Storm WordCount示例

**一、关联代码**

使用maven，代码如下。

RandomSentenceSpout.java

/\*\*

\* Licensed to the Apache Software Foundation (ASF) under one

\* or more contributor license agreements. See the NOTICE file

\* distributed with this work for additional information

\* regarding copyright ownership. The ASF licenses this file

\* to you under the Apache License, Version 2.0 (the

\* "License"); you may not use this file except in compliance

\* with the License. You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package cn.ljh.storm.wordcount;

import org.apache.storm.spout.SpoutOutputCollector;

import org.apache.storm.task.TopologyContext;

import org.apache.storm.topology.OutputFieldsDeclarer;

import org.apache.storm.topology.base.BaseRichSpout;

import org.apache.storm.tuple.Fields;

import org.apache.storm.tuple.Values;

import org.apache.storm.utils.Utils;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Map;

import java.util.Random;

public class RandomSentenceSpout extends BaseRichSpout {

private static final Logger LOG = LoggerFactory.getLogger(RandomSentenceSpout.class);

SpoutOutputCollector \_collector;

Random \_rand;

public void open(Map conf, TopologyContext context, SpoutOutputCollector collector) {

\_collector = collector;

\_rand = new Random();

}

public void nextTuple() {

Utils.sleep(100);

String[] sentences = new String[]{

sentence("the cow jumped over the moon"),

sentence("an apple a day keeps the doctor away"),

sentence("four score and seven years ago"),

sentence("snow white and the seven dwarfs"),

sentence("i am at two with nature")};

final String sentence = sentences[\_rand.nextInt(sentences.length)];

LOG.debug("Emitting tuple: {}", sentence);

\_collector.emit(new Values(sentence));

}

protected String sentence(String input) {

return input;

}

@Override

public void ack(Object id) {

}

@Override

public void fail(Object id) {

}

public void declareOutputFields(OutputFieldsDeclarer declarer) {

declarer.declare(new Fields("word"));

}

// Add unique identifier to each tuple, which is helpful for debugging

public static class TimeStamped extends RandomSentenceSpout {

private final String prefix;

public TimeStamped() {

this("");

}

public TimeStamped(String prefix) {

this.prefix = prefix;

}

protected String sentence(String input) {

return prefix + currentDate() + " " + input;

}

private String currentDate() {

return new SimpleDateFormat("yyyy.MM.dd\_HH:mm:ss.SSSSSSSSS").format(new Date());

}

}

}

WordCountTopology.java

/\*\*

\* Licensed to the Apache Software Foundation (ASF) under one

\* or more contributor license agreements. See the NOTICE file

\* distributed with this work for additional information

\* regarding copyright ownership. The ASF licenses this file

\* to you under the Apache License, Version 2.0 (the

\* "License"); you may not use this file except in compliance

\* with the License. You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package cn.ljh.storm.wordcount;

import org.apache.storm.Config;

import org.apache.storm.LocalCluster;

import org.apache.storm.StormSubmitter;

import org.apache.storm.task.OutputCollector;

import org.apache.storm.task.TopologyContext;

import org.apache.storm.topology.BasicOutputCollector;

import org.apache.storm.topology.IRichBolt;

import org.apache.storm.topology.OutputFieldsDeclarer;

import org.apache.storm.topology.TopologyBuilder;

import org.apache.storm.topology.base.BaseBasicBolt;

import org.apache.storm.tuple.Fields;

import org.apache.storm.tuple.Tuple;

import org.apache.storm.tuple.Values;

import java.util.ArrayList;

import java.util.Collections;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class WordCountTopology {

public static class SplitSentence implements IRichBolt {

private OutputCollector \_collector;

public void declareOutputFields(OutputFieldsDeclarer declarer) {

declarer.declare(new Fields("word"));

}

public Map<String, Object> getComponentConfiguration() {

return null;

}

public void prepare(Map stormConf, TopologyContext context,

OutputCollector collector) {

\_collector = collector;

}

public void execute(Tuple input) {

String sentence = input.getStringByField("word");

String[] words = sentence.split(" ");

for(String word : words){

this.\_collector.emit(new Values(word));

}

}

public void cleanup() {

// TODO Auto-generated method stub

}

}

public static class WordCount extends BaseBasicBolt {

Map<String, Integer> counts = new HashMap<String, Integer>();

public void execute(Tuple tuple, BasicOutputCollector collector) {

String word = tuple.getString(0);

Integer count = counts.get(word);

if (count == null)

count = 0;

count++;

counts.put(word, count);

collector.emit(new Values(word, count));

}

public void declareOutputFields(OutputFieldsDeclarer declarer) {

declarer.declare(new Fields("word", "count"));

}

}

public static class WordReport extends BaseBasicBolt {

Map<String, Integer> counts = new HashMap<String, Integer>();

public void execute(Tuple tuple, BasicOutputCollector collector) {

String word = tuple.getStringByField("word");

Integer count = tuple.getIntegerByField("count");

this.counts.put(word, count);

}

public void declareOutputFields(OutputFieldsDeclarer declarer) {

}

@Override

public void cleanup() {

System.out.println("-----------------FINAL COUNTS START-----------------------");

List<String> keys = new ArrayList<String>();

keys.addAll(this.counts.keySet());

Collections.sort(keys);

for(String key : keys){

System.out.println(key + " : " + this.counts.get(key));

}

System.out.println("-----------------FINAL COUNTS END-----------------------");

}

}

public static void main(String[] args) throws Exception {

TopologyBuilder builder = new TopologyBuilder();

builder.setSpout("spout", new RandomSentenceSpout(), 5);

//ShuffleGrouping：随机选择一个Task来发送。

builder.setBolt("split", new SplitSentence(), 8).shuffleGrouping("spout");

//FiledGrouping：根据Tuple中Fields来做一致性hash，相同hash值的Tuple被发送到相同的Task。

builder.setBolt("count", new WordCount(), 12).fieldsGrouping("split", new Fields("word"));

//GlobalGrouping：所有的Tuple会被发送到某个Bolt中的id最小的那个Task。

builder.setBolt("report", new WordReport(), 6).globalGrouping("count");

Config conf = new Config();

conf.setDebug(true);

if (args != null && args.length > 0) {

conf.setNumWorkers(3);

StormSubmitter.submitTopologyWithProgressBar(args[0], conf, builder.createTopology());

}

else {

conf.setMaxTaskParallelism(3);

LocalCluster cluster = new LocalCluster();

cluster.submitTopology("word-count", conf, builder.createTopology());

Thread.sleep(20000);

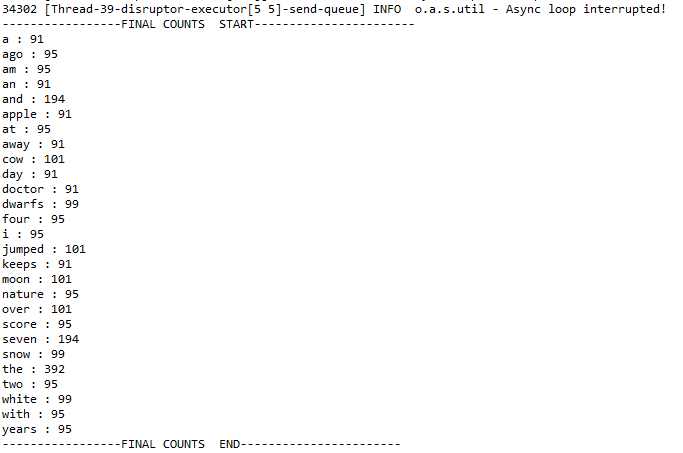
cluster.shutdown();

}

}

}

二、执行效果

[](http://images2015.cnblogs.com/blog/1157155/201706/1157155-20170609095450137-1167588933.png)