

zxcvg的专栏

目录视图 摘要视图 RSS 订阅

个人资料



zxcvg

访问： 130124次
积分： 1420
等级： 4
排名： 第18027名

原创： 30篇
转载： 29篇
译文： 0篇
评论： 31条

文章搜索

文章分类

- Eclipse (0)
- struts+hibernate+spring (6)
- swt/Jface (1)
- EJB (3)
- 设计模式 (2)
- Oracle (4)
- SVN (2)
- java (22)
- C (0)
- 敏捷 (3)
- db (3)
- PHP (2)
- clojure (1)
- storm (2)
- kafka (2)
- flume (1)
- linux (1)
- arduino (0)
- arduino (1)

文章存档

2015年11月 (1)

Unity3D中基于订阅者模式实现事件机制 云计算行业圆桌论坛 【征文】Hadoop十周年特别策划——我与Hadoop不得不说的故事

Flume-ng+Kafka+storm的学习笔记

2014-01-21 11:33 41714人阅读 评论(4) 收藏 举报

分类： storm (1) kafka (1) flume

版权声明：本文为博主原创文章，未经博主允许不得转载。

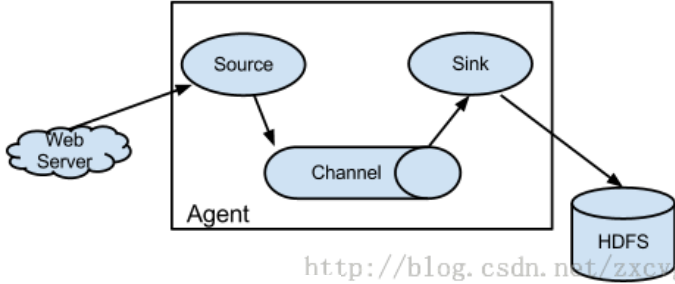
吐个槽：在word文档中写好的 包括图片 在csdn粘贴过来后 图片必须重新上传 不爽啊！！

Flume-ng+Kafka+storm的学习笔记

Flume-ng

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的文档可以看<http://flume.apache.org/FlumeUserGuide.html> 官方的英文文档 介绍的比较全面。
不过这里写写自己的见解



这个是flume的架构图

从上图可以看到几个名

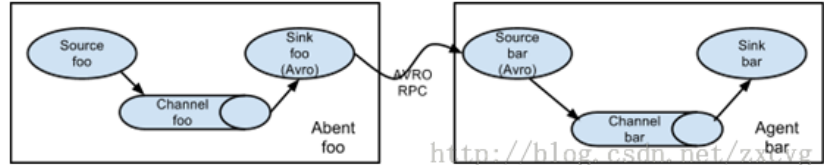
Agent: 一个Agent包含Source、Channel、Sink和其他的组件。Flume就是一个或多个Agent构成的。

Source: 数据源。简单的说就是agent获取数据的入口。

Channel: 管道。数据流通和存储的通道。一个source必须至少和一个channel关联。

Sink: 用来接收channel传输的数据并将之传送到指定的地方。传送成功后数据从channel中删除。

Flume具有高可扩展性 可随意组合:



注意 source是接收源 sink是发送源

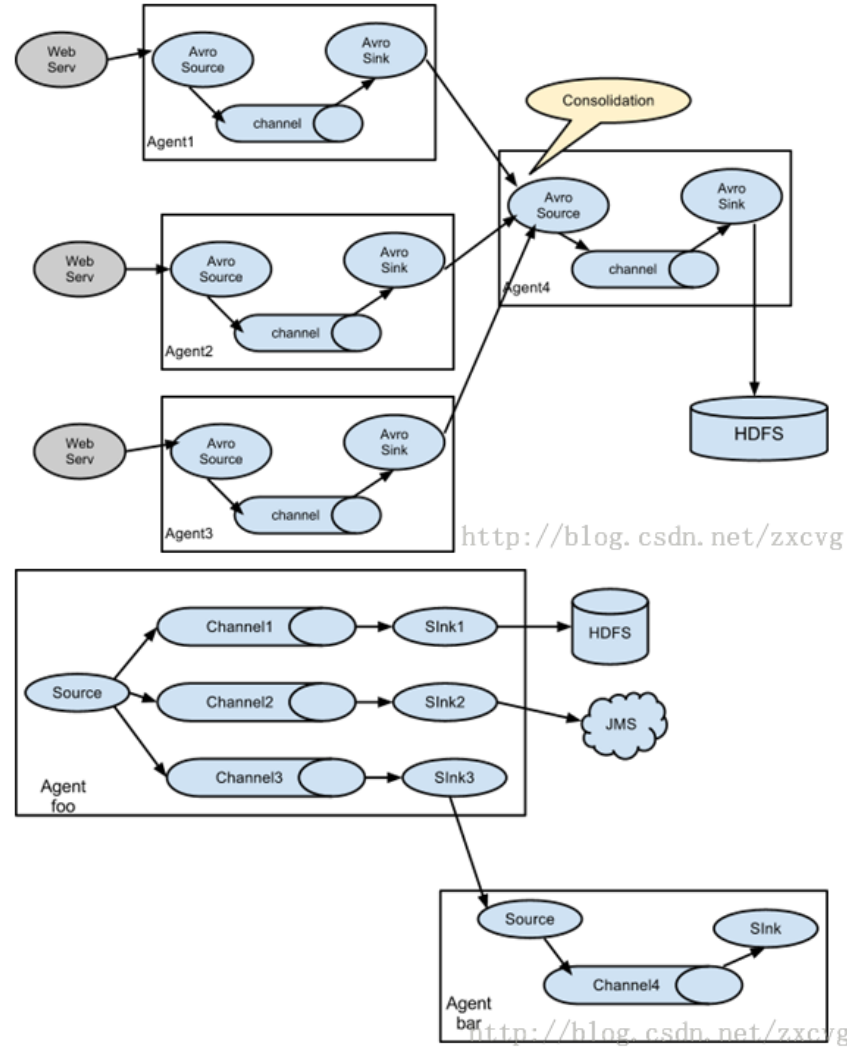
2015年10月 (1)
2015年08月 (1)
2014年07月 (1)
2014年03月 (2)
展开

阅读排行
Flume-ng+Kafka+storm的
order by用法 (41661)
kafka Failed to send mes
(10570)
Java软件低级错误 (十 五) (3460)
新浪微博图床架构解析 (3054)
Java软件低级错误 (十 八) (2732)
openfire性能 (2683)
Java软件低级错误 (五) (1987)
Java软件低级错误 (六) (1864)
SpringMVC (1714)

评论排行
kafka Failed to send mes (21)
order by用法 (4)
Flume-ng+Kafka+storm的 (4)
storm 例子 记录 (2)
SpringMVC (1)
oracle前10条记录取法(rc (0)
EJB-----会话Bean (0)
EJB基础--JBoss服务器 (0)
arduino开发板在MAC OS (0)
重新梳理三大框架-----str (0)

推荐文章
* HDFS如何检测并删除多余副本块
* Project Perfect让Swift在服务器端跑起来—让Perfect更Rails (五)
* 数据库性能优化之SQL语句优化
* Animation动画详解(七)——ObjectAnimator基本使用
* 机器学习系列(7)_机器学习路线图 (附资料)
* 一个程序员的Java和C, C++学习之路(整理)

最新评论
Flume-ng+Kafka+storm的学习笔记 zengqingchun: JedisUtils 这个类是自己写的吧 查api没有这个类
kafka Failed to send messages 民工小勇: 既然有两种方式都可以搞定，那到底哪种方式是推荐的呢？
Flume-ng+Kafka+storm的学习笔记 u010409394: 前辈，本文中的“Worker: 处理逻辑的进程，在其中运行着多个Task，每个task是一组spou...
order by用法 小颜Kevin: 谢谢 初学者，帮了我理解
kafka Failed to send messages 民工小勇: 改过后在kafka服务器本地运行producer的话./kafka-console-producer...
kafka Failed to send messages 民工小勇: 改过后在kafka服务器本地运行producer的话./kafka-console-producer...



上图是一个source将数据发给3个channel 其中的sink2将数据发给JMS， sink3将数据发给另一个source。总的来说flume的扩展性非常高 根据需要可随意组合。

现在在说说一个概念叫Event:

Event是flume的数据传输的基本单元。Flume本质上是将数据作为一个event从源头传到结尾。是由可选的Headers和载有数据的一个byte array构成。

代码结构:

```
[java] view plain copy print ?
01. /**
02.  * Basic representation of a data object inFlume.
03.  * Provides access to data as it flows throughthe system.
04.  */
05. public interface Event{
06.  /**
07.  * Returns a map of name-valuepairs describing the data stored in the body.
08.  */
09.  public Map<String, String> getHeaders();
10.  /**
11.  * Set the event headers
12.  * @param headersMap of headers to replace the current headers.
13.  */
14.  public void setHeaders(Map<String, String> headers);
15.  /**
16.  * Returns the raw byte array of the datacontained in this event.
17.  */
18.  public byte[] getBody();
19.  /**
20.  * Sets the raw byte array of the datacontained in this event.
21.  * @param body Thedata.
22.  */
23.  public void setBody(byte[] body);
24. }
```

tyzl1988: @poorzerg:成功了，感谢！改成host.name = 192.18.0.69192.18.0...

kafka Failed to send messages a sihuanyiyu: 我也遇到了一样的问题。最后在运行producer 的机器中的hosts文件下加入kafka serv...

kafka Failed to send messages a sihuanyiyu: 我也遇到了一样的问题。最后在运行producer 的机器中的hosts文件下加入kafka serv...

kafka Failed to send messages a poorzerg: 在kafka服务端的配置文件里，有一个host.name的配置，需要把值配置成ip。服务端是会把ho...

order by用法
snoopyy: 茅塞顿开

这个是网上找的flume channel ， source， sink的汇总

链接是<http://abloz.com/2013/02/26/flume-channel-source-sink-summary.html>

Component	Type	Description
Channel	memory	In-memory, fast, non-c
Channel	file	A channel for reading, manipulating a file
Channel	jdbc	JDBC-based, durable based)
Channel	recoverablememory	A durable channel imp the local file system fo
Channel	org.apache.flume.channel.PseudoTxnMemoryChannel	Mainly for testing purp production use.
Channel	(custom type as FQCN)	Your own Channel imp
Source	avro	Avro Netty RPC event
Source	exec	Execute a long-lived L from stdout
Source	netcat	Netcat style TCP even
Source	seq	Monotonically increme generator event sourc
Source	org.apache.flume.source.StressSource	Mainly for testing purp production use. Serve: source of events wher same payload. The pa number of bytes (spec defaults to 500) where signed value Byte.MA: 127).
Source	syslogtcp	SyslogTcpSource
Source	syslogudp	SyslogUDPSource
Source	org.apache.flume.source.avroLegacy.AvroLegacySource	AvroLegacySource
Source	org.apache.flume.source.thriftLegacy.ThriftLegacySource	ThriftLegacySource
Source	org.apache.flume.source.scribe.ScribeSource	ScribeSource
Source	(custom type as FQCN)	Your own Source impl
Sink	hdfs	Writes all events recei support for rolling, buc append, and more)
Sink	org.apache.flume.sink.hbase.HBaseSink	A simple sink that reac and writes them to HB
Sink	org.apache.flume.sink.hbase.AsyncHBaseSink	org.apache.flume.sink
Sink	logger	Log events at INFO le logging subsystem (lo
Sink	avro	Sink that invokes a pr method for all events i with an avro source, fr
Sink	file_roll	RollingFileSink
Sink	irc	IRCSink
Sink	null	/dev/null for Flume – b received
Sink	(custom type as FQCN)	Your own Sink impl.
ChannelSelector	replicating	ReplicatingChannelSe
ChannelSelector	multiplexing	MultiplexingChannelSi
ChannelSelector	(custom type)	Your own ChannelSel

SinkProcessor	default	DefaultSinkProcessor
SinkProcessor	failover	FailoverSinkProcessor
SinkProcessor	load_balance	Provides the ability to multiple sinks.
SinkProcessor	(custom type as FQCN)	Your own SinkProcessor
Interceptor\$Builder	host	HostInterceptor\$Builder
Interceptor\$Builder	timestamp	TimestampInterceptor
Interceptor\$Builder	static	StaticInterceptor\$Builder
Interceptor\$Builder	regex_filter	RegexFilteringInterceptor
Interceptor\$Builder	(custom type as FQCN)	Your own Interceptor\$Builder
EventSerializer\$Builder	text	BodyTextEventSerializer
EventSerializer\$Builder	avro_event	FlumeEventAvroEventSerializer
EventSerializer	org.apache.flume.sink.hbase.SimpleHbaseEventSerializer	SimpleHbaseEventSerializer
EventSerializer	org.apache.flume.sink.hbase.SimpleAsyncHbaseEventSerializer	SimpleAsyncHbaseEventSerializer
EventSerializer	org.apache.flume.sink.hbase.RegexHbaseEventSerializer	RegexHbaseEventSerializer
HbaseEventSerializer	Custom implementation of serializer for HBaseSink.	Your own HbaseEventSerializer
AsyncHbaseEventSerializer	Custom implementation of serializer for AsyncHbase sink.	Your own AsyncHbaseEventSerializer
EventSerializer\$Builder	Custom implementation of serializer for all sinks except for HBaseSink and AsyncHBaseSink.	Your own EventSerializer
	(custom type as FQCN)	

下面介绍下kafka以及kafka和flume的整合

Kafka:

从这个链接抄了些内容下来<http://dongxicheng.org/search-engine/kafka/>

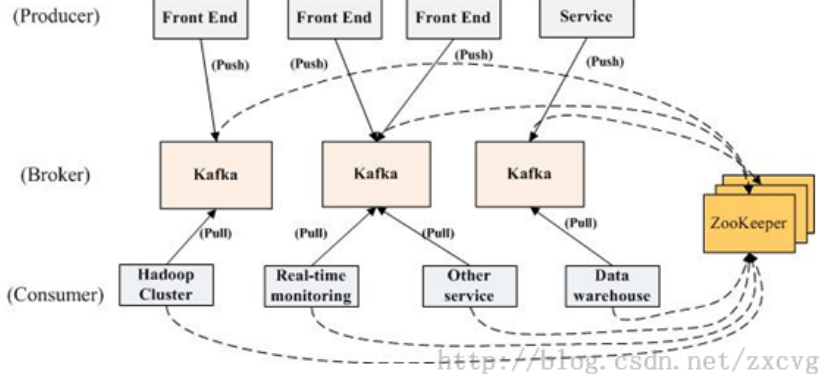
Kafka是Linkedin于2010年12月份开源的消息系统，它主要用于处理活跃的流式数据。活跃的流式数据在web网站应用中非常常见，这些数据包括网站的pv、用户访问了什么内容，搜索了什么内容等。这些数据通常以日志的形式记录下来，然后每隔一段时间进行一次统计处理。

传统的日志分析系统提供了一种离线处理日志信息的可扩展方案，但若要进行实时处理，通常会有较大延迟。而现有的消（队）列系统能够很好的处理实时或者近似实时的应用，但未处理的数据通常不会写到磁盘上，这对于Hadoop之类（一小时或者一天只处理一部分数据）的离线应用而言，可能存在问题。Kafka正是为了解决以上问题而设计的，它能够很好地离线和在线应用。

2、 设计目标

- （1）数据在磁盘上存取代价为O(1)。一般数据在磁盘上是使用BTree存储的，存取代价为O(lgn)。
- （2）高吞吐率。即使在普通的节点上每秒钟也能处理成百上千的message。
- （3）显式分布式，即所有的producer、broker和consumer都会有多个，均为分布式的。
- （4）支持数据并行加载到Hadoop中。

3、 KafKa部署结构



kafka是显式分布式架构，producer、broker（Kafka）和consumer都可以有多个。Kafka的作用类似于缓存，即活跃的数据和离线处理系统之间的缓存。几个基本概念：

- （1）message（消息）是通信的基本单位，每个producer可以向一个topic（主题）发布一些消息。如果consumer订阅了这个主题，那么新发布的消息就会广播给这些consumer。

(2) Kafka是显式分布式的，多个producer、consumer和broker可以运行在一个大的集群上，作为一个逻辑整体对外提供服务。对于consumer，多个consumer可以组成一个group，这个message只能传输给某个group中的某一个consumer。

数据从producer推送到broker，接着consumer在从broker上拉取数据。Zookeeper是一个分布式服务框架 用来解决分布式应用中的数据管理问题等。

在kafka中 有几个重要概念producer生产者 consumer 消费者 topic 主题。

我们来实际开发一个简单的生产者消费者的例子。

生产者：

```
[java] view plain copy print ?
01. public class ProducerTest {
02.
03.     public static void main(String[] args) {
04.         Properties props = new Properties();
05.         props.setProperty("metadata.broker.list", "xx.xx.xx.xx:9092");
06.         props.setProperty("serializer.class", "kafka.serializer.StringEncoder");
07.         props.put("request.required.acks", "1");
08.         ProducerConfig config = new ProducerConfig(props);
09.         Producer<String, String> producer = new Producer<String, String>(config);
10.         KeyedMessage<String, String> data = new KeyedMessage<String, String>
("kafka", "test-kafka");
11.         try {
12.             producer.send(data);
13.         } catch (Exception e) {
14.             e.printStackTrace();
15.         }
16.         producer.close();
17.     }
18. }
```

上面的代码中的xx.xx.xx.xx是kafka server的地址。

上面代码的意思就是向主题 kafka中同步（不配置的话 默认是同步发射）发送了一个信息 是test-kafka。

下面来看看消费者：

```
[java] view plain copy print ?
01. public class ConsumerTest extends Thread {
02.     private final ConsumerConnector consumer;
03.     private final String topic;
04.
05.     public static void main(String[] args) {
06.         ConsumerTest consumerThread = new ConsumerTest("kafka");
07.         consumerThread.start();
08.     }
09.     public ConsumerTest(String topic) {
10.         consumer = kafka.consumer.Consumer
11.             .createJavaConsumerConnector(createConsumerConfig());
12.         this.topic = topic;
13.     }
14.
15.     private static ConsumerConfig createConsumerConfig() {
16.         Properties props = new Properties();
17.         props.put("zookeeper.connect", "xx.xx.xx.xx:2181");
18.         props.put("group.id", "0");
19.         props.put("zookeeper.session.timeout.ms", "10000");
20.         // props.put("zookeeper.sync.time.ms", "200");
21.         // props.put("auto.commit.interval.ms", "1000");
22.
23.         return new ConsumerConfig(props);
24.     }
25.
26.
27.     public void run(){
28.
29.         Map<String, Integer> topicMap = new HashMap<String, Integer>();
30.         topicMap.put(topic, 1);
31.         Map<String, List<KafkaStream<byte[], byte[]>>> streamMap = consumer.createMessageStream
32.             (KafkaStream<byte[], byte[]> stream = streamMap.get(topic).get(0);
```

```

33.         ConsumerIterator<byte[],byte[]> it =stream.iterator();
34.         System.out.println("-----");
35.         while(it.hasNext()){
36.             //
37.             System.out.println("(consumer)--> " +new String(it.next().message()));
38.         }
39.     }
40. }
41. }

```

上面的代码就是负责接收生产者发送过来的消息 测试的时候先开启消费者 然后再运行生产者即可看到效果。

接下来 我们将flume 和kafka进行整合:

在flume的source数据源接收到数据后 通过管道 到达sink, 我们需要写一个kafkaSink 来将sink从channel接收的数据作为kafka的生产者 将数据 发送给消费者。

具体代码:

```

[java] view plain copy print ?

01. public class KafkaSink extends AbstractSinkimplementsConfigurable {
02.
03.     private static final Log logger = LogFactory.getLog(KafkaSink.class);
04.
05.     private Stringtopic;
06.     private Producer<String, String>producer;
07.
08.
09.     @Override
10.     public Status process()throwsEventDeliveryException {
11.
12.         Channel channel =getChannel();
13.         Transaction tx =channel.getTransaction();
14.         try {
15.             tx.begin();
16.             Event e = channel.take();
17.             if(e ==null) {
18.                 tx.rollback();
19.                 return Status.BACKOFF;
20.             }
21.             KeyedMessage<String,String> data = new KeyedMessage<String, String>
(topic,newString(e.getBody()));
22.             producer.send(data);
23.             logger.info("Message: {}"+new String( e.getBody()));
24.             tx.commit();
25.             return Status.READY;
26.         } catch(Exceptione) {
27.             logger.error("KafkaSinkException:{})",e);
28.             tx.rollback();
29.             return Status.BACKOFF;
30.         } finally {
31.             tx.close();
32.         }
33.     }
34.
35.     @Override
36.     public void configure(Context context) {
37.         topic = "kafka";
38.         Properties props = newProperties();
39.         props.setProperty("metadata.broker.list","xx.xx.xx.xx:9092");
40.         props.setProperty("serializer.class","kafka.serializer.StringEncoder");
41.         // props.setProperty("producer.type", "async");
42.         // props.setProperty("batch.num.messages", "1");
43.         props.put("request.required.acks","1");
44.         ProducerConfigconfig = new ProducerConfig(props);
45.         producer = newProducer<String, String>(config);
46.     }
47. }
48.

```

将此文件打成jar包 传到flume的lib下面 如果你也用的是maven的话 需要用到assembly 将依赖的jar包一起打包进去。

在flume的配置是如下:

```
[plain] view plain copy print ?
01.     agent1.sources = source1
02.     agent1.sinks = sink1
03.     agent1.channels =channel1
04.
05.     # Describe/configuresource1
06.     agent1.sources.source1.type= avro
07.     agent1.sources.source1.bind= localhost
08.     agent1.sources.source1.port= 44444
09.     # Describe sink1
10.     agent1.sinks.sink1.type= xx.xx.xx.KafkaSink(这是类的路径地址)
11.
12.     # Use a channel whichbuffers events in memory
13.     agent1.channels.channel1.type= memory
14.     agent1.channels.channel1.capacity= 1000
15.     agent1.channels.channel1.transactionCapactiy= 100
16.
17.     # Bind the source andsink to the channel
18.     agent1.sources.source1.channels= channel1
19.     agent1.sinks.sink1.channel= channel1
```

测试的话是avro的方式传送数据的 可以这样测试

bin/flume-ng avro-client--conf conf -H localhost -p 44444 -F/data/flumetmp/a
/data/flumetmp/a 这个为文件的地址.

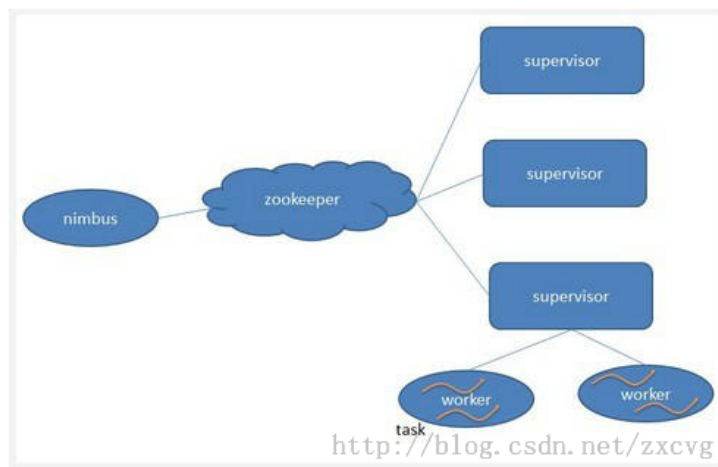
测试的时候在本地 一定要把上面写的消费者程序打开 以便接收数据测试是否成功。

接下来我们介绍下storm然后将kafka的消费者和storm进行整合：

Storm:

Storm是一个分布式的实时消息处理系统。

Storm各个组件之间的关系：



Storm集群主要由一个主节点和一群工作节点（worker node）组成，通过 Zookeeper进行协调。

主节点：主节点通常运行一个后台程序 —— Nimbus，用于响应分布在集群中的节点，分配任务和监测故障。

工作节点： Supervisor,负责接受nimbus分配的任务，启动和停止属于自己管理的worker进程。Nimbus和Supervisor之间的协调由zookeeper完成。

Worker：处理逻辑的进程，在其中运行着多个Task，每个task 是一组spout/blots的组合。

Topology:是storm的实时应用程序，从启动开始一直运行，只要有tuple过来 就会触发执行。拓扑： storm的消息流动很像一个拓扑结构。

2. stream是storm的核心概念，一个stream是一个持续的tuple序列，这些tuple被以分布式并行的方式创建和处理。

3. spouts是一个stream的源头，spouts负责从外部系统读取数据，并组装成tuple发射出去，tuple被发射后就开始再topology中传播。

4. bolt是storm中处理 数据的核心，storm中所有的数据处理都是在bolt中完成的
这里就简单介绍一些概念 具体的可以看看详细的教程。

我们接下来开始整合storm和kafka。

从上面的介绍得知storm的spout是负责从外部读取数据的 所以我们需要开发一个KafkaSpout 来作为kafka的消费者和storm的数据接收源。可以看看这个<https://github.com/HolmesNL/kafka-spout>。我在下面只写一个简单的可供测试。

具体代码:

```
[java] view plain copy print ?

01. public class KafkaSpout implements IRichSpout {
02.
03.     private static final Log logger = LoggerFactory.getLog(KafkaSpout.class);
04.     /**
05.      *
06.      */
07.     private static final long serialVersionUID = -5569857211173547938L;
08.     SpoutOutputCollector collector;
09.     private ConsumerConnector consumer;
10.     private String topic;
11.
12.     public KafkaSpout(String topic) {
13.         this.topic = topic;
14.     }
15.
16.     @Override
17.     public void open(Map conf, TopologyContext context,
18.         SpoutOutputCollector collector) {
19.         this.collector = collector;
20.
21.     }
22.
23.     private static ConsumerConfig createConsumerConfig() {
24.         Properties props = new Properties();
25.         props.put("zookeeper.connect", "xx.xx.xx.xx:2181");
26.         props.put("group.id", "0");
27.         props.put("zookeeper.session.timeout.ms", "10000");
28.         //props.put("zookeeper.sync.time.ms", "200");
29.         //props.put("auto.commit.interval.ms", "1000");
30.
31.         return new ConsumerConfig(props);
32.     }
33.
34.     @Override
35.     public void close() {
36.         // TODOAuto-generated method stub
37.
38.     }
39.
40.     @Override
41.     public void activate() {
42.         this.consumer = Consumer.createJavaConsumerConnector(createConsumerConfig());
43.         Map<String, Integer> topicMap = new HashMap<String, Integer>();
44.         topicMap.put(topic, new Integer(1));
45.         Map<String, List<KafkaStream<byte[], byte[]>>> streamMap = consumer.createMessageStream
46.             KafkaStream<byte[], byte[]> stream = streamMap.get(topic).get(0);
47.         ConsumerIterator<byte[], byte[]> it = stream.iterator();
48.         while (it.hasNext()) {
49.             String value = new String(it.next().message());
50.             System.out.println("(consumer)-->" + value);
51.             collector.emit(new Values(value), value);
52.         }
53.
54.     }
55.
56.     @Override
57.     public void deactivate() {
58.         // TODOAuto-generated method stub
59.
60.     }
61.
62.     private boolean isComplete;
63.
64.     @Override
65.     public void nextTuple() {
66.
67.     }
68.
69.     @Override
70.     public void ack(Object msgId) {
```



```
71.         // TODOAuto-generated method stub
72.
73.     }
74.
75.     @Override
76.     public void fail(Object msgId) {
77.         // TODOAuto-generated method stub
78.
79.     }
80.
81.     @Override
82.     public void declareOutputFields(OutputFieldsDeclarer declarer) {
83.         declarer.declare(new Fields("KafkaSpout"));
84.
85.     }
86.
87.     @Override
88.     public Map<String, Object> getComponentConfiguration() {
89.         // TODOAuto-generated method stub
90.         return null;
91.     }
92.
93. }
94.
95. public class FileBlots implementsIRichBolt{
96.
97.     OutputCollector collector;
98.
99.     public void prepare(Map stormConf, TopologyContext context,
100.         OutputCollector collector) {
101.         this.collector = collector;
102.
103.     }
104.
105.     public void execute(Tuple input) {
106.         String line = input.getString(0);
107.         for(String str : line.split("\\s+")){
108.             List a = newArrayList();
109.             a.add(input);
110.             this.collector.emit(a,newValues(str));
111.         }
112.         this.collector.ack(input);
113.     }
114.
115.     public void cleanup() {
116.
117.     }
118.
119.     public void declareOutputFields(OutputFieldsDeclarer declarer) {
120.         declarer.declare(new Fields("words"));
121.
122.     }
123.
124.     public Map<String, Object> getComponentConfiguration() {
125.         // TODOAuto-generated method stub
126.         return null;
127.     }
128.
129. }
130. public class WordsCounterBlots implementsIRichBolt{
131.
132.     OutputCollector collector;
133.     Map<String, Integer> counter;
134.
135.     public void prepare(Map stormConf, TopologyContext context,
136.         OutputCollector collector) {
137.         this.collector = collector;
138.         this.counter =new HashMap<String, Integer>();
139.
140.     }
141.
142.     public void execute(Tuple input) {
143.         String word = input.getString(0);
144.         Integer integer = this.counter.get(word);
145.         if(integer !=null){
146.             integer +=1;
147.             this.counter.put(word, integer);
148.         }else{
149.             this.counter.put(word, 1);
```

```
150.         }
151.         System.out.println("execute");
152.         Jedis jedis = JedisUtils.getJedis();
153.         jedis.incrBy(word, 1);
154.         System.out.println("=====");
155.         this.collector.ack(input);
156.     }
157.
158.     public void cleanup() {
159.         for(Entry<String, Integer> entry :this.counter.entrySet()){
160.             System.out.println("-----:"+entry.getKey()+"=="+entry.getValue());
161.         }
162.     }
163.
164.
165.     public void declareOutputFields(OutputFieldsDeclarer declarer) {
166.
167.
168.     }
169.
170.     public Map<String, Object> getComponentConfiguration() {
171.         // TODOAuto-generated method stub
172.         return null;
173.     }
174.
175. }
```

Topology测试:

```
[java] view plain copy print ?
01. public class KafkaTopology {
02.
03.     public static void main(String[] args) {
04.         try {
05.             JedisUtils.initialPool("xx.xx.xx.xx", 6379);
06.         } catch (Exception e) {
07.             e.printStackTrace();
08.         }
09.
10.         TopologyBuilder builder = newTopologyBuilder(); builder.setSpout("kafka",1);
11.         builder.setBolt("file-blots",new FileBlots()).shuffleGrouping("kafka");
12.         builder.setBolt("words-counter",new WordsCounterBlots(),2).fieldsGrouping("file-
13.         blots",new Fields("words"));
14.         Config config = new Config();
15.         config.setDebug(true);
16.         LocalCluster local = newLocalCluster();
17.         local.submitTopology("counter", config, builder.createTopology());
18.     }
19. }
```

至此flume + kafka+storm的整合就写完了。注意 这个是 初始学习阶段做的测试 不可正式用于线上环境，在写本文之时 已经离测试过去了一段时间 所以可能会有些错误 请见谅。

顶 6 踩 0

- 上一篇 [kafka Failed to send messages after 3 tries 问题解决](#)
- 下一篇 [jdk6的下载地址](#)

我的同类文章

storm (1)	kafka (1)
-----------	-----------

• storm 例子 记录

2014-01-10 阅读 632

猜你在找

- Storm应用开发系列从入门到精通
- hbase入门精讲
- Ceph—分布式存储系统的另一个选择
- Jmeter性能测试全程实战
- 移动APP测试基础到进阶

计算机学院排

计算机学校排

ios学习路线

中国网游排行


外星人笔记本

硬盘数据恢复

linux学习

查看评论

4楼 [zengqingchun](#) 2015-12-26 22:04发表



JedisUtils 这个类是自己写的吧 查api没有这个类

3楼 [u010409394](#) 2015-09-24 11:43发表




前辈，本文中的“Worker：处理逻辑的进程，在其中运行着多个Task，每个task 是一组spout/blots的组合。”这句话，描述的好像不太合理，希望及时更正过来，我qq是826855366

2楼 [cepgurugen](#) 2014-12-15 14:12发表



感谢分享，Kafka可以连接到hadoop,DW等非流式数据处理系统。如果只是为了连接到流式处理引擎，需要保证消息队列系统的稳定。<http://www.sodbase.com/news/html/?422.html>

1楼 [yebai](#) 2014-06-03 20:58发表



弄得我以为是线上系统呢。

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题
- Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
- VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
- BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
- Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack
- FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
- Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
- Angular Cloud Foundry Redis Scala Django Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

