Flume和Morphlines实现数据的实时ETL

本文来自徐宇辉（微信号：xuyuhui263）的投稿，目前在中国移动从事数字营销的业务支撑工作，感谢他的文章。

Apache Flume简介

Apache Flume是一个Apache的开源项目，是一个分布的、可靠的软件系统，主要目的是从大量的分散的数据源中收集、汇聚以及迁移大规模的日志数据，最后存储到一个集中式的数据系统中。

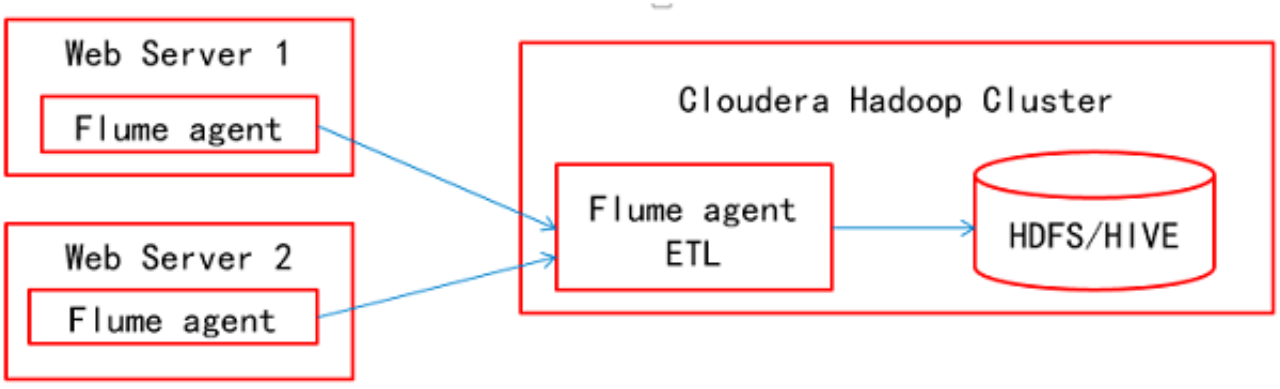
Apache Flume是由运行在不同主机系统的软件进程组成，一个主机的软件进程叫agent, 1个agent由source、channel、以及sink组成：Source负责盯住主机上数据源，如各种Weblog以及Syslog等等；Channel负责使用内存存储传输数据；Sink负责将数据放在在这个主机上最终的目的。

由于两台主机的Agent之间可以进行串联，1个agent的sink可以对接另1个agent的source；当然，1个agent可以将source的数据同时分发给两个并联的Channel,
同时sink到两个不同的数据目的地。 Flume中agent的组合方式非常灵活多样，在此不做过多描述，有兴趣可以去http://flume.apache.org/详细了解。

项目方案

项目平台：在笔者的系统中，有若干台Web Server，每台Web Server安装Nginx作为HTTP服务。Nginx自带host.access.log文件作为日志文件，该日志文件以行方式存储Web访问记录，每1行代表1次HTTP请求，记录下请求的方法，URL，以及回应码等等。

项目目的：由于请求的URL的参数格式不统一，笔者希望针对特定参数进行提取操作,并且对一些不规范的参数进行整形；同时希望过滤掉除去200 OK以及302 Redirection之外其他所有回应码的请求（如499,500）；最后，经过ETL之后用户访问记录，以Avro形式存储在HDFS中，形成HIVE表。在以上所有操作中，通过Flume的各个agent进行数据收集合并，并在agent的进程内存中进行ETL，中间过程不经过硬盘以及文件操作，以达到实时快速的目的。



如果想

及时了解Spar
k、Hadoop、Flink或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

项目部署方案：在所有Web Server上安装Web Flume
agent，只是负责收集本主机Weblog并且发送给下游的Hadoop Flume
agent。在Hadoop中选一台主机安装Flume agent，所有Weblog汇聚到Hadoop中的这个agent当中，被写入到HDFS文件当中，所有ETL操作都由这个agent完成。

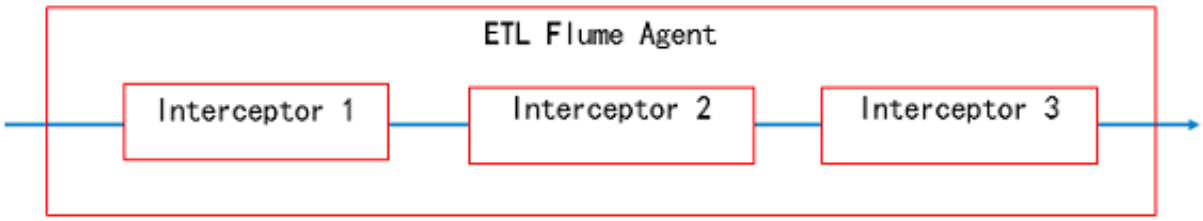
技术实现

Flume的功能如果只是收集以及汇聚log数据的话，配置可以非常简单。如果要实现ETL，就要复杂的配置，需要利用Flume给我们提供的一种叫interceptor的功能。我们可以想象interceptor是一个Flume的插件，每一个插件进行一种操作，前一个interceptor处理后的结果会被送到下一个interceptor。

Flume的interceptor主要有三种：

RegexFilter：顾名思义，这种Interceptor根据正则表达式的标准过滤或者放通特定的Weblog;
RegexReplace:
这种Interceptor根据正则表达式替代Weblog中的特定字符串，类似于正则group处理；
Morphlines: 最为强大的interceptor，可以将行格式记录转化成Avro记录，可以针对记录中的字段摘取特定的内容而放弃无用的内容，可以进行时间戳的转化，也可以针对记录字段进行查找更换。

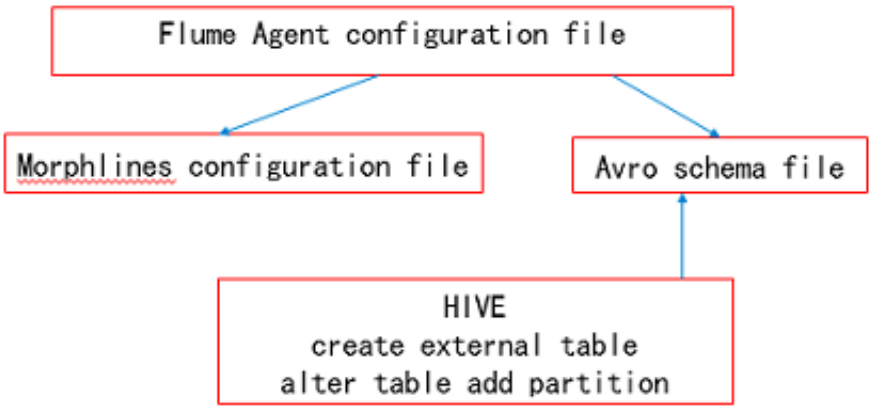
Morphlines interceptor的细节功能可以参照以下链接：
http://cloudera.github.io/cdk/docs/current/cdk-morphlines/morphlinesReferenceGuide.html



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

利用Flume的interceptor来实现ETL，工作就是编写几种配置文件：

- Flume agent的配置文件，这个文件定义agent的source、channel以及sink以及要实现的interceptor;
- 如果要使用Morphlines interceptor，就要定义一个Morphlines的配置文件；
- 因为Morphlines会将行数据转化成Avro数据格式的文件，所以要定义一个Avro schema；
- Avro文件存在HDFS中，如果要使用HIVE进行访问，就要进行表创建，同样要用到Avro schema。



iteblog_hadoop

以下是一个Nginx的HTTP请求的典型行记录，我们通过一些配置文件的片段来深入了解一下Flume ETL的工作机制。

117.136.41.65 - - [05/Mar/2017:16:28:20 +0800] "GET /?usrid=8613715935023&oriurl=http%3A%2F%2Fwww.7710086.com%2F HTTP/1.1" 200 520 "-" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36" "-" "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8" "text/hmtl; charset=UTF-8" "-" "-" "-" "861371593xxxx"

iteblog.sources.source1.interceptors=attach-schema
iteblog.sources.source1.interceptors.attach-schema.typr=static
iteblog.sources.source1.interceptors.attach-schema.key=flume.avro.schema.url
iteblog.sources.source1.interceptors.attach-schema.value=file:/etc/flume-ng/conf/schemas/adss.avsc
iteblog.sources.source1.interceptors.morphline.type=org.apache.flume.sink.solr.morphline.MorphlineInterceptor$Builder
iteblog.sources.source1.interceptors.morphline.morphlineFile=/etc/flume-ng/conf/morphline.conf
iteblog.sources.source1.interceptors.morphline.morphlineId=convertAdsStreamLogsToAvro
iteblog.sources.source1.interceptors.filterInter.type=regex_filter
iteblog.sources.source1.interceptors.filterInter.regex=^(₩₩d{1,3}.₩₩d{1,3}.₩₩d{1,3}.₩₩d{1,3})₩₩s-₩₩s-₩₩s₩₩[([^"₩₩n]*)₩₩]₩₩s"(GET|HEAD)₩₩s(/₩₩?usrid=|/₩₩?isdn=|/₩₩?oriurl=)([^"₩₩n]*)"₩₩s(200|302)₩₩s(₩₩d+)₩₩s"([^"₩₩n"]*)"₩₩s"([^"₩₩n"]*)"
iteblog.sources.source1.interceptors.filterInter.excludeEvents=false

上面是Flume agent的配置文件片段，可以看到定义了2个interceptor，正则过滤RegexFilter在前，Morphlines在后。通过正则表达式可以看出，RegexFilter把非302与200的请求行格式记录过滤掉，并且也同时过滤掉了非GET以及HEAD方法的请求行记录。Morphlines interceptor定义了Morphlines配置文件的位置。

iteblog_hadoop

上面是Morphline的配置文件片段，可以看到定义了3个命令：

- readCSV读取了行记录，使用tab作为字段分隔符；
- “grok”命令通过正则表达式匹配的方式，把GET /?usrid=8613715935023&oriurl=http%3A%2F%2Fwww.7710086.com%2F HTTP/1.1 中的http%3A%2F%2Fwww.7710086.com%2F抓了出来；
- “findReplace”命令把“http%3A%2F%2Fwww.7710086.com%2F”中的%2F替换成/。

```
{
 "namespace": "company.schema",
 "type": "record",
 "name": "adsstream",
 "fields": [
  {"name": "client_ip", "type": ["null", "string"]},
  {"name": "new_uri", "type": ["null", "string"]},
  {"name": "new_time", "type": ["null", "string"]},
  {"name": "new_date", "type": ["null", "string"]},
  {"name": "response_status", "type": ["null", "long"]},
  {"name": "request_refer", "type": ["null", "string"]},
  {"name": "user_agent", "type": ["null", "string"]},
  {"name": "adv_id", "type": ["null", "long"]},
  {"name": "redirect_cause", "type": ["null", "string"]},
  {"name": "request_type", "type": ["null", "string"]},
  {"name": "response_length", "type": ["null", "long"]},
  {"name": "time_cause", "type": ["null", "long"]},
  {"name": "msisdn_num", "type": ["null", "long"]}
 ]
}
```


上面是Avro schema的配置片段，其定义较为直观，针对不同字段定义不同数据类型。

```
CREATE EXTERNAL TABLE web00 PARTITIONED BY(day STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
WITH SERDEPROPERTIES  ('avro.schema.url'='/user/flume/webads/config/adsstream.avsc')
STORED as INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT  'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
LOCATION '/user/flume/web/raw/hive00/';

ALTER TABLE web00 add partition (day='20170303') location '/user/flume/web/raw/hive00/20170303';
```


上面是HIVE的命令类型，第一个创建一个基于Avro的外部表，第二个添加一个基于HDFS路径的partition。

总结

提起大数据的实时流处理，我们首先会想到复杂的Storm₩Spark₩Kafka等等。但是，如果只是针对单条记录进行简单的ETL运算，使用Flume+Morphlines不失为一种优雅以及简约的方法。