实时日志分析部署手册

|  |  |  |
| --- | --- | --- |
| 修订人 | 内容 | 日期 |
| 刘巍 | 创建 | 2014-11-05 |
|  |  |  |
|  |  |  |

**目 录**

目录

[1. 系统概要 3](#_Toc403138205)

[2. 系统构架图 3](#_Toc403138206)

[3. 系统安装 3](#_Toc403138207)

[3.1. ZOOKEEPER 3](#_Toc403138208)

[3.2. FLUME 4](#_Toc403138209)

[3.3. KAFKA 5](#_Toc403138210)

[3.4. STORM 5](#_Toc403138211)

1. 系统概要

apache-flume-1.5.0.1

kafka\_2.9.2-0.8.1.1

apache-storm-0.9.2-incubating

jdk-7u71-linux-x64

zookeeper-3.4.6

1. 系统构架图



Flume Agent



Kafka



Storm



Hbase/Mongo



Zookeeper



1. 系统安装
   1. ZOOKEEPER

假设以三台服务器为例:

|  |  |
| --- | --- |
| IP | 域名 |
| 192.168.56.101 | zkserver1 |
| 192.168.56.102 | zkserver2 |
| 192.168.56.103 | zkserver3 |

* 关闭防火墙

service iptables stop

* 解压zookeeper安装包

tar zxvf zookeeper-3.4.6.tar.gz

* 编辑zookeeper配置文件

cp conf/zoo\_sample.cfg conf/zoo.cfg

|  |
| --- |
| tickTime=2000  initLimit=10  syncLimit=5  dataDir=/tmp/zookeeper  clientPort=2181  server.1=zkserver1:2888:3888  server.2=zkserver2:2888:3888  server.3=zkserver3:2888:3888 |

* 拷贝到其他机器上

scp -r zookeeper-3.4.6 root@zkserver2:/ home/test/

scp -r zookeeper-3.4.6 root@zkserver2:/home/test/

* 创建MYID

#zkserver1

echo "1" > /tmp/zookeeper/myid

#zkserver2

echo "2" > /tmp/zookeeper/myid

#zkserver3

echo "3" > /tmp/zookeeper/myid

* 启动zookeeper

分别在三台机器上执行

bin/zkServer.sh start

* 查看状态

bin/zkServer.sh status

* 1. FLUME
* 解压flume安装包

tar zxvf apache-flume-1.5.0.1-bin.tar.gz

* 引入flume-kafka-plugin及相关jar

<https://github.com/beyondj2ee/flumeng-kafka-plugin>

引入相关JAR包

flumeng-kafka-plugin.jar

kafka\_2.9.2-0.8.1.1.jar

metrics-core-2.2.0.jar

scala-library-2.9.2.jar

zkclient-0.3.jar

* 编辑flume配置文件

|  |
| --- |
| a1.sources = r1  a1.channels = c1  a1.sinks = k1  a1.sources.r1.type = spooldir  a1.sources.r1.spoolDir = /var/log/apache/flumeSpool  a1.sources.r1.channels = c1  a1.sinks.k1.type = org.apache.flume.plugins.KafkaSink  a1.sinks.k1.metadata.broker.list=192.168.56.101:9092  a1.sinks.k1.partition.key=0  a1.sinks.k1.partitioner.class=org.apache.flume.plugins.SinglePartition  a1.sinks.k1.serializer.class=kafka.serializer.StringEncoder  a1.sinks.k1.request.required.acks=0  a1.sinks.k1.max.message.size=1000000  a1.sinks.k1.producer.type=sync  a1.sinks.k1.custom.encoding=UTF-8  a1.sinks.k1.custom.topic.name=test  a1.sinks.k1.channel = c1  a1.channels.c1.type = memory  a1.channels.c1.capacity = 1000  a1.channels.c1.transactionCapacity = 100 |

* 启动FLUME

bin/flume-ng agent -c conf/ -f conf/flume\_kafka.conf -n agent &

* 1. KAFKA

假设为两台服务器为例

|  |  |
| --- | --- |
| IP | 域名 |
| 192.168.56.105 | kafkaserver1 |
| 192.168.56.106 | kafkaserver2 |

* 解压kafka安装包

tar zxvf kafka\_2.9.2-0.8.1.1.tgz

* 编辑kafka配置文件

vi config/server.properties

修改相应broker.id,host.name及zookeeper.connect

|  |
| --- |
| broker.id=0  host.name= kafkaserver1  zookeeper.connect= zkserver1:2181, zkserver2:2181, zkserver3:2181 |

* 启动kafka server

bin/kafka-server-start.sh config/server.properties

* 创建topic

bin/kafka-topics.sh --create --zookeeper zkserver1:2181 --replication-factor 1 --partitions 1 --topic test

* 查看是否创建成功

bin/kafka-topics.sh --list --zookeeper zkserver1:2181

* 发送测试消息

bin/kafka-console-producer.sh --broker-list zkserver1:9092 --topic test

This is a message

This is another message

* 接收测试消息

bin/kafka-console-consumer.sh --zookeeper zkserver1:2181 --topic test --from-beginning

* 1. STORM

假设以三台服务器为例

|  |  |
| --- | --- |
| IP | 域名 |
| 192.168.56.107 | stormserver1 |
| 192.168.56.108 | Stormserver2 |
| 192.168.56.109 | Stormserver3 |

* 解压storm

tar zxvf apache-storm-0.9.2-incubating.tar.gz

* 集成kafka插件

<https://github.com/wurstmeister/storm-kafka-0.8-plus>

引入相关JAR包

storm-kafka-0.9.2-incubating-sources.jar

kafka\_2.9.2-0.8.1.1.jar

metrics-core-2.2.0.jar

scala-library-2.9.2.jar

zkclient-0.3.jar

* 修改storm.yaml配置文件

|  |
| --- |
| storm.zookeeper.servers:  - " zkserver1"  - " zkserver2"  - " zkserver3"  nimbus.host:  " stormserver1"  storm.local.dir:  "/home/vtest/storm-0.9.2/workdir"  supervisor.slots.ports:  - 6700  - 6701  - 6702  - 6703 |

* 启动Storm进程

在nimbus机器的Storm的bin目录下执行

启动nimbus

bin/storm nimbus &

启动ui

bin/storm ui &

在supervisor机器的Storm的bin目录下执行

bin/storm supervisor &

* 查看storm运行结果

查看UI界面

stormserver1:8080

命令行

bin/storm list

* 运行自带例子

bin/storm jar storm-starter-0.0.1-SNAPSHOT-jar-with-dependencies.jar storm.starter.WordCountTopology test

1. 单机STORM 开发环境搭建

* 在eclipse中新建maven的quickstart项目,POM文件修改如下:

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.mobogenie</groupId>  <artifactId>storm-mobo</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>storm-mobo</name>  <url>http://maven.apache.org</url>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  </properties>  <repositories>  <repository>  <releases>  <enabled>true</enabled>  </releases>  <snapshots>  <enabled>false</enabled>  </snapshots>  <id>central</id>  <url>http://repo1.maven.org/maven2/</url>  </repository>  <repository>  <releases>  <enabled>true</enabled>  </releases>  <snapshots>  <enabled>true</enabled>  </snapshots>  <id>clojars</id>  <url>https://clojars.org/repo/</url>  </repository>  </repositories>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>3.8.1</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.apache.storm</groupId>  <artifactId>storm-core</artifactId>  <version>0.9.2-incubating</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>org.twitter4j</groupId>  <artifactId>twitter4j-stream</artifactId>  <version>3.0.3</version>  </dependency>  <dependency>  <groupId>commons-collections</groupId>  <artifactId>commons-collections</artifactId>  <version>3.2.1</version>  </dependency>    <dependency>  <groupId>org.apache.storm</groupId>  <artifactId>storm-kafka</artifactId>  <version>0.9.2-incubating</version>  </dependency>    <dependency>  <groupId>org.apache.kafka</groupId>  <artifactId>kafka\_2.9.2</artifactId>  <version>0.8.1.1</version>  <exclusions>  <exclusion>  <groupId>org.apache.zookeeper</groupId>  <artifactId>zookeeper</artifactId>  </exclusion>  <exclusion>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  </exclusion>  </exclusions>  </dependency>    <dependency>  <groupId>org.apache.tomcat</groupId>  <artifactId>tomcat-jdbc</artifactId>  <version>7.0.56</version>  </dependency>    <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.22</version>  </dependency>  </dependencies>  </project> |

* LocalCluster类说明

LocalCluster类可以在本机开发环境上进行Storm的topology的测试,例如以下代码的绿色部分

|  |
| --- |
| **public** **class** MyTopology {  **public** **static** **void** main(String[] args) **throws** Exception {  TopologyBuilder builder = **new** TopologyBuilder();    builder.setSpout("spout", **new** RandomSpout());  builder.setBolt("mybolt", **new** Mybolt()).shuffleGrouping("spout");  builder.setBolt("print", **new** PrintBolt()).shuffleGrouping("mybolt");    Config config = **new** Config();  config.setDebug(**true**);    **if**(args != **null** && args.length > 0) {  config.setNumWorkers(3);  StormSubmitter.*submitTopology*(args[0], config, builder.createTopology());  }**else** {  LocalCluster cluster = **new** LocalCluster();  cluster.submitTopology("test", config, builder.createTopology());  Utils.*sleep*(10000);  cluster.killTopology("test");  cluster.shutdown();  }    }  } |

1. STORM TOPOLOGY例子
   1. 例子1 - 时间(秒级)单维度PV统计

KafkaSpout从kafka队列中读取日志并以随机数据流组的方式发送到FormatBolt中

FormatBolt对接收到的日志进行截取,得到时间部分以域数据流组的方式发送到PvBolt中

PvBolt按时间的秒为单位进行计数累加,并存入MYSQL数据库中

* FormatBolt类实现

|  |
| --- |
| import java.util.List;  import backtype.storm.topology.BasicOutputCollector;  import backtype.storm.topology.OutputFieldsDeclarer;  import backtype.storm.topology.base.BaseBasicBolt;  import backtype.storm.tuple.Fields;  import backtype.storm.tuple.Tuple;  import backtype.storm.tuple.Values;  public class FormatBolt extends BaseBasicBolt {  private List<Integer> pvTasks;    @Override  public void execute(Tuple input, BasicOutputCollector collector) {  String log = input.getString(0);  String[] logfield = log.split("###");  if(logfield.length > 0) {  collector.emit(new Values(logfield[0]));  }  }  @Override  public void declareOutputFields(OutputFieldsDeclarer declarer) {  declarer.declare(new Fields("logDate"));  }    } |

* PvBolt类实现

|  |
| --- |
| **import** java.sql.Connection;  **import** java.sql.ResultSet;  **import** java.sql.SQLException;  **import** java.sql.Statement;  **import** java.util.HashMap;  **import** java.util.Map;  **import** java.util.Map.Entry;  **import** org.apache.tomcat.jdbc.pool.DataSource;  **import** backtype.storm.task.TopologyContext;  **import** backtype.storm.topology.BasicOutputCollector;  **import** backtype.storm.topology.OutputFieldsDeclarer;  **import** backtype.storm.topology.base.BaseBasicBolt;  **import** backtype.storm.tuple.Tuple;  **import** com.v.SimplePOJOExample;  **public** **class** PvBolt **extends** BaseBasicBolt {  **private** Map<String, Integer> pvMap;  **private** DataSource datasource;  @Override  **public** **void** prepare(Map stormConf, TopologyContext context) {  **this**.pvMap = **new** HashMap<String, Integer>();  **this**.datasource = SimplePOJOExample.*getDataSource*();  }  @Override  **public** **void** execute(Tuple input, BasicOutputCollector collector) {  String dateStr = input.getString(0);  System.*out*.println("==========================" + dateStr);  **if** (pvMap.containsKey(dateStr)) {  pvMap.put(dateStr, pvMap.get(dateStr) + 1);  } **else** {  **if** (!pvMap.isEmpty()) {  // **TODO** 存入MYSQL  Connection con = **null**;  **try** {  con = datasource.getConnection();  Statement st = con.createStatement();  **for** (Entry<String, Integer> entry : pvMap.entrySet()) {  **int** count = st.executeUpdate("INSERT INTO testpv (log\_date, log\_count) "  + "VALUES ('" + entry.getKey() + "', " + entry.getValue() + ")");  }  st.close();  } **catch** (SQLException e) {  e.printStackTrace();  }**finally** {  **if** (con!=**null**) **try** {con.close();}**catch** (Exception ignore) {}  }  pvMap.clear();  }  pvMap.put(dateStr, 1);  }  }  @Override  **public** **void** declareOutputFields(OutputFieldsDeclarer declarer) {  }  } |

* PvTopology类实现

|  |
| --- |
| **import** storm.kafka.BrokerHosts;  **import** storm.kafka.KafkaSpout;  **import** storm.kafka.SpoutConfig;  **import** storm.kafka.StringScheme;  **import** storm.kafka.ZkHosts;  **import** backtype.storm.Config;  **import** backtype.storm.LocalCluster;  **import** backtype.storm.StormSubmitter;  **import** backtype.storm.generated.StormTopology;  **import** backtype.storm.spout.SchemeAsMultiScheme;  **import** backtype.storm.topology.TopologyBuilder;  **import** backtype.storm.tuple.Fields;  **public** **class** PvTopology {    **private** **final** BrokerHosts brokerHosts;  **public** PvTopology(String kafkaZookeeper) {  brokerHosts = **new** ZkHosts(kafkaZookeeper);  }  **public** StormTopology buildTopology() {  SpoutConfig kafkaConfig = **new** SpoutConfig(brokerHosts, "test", "", "storm");  kafkaConfig.scheme = **new** SchemeAsMultiScheme(**new** StringScheme());    TopologyBuilder builder = **new** TopologyBuilder();  builder.setSpout("logs", **new** KafkaSpout(kafkaConfig), 10);  builder.setBolt("format", **new** FormatBolt(), 3).shuffleGrouping("logs");  builder.setBolt("pv", **new** PvBolt(), 3).fieldsGrouping("format", **new** Fields("logDate"));  **return** builder.createTopology();  }  **public** **static** **void** main(String[] args) **throws** Exception {  String kafkaZk = "192.168.56.101";  PvTopology pvTopology = **new** PvTopology(kafkaZk);  Config config = **new** Config();  config.put(Config.*TOPOLOGY\_TRIDENT\_BATCH\_EMIT\_INTERVAL\_MILLIS*, 2000);  StormTopology stormTopology = pvTopology.buildTopology();  **if** (args != **null** && args.length > 1) {  String name = args[0];  config.setNumWorkers(2);  config.setMaxTaskParallelism(5);  StormSubmitter.*submitTopology*(name, config, stormTopology);  } **else** {  config.setNumWorkers(2);  config.setMaxTaskParallelism(2);  LocalCluster cluster = **new** LocalCluster();  cluster.submitTopology("kafka", config, stormTopology);  }  }  } |

* 1. 例子2

1. 系统监控

ganglia nagios zabbix

flume,kafka,storm数据走zookeeper接口的数据报警监控

flume,kafka,storm 数据量监控

略

1. JVM监控

Jstat ?

略

1. 简单图表输出

Kibana ?

略

1. 系统优化

Flume spooldir已读取日志可删除?

GC优化

略