sources.r1.interceptors.i2.type = timestamp  # set sink1  agent1.sinks.k1.channel = c1  agent1.sinks.k1.type = avro  agent1.sinks.k1.hostname = mini2  agent1.sinks.k1.port = 52020  # set sink2  agent1.sinks.k2.channel = c1  agent1.sinks.k2.type = avro  agent1.sinks.k2.hostname = mini3  agent1.sinks.k2.port = 52020  #set sink group  agent1.sinkgroups.g1.sinks = k1 k2  #set failover  agent1.sinkgroups.g1.processor.type = failover  agent1.sinkgroups.g1.processor.priority.k1 = 10  agent1.sinkgroups.g1.processor.priority.k2 = 1  agent1.sinkgroups.g1.processor.maxpenalty = 10000 |

启动命令：

|  |
| --- |
| bin/flume-ng agent -n agent1 -c conf -f conf/agent.properties -Dflume.root.logger=DEBUG,console |

2、collector配置

|  |
| --- |
| [root@mini2 conf]# vi collector.properties  #set Agent name  a1.sources = r1  a1.channels = c1  a1.sinks = k1  #set channel  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100  # other node,nna to nns  a1.sources.r1.type = avro  a1.sources.r1.bind = mini2  a1.sources.r1.port = 52020  a1.sources.r1.interceptors = i1  a1.sources.r1.interceptors.i1.type = static  a1.sources.r1.interceptors.i1.key = Collector  a1.sources.r1.interceptors.i1.value = mini2  a1.sources.r1.channels = c1  #set sink to hdfs  a1.sinks.k1.type=hdfs  a1.sinks.k1.hdfs.path=/home/hdfs/flume/logdfs  a1.sinks.k1.hdfs.fileType=DataStream  a1.sinks.k1.hdfs.writeFormat=TEXT  a1.sinks.k1.hdfs.rollInterval=10  a1.sinks.k1.channel=c1  a1.sinks.k1.hdfs.filePrefix=%Y-%m-%d |

在mini3上，需要修改上述配置中的红色字体主机名为mini3

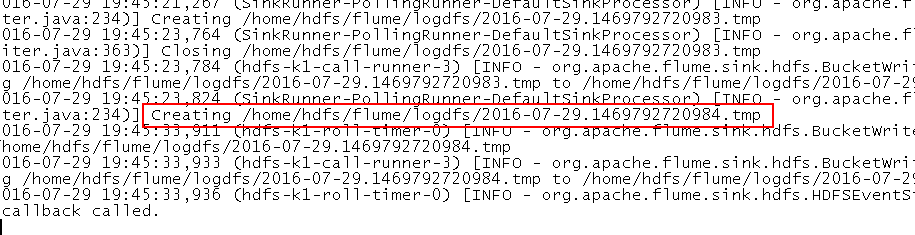
启动命令：

|  |
| --- |
| bin/flume-ng agent -n a1 -c conf -f conf/collector.properties -Dflume.root.logger=DEBUG,console |

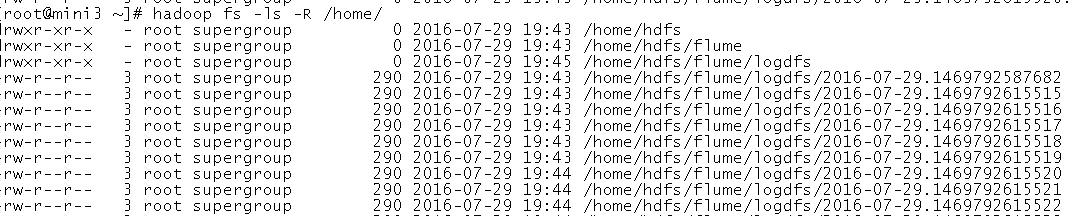
### 1.4.3 FAILOVER测试

下面我们来测试下Flume NG集群的高可用（故障转移）。场景如下：我们在Agent1节点上传文件，由于我们配置Collector1的权重比Collector2大，所以 Collector1优先采集并上传到存储系统。然后我们kill掉Collector1，此时有Collector2负责日志的采集上传工作，之后，我 们手动恢复Collector1节点的Flume服务，再次在Agent1上次文件，发现Collector1恢复优先级别的采集工作。具体截图如下所 示：

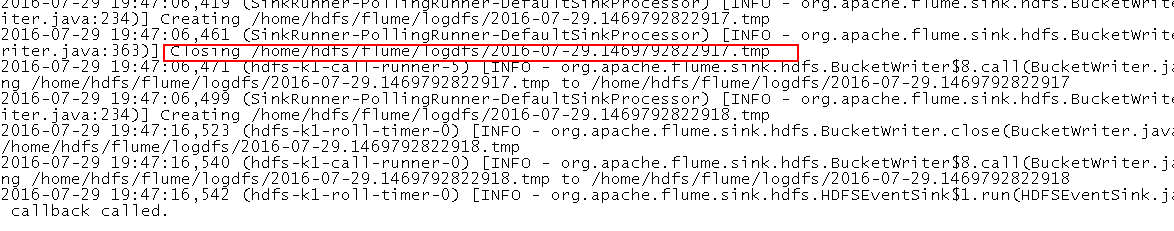
Collector1优先上传



HDFS集群中上传的log内容预览



Collector1宕机，Collector2获取优先上传权限



重启Collector1服务，Collector1重新获得优先上传的权限

## 1.5 案例：多服务器多类型文件--分类--汇聚采集

《见文档》

## 1.6 案例：自定义拦截器--字段抽取--字段加密

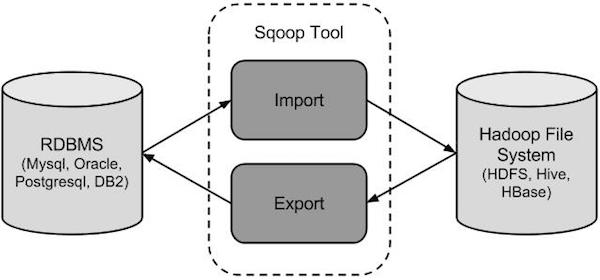
# 3. sqoop数据迁移工具

## 2.1 概述

sqoop是apache旗下一款**“Hadoop和关系数据库服务器之间传送数据”**的工具。

**导入数据**：MySQL，Oracle导入数据到Hadoop的HDFS、HIVE、HBASE等数据存储系统；

**导出数据：**从Hadoop的文件系统中导出数据到关系数据库mysql等



## 2.2 工作机制

将导入或导出命令翻译成mapreduce程序来实现

在翻译出的mapreduce中主要是对inputformat和outputformat进行定制

## 2.3 sqoop实战及原理

### 2.3.1 sqoop安装

安装sqoop的前提是已经具备java和hadoop的环境

#### 1、下载并解压

最新版下载地址http://ftp.wayne.edu/apache/sqoop/1.4.6/

#### 2、修改配置文件

$ cd $SQOOP\_HOME/conf

$ mv sqoop-env-template.sh sqoop-env.sh

打开**sqoop-env.sh**并编辑下面几行：

export HADOOP\_COMMON\_HOME=/home/hadoop/apps/hadoop-2.6.1/

export HADOOP\_MAPRED\_HOME=/home/hadoop/apps/hadoop-2.6.1/

export HIVE\_HOME=/home/hadoop/apps/hive-1.2.1

#### 加入mysql的jdbc驱动包

cp ~/app/hive/lib/mysql-connector-java-5.1.28.jar $SQOOP\_HOME/lib/

#### 4、验证启动

$ cd $SQOOP\_HOME/bin

$ sqoop-version

预期的输出：

15/12/17 14:52:32 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6

Sqoop 1.4.6 git commit id 5b34accaca7de251fc91161733f906af2eddbe83

Compiled by abe on Fri Aug 1 11:19:26 PDT 2015

到这里，整个Sqoop安装工作完成。

验证sqoop到mysql业务库之间的连通性：

bin/sqoop-list-databases --connect jdbc:mysql://localhost:3306 --username root --password root

bin/sqoop-list-tables --connect jdbc:mysql://localhost:3306/userdb --username root --password root

## 2.4 Sqoop的数据导入

“导入工具”导入单个表从RDBMS到HDFS。表中的每一行被视为HDFS的记录。所有记录都存储为文本文件的文本数据（或者Avro、sequence文件等二进制数据）

### 2.4.1 语法

下面的语法用于将数据导入HDFS。

|  |
| --- |
| $ sqoop import (generic-args) (import-args) |

### 2.4.2 示例

#### 表数据

在mysql中有一个库userdb中三个表：emp, emp\_add和emp\_conn

表emp:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **name** | **deg** | **salary** | **dept** |
| 1201 | gopal | manager | 50,000 | TP |
| 1202 | manisha | Proof reader | 50,000 | TP |
| 1203 | khalil | php dev | 30,000 | AC |
| 1204 | prasanth | php dev | 30,000 | AC |
| 1205 | kranthi | admin | 20,000 | TP |

表emp\_add:

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **hno** | **street** | **city** |
| 1201 | 288A | vgiri | jublee |
| 1202 | 108I | aoc | sec-bad |
| 1203 | 144Z | pgutta | hyd |
| 1204 | 78B | old city | sec-bad |
| 1205 | 720X | hitec | sec-bad |

表emp\_conn:

|  |  |  |
| --- | --- | --- |
| **id** | **phno** | **email** |
| 1201 | 2356742 | gopal@tp.com |
| 1202 | 1661663 | manisha@tp.com |
| 1203 | 8887776 | khalil@ac.com |
| 1204 | 9988774 | prasanth@ac.com |
| 1205 | 1231231 | kranthi@tp.com |

#### 导入表表数据到HDFS

下面的命令用于从MySQL数据库服务器中的emp表导入HDFS。

|  |
| --- |
| bin/sqoop import \  --connect jdbc:mysql://hdp-node-01:3306/test \  --username root \  --password root \  --table emp \  --m 1 |

如果成功执行，那么会得到下面的输出。

|  |
| --- |
| 14/12/22 15:24:54 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5  14/12/22 15:24:56 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-hadoop/compile/cebe706d23ebb1fd99c1f063ad51ebd7/emp.jar  -----------------------------------------------------  O mapreduce.Job: map 0% reduce 0%  14/12/22 15:28:08 INFO mapreduce.Job: map 100% reduce 0%  14/12/22 15:28:16 INFO mapreduce.Job: Job job\_1419242001831\_0001 completed successfully  -----------------------------------------------------  -----------------------------------------------------  14/12/22 15:28:17 INFO mapreduce.ImportJobBase: Transferred 145 bytes in 177.5849 seconds (0.8165 bytes/sec)  14/12/22 15:28:17 INFO mapreduce.ImportJobBase: Retrieved 5 records. |

为了验证在HDFS导入的数据，请使用以下命令查看导入的数据

|  |
| --- |
| $ $HADOOP\_HOME/bin/hadoop fs -cat /user/hadoop/emp/part-m-00000 |

emp表的数据和字段之间用逗号(,)表示。

|  |
| --- |
| 1201, gopal, manager, 50000, TP  1202, manisha, preader, 50000, TP  1203, kalil, php dev, 30000, AC  1204, prasanth, php dev, 30000, AC  1205, kranthi, admin, 20000, TP |

#### 导入到HDFS指定目录

在导入表数据到HDFS使用Sqoop导入工具，我们可以指定目标目录。

以下是指定目标目录选项的Sqoop导入命令的语法。

|  |
| --- |
| --target-dir <new or exist directory in HDFS> |

下面的命令是用来导入emp\_add表数据到'/queryresult'目录。

|  |
| --- |
| bin/sqoop import \  --connect jdbc:mysql://hdp-node-01:3306/test \  --username root \  --password root \  --target-dir /queryresult \  --fields-terminated-by ‘\001’ \  --table emp  --split-by id  --m 1 |

注意：如果报错，说emp类找不到，则可以手动从sqoop生成的编译目录(/tmp/sqoop-root/compile)中，找到这个emp.class和emp.jar，拷贝到sqoop的lib目录下：

如果设置了 --m 1，则意味着只会启动一个maptask执行数据导入

如果不设置 --m 1，则默认为启动4个map task执行数据导入，则需要指定一个列来作为划分map task任务的依据

下面的命令是用来验证 /queryresult 目录中 emp\_add表导入的数据形式。

|  |
| --- |
| $HADOOP\_HOME/bin/hadoop fs -cat /queryresult/part-m-\* |

它会用逗号（，）分隔emp\_add表的数据和字段。

|  |
| --- |
| 1201, 288A, vgiri, jublee  1202, 108I, aoc, sec-bad  1203, 144Z, pgutta, hyd  1204, 78B, oldcity, sec-bad  1205, 720C, hitech, sec-bad |

#### 导入关系表到HIVE

|  |
| --- |
| bin/sqoop import --connect jdbc:mysql://hdp-node-01:3306/test --username root --password root --table emp --hive-import --split-by id --m 1 |

#### 导入表数据子集

我们可以导入表的使用Sqoop导入工具，"where"子句的一个子集。它执行在各自的数据库服务器相应的SQL查询，并将结果存储在HDFS的目标目录。

where子句的语法如下。

|  |
| --- |
| --where <condition> |

下面的命令用来导入emp\_add表数据的子集。子集查询检索员工ID和地址，居住城市为：Secunderabad

|  |
| --- |
| bin/sqoop import \  --connect jdbc:mysql://hdp-node-01:3306/test \  --username root \  --password root \  --where "city ='sec-bad'" \  --target-dir /wherequery \  --table emp\_add \  --m 1 |

按需导入

|  |
| --- |
| bin/sqoop import \  --connect jdbc:mysql://hdp-node-01:3306/test \  --username root \  --password root \  --target-dir /wherequery2 \  --query 'select id,name,deg from emp WHERE id>1207 **and $CONDITIONS'** \  --split-by id \  --fields-terminated-by '\t' \  --m 2 |

下面的命令用来验证数据从emp\_add表导入/wherequery目录

|  |
| --- |
| $HADOOP\_HOME/bin/hadoop fs -cat /wherequery/part-m-\* |

它用逗号（，）分隔 emp\_add表数据和字段。

|  |
| --- |
| 1202, 108I, aoc, sec-bad  1204, 78B, oldcity, sec-bad  1205, 720C, hitech, sec-bad |

#### 增量导入

增量导入是仅导入新添加的表中的行的技术。

sqoop支持两种增量MySql导入到hive的模式，

一种是append，即通过指定一个递增的列，比如：

--incremental append --check-column num\_id --last-value 0

另种是可以根据时间戳，比如：

--incremental lastmodified --check-column created --last-value '2012-02-01 11:0:00'

就是只导入created 比'2012-02-01 11:0:00'更大的数据。

**1/ append模式**

它需要添加‘incremental’, ‘check-column’, 和 ‘last-value’选项来执行增量导入。

下面的语法用于Sqoop导入命令增量选项。

|  |
| --- |
| --incremental <mode>  --check-column <column name>  --last value <last check column value> |

假设新添加的数据转换成emp表如下：

1206, satish p, grp des, 20000, GR

下面的命令用于在EMP表执行增量导入。

|  |
| --- |
| bin/sqoop import \  --connect jdbc:mysql://hdp-node-01:3306/test \  --username root \  --password root \  --table emp --m 1 \  --incremental append \  --check-column id \  --last-value 1208 |

以下命令用于从emp表导入HDFS emp/ 目录的数据验证。

|  |
| --- |
| $ $HADOOP\_HOME/bin/hadoop fs -cat /user/hadoop/emp/part-m-\*  1201, gopal, manager, 50000, TP  1202, manisha, preader, 50000, TP  1203, kalil, php dev, 30000, AC  1204, prasanth, php dev, 30000, AC  1205, kranthi, admin, 20000, TP  1206, satish p, grp des, 20000, GR |

下面的命令是从表emp 用来查看修改或新添加的行

|  |
| --- |
| $HADOOP\_HOME/bin/hadoop fs -cat /emp/part-m-\*1  1206, satish p, grp des, 20000, GR |

## 2.5 Sqoop的数据导出

**1/ 将数据从HDFS把文件导出到RDBMS数据库**

导出前，目标表必须存在于目标数据库中。

* 默认操作是从将文件中的数据使用INSERT语句插入到表中
* 更新模式下，是生成UPDATE语句更新表数据

### 语法

以下是export命令语法。

|  |
| --- |
| $ sqoop export (generic-args) (export-args) |

### 示例

数据是在HDFS 中“EMP/”目录的emp\_data文件中。所述emp\_data如下：

|  |
| --- |
| 1201, gopal, manager, 50000, TP  1202, manisha, preader, 50000, TP  1203, kalil, php dev, 30000, AC  1204, prasanth, php dev, 30000, AC  1205, kranthi, admin, 20000, TP  1206, satish p, grp des, 20000, GR |

1、首先需要手动创建mysql中的目标表

|  |
| --- |
| $ mysql  mysql> USE db;  mysql> CREATE TABLE employee (  id INT NOT NULL PRIMARY KEY,  name VARCHAR(20),  deg VARCHAR(20),  salary INT,  dept VARCHAR(10)); |

2、然后执行导出命令

|  |
| --- |
| bin/sqoop export \  --connect jdbc:mysql://hdp-node-01:3306/test \  --username root \  --password root \  --table employee \  --export-dir /user/hadoop/emp/ |

3、验证表mysql命令行。

|  |
| --- |
| mysql>select \* from employee;  如果给定的数据存储成功，那么可以找到数据在如下的employee表。  +------+--------------+-------------+-------------------+--------+  | Id | Name | Designation | Salary | Dept |  +------+--------------+-------------+-------------------+--------+  | 1201 | gopal | manager | 50000 | TP |  | 1202 | manisha | preader | 50000 | TP |  | 1203 | kalil | php dev | 30000 | AC |  | 1204 | prasanth | php dev | 30000 | AC |  | 1205 | kranthi | admin | 20000 | TP |  | 1206 | satish p | grp des | 20000 | GR |  +------+--------------+-------------+-------------------+--------+ |