**Kafka 学习笔记**

**Kafka 搭建好了之后，在本地使用console启动生产者和消费者都没有问题，但是在另外一台主机上编写客户端代码来发布和接收消息却没有反应，找了半天原因， 终于找到了解决办法 :**

**修改kafka 配置文件中的中的IP 为服务器的实际IP**

**advertised.listeners=PLAINTEXT://192.168.0.101:9092**

**原文描述如下:**

**Hostname and port the broker will advertise to producers and consumers. If not set,**

**it uses the value for "listeners" if configured. Otherwise, it will use the value**

**returned from java.net.InetAddress.getCanonicalHostName().**

**发送给生产者和消费者的主机名和端口号，如果该值没有设置，则为使用listeners属性配置的值。如果listeners 也没有设置，它将会使用 java.net.InetAddress.getCanonicalHostName()获取的值。 java.net.InetAddress.getCanonicalHostName()获取的是localhost**

**现在知道原因了将其改为主机的实际IP和端口后就正常了。**

客户端代码:

生产者

Properties props=**new** Properties();

props.put("bootstrap.servers", "192.168.0.101:9092");

props.put("key.serializer", StringSerializer.**class**);

props.put("value.serializer", StringSerializer.**class**);

Producer<String,String> producer=**new** KafkaProducer(props);

**for** (**int** i = 0; i <100000; i++) {

Thread.*sleep*(1000);

producer.send(**new** ProducerRecord<String, String>("test", Integer.*toString*(i),"mmmmmm nnnn gggg"));

}

producer.close();

消费者

Properties props=**new** Properties();

props.put("bootstrap.servers", "192.168.0.101:9092");

props.put("group.id", "testab");

props.put("enable.auto.commit", "true");

props.put("auto.commit.intervals.ms", "1000");

props.put("key.deserializer", StringDeserializer.**class**);

props.put("value.deserializer", StringDeserializer.**class**);

props.put("auto.offset.reset", "latest");

KafkaConsumer<String, String> consumer=**new** KafkaConsumer<String, String>(props);

consumer.subscribe(Arrays.*asList*("test"));

**while** (**true**) {

ConsumerRecords<String, String> records=consumer.poll(100);

**for** (ConsumerRecord<String, String> record:records) {

System.***out***.println(record.key()+" "+record.value());

}

}

上面是生产者和消费者模式的代码

kafka也支持流计算，下面是示例代码

**public** **class** KafkaMain {

//自定义一个序列反序列化类

**public** **static** **class** MySerde **implements** Serde<Windowed<String>>{

**public** **void** configure(Map<String, ?> configs, **boolean** isKey) {

}

**public** **void** close() {

}

**public** Serializer<Windowed<String>> serializer() {

**return** **new** Serializer<Windowed<String>>() {

**public** **void** configure(Map<String, ?> configs, **boolean** isKey) {

}

**public** **byte**[] serialize(String topic, Windowed<String> data) {

**return** **new** org.apache.kafka.common.serialization.StringSerializer().serialize(topic,

data.key() + "\_" + data.window().start() + "\_" + data.window().end());

}

**public** **void** close() {

}

};

}

**public** Deserializer<Windowed<String>> deserializer() {

**return** **new** Deserializer<Windowed<String>>() {

**public** Windowed<String> deserialize(String topic, **byte**[] data) {

String v = **new** StringDeserializer().deserialize(topic, data);

String[] strs = v.split("\_");

org.apache.kafka.streams.kstream.Window window = **new** Window(Long.*valueOf*(strs[1]),

Long.*valueOf*(strs[2])) {

};

Windowed<String> windowed = **new** Windowed<String>(strs[0], window);

**return** windowed;

}

**public** **void** configure(Map<String, ?> configs, **boolean** isKey) {

}

**public** **void** close() {

}

};

}

}

//主函数

**public** **static** **void** main(String[] args) {

java.util.Properties config = **new** Properties();

config.put(StreamsConfig.***APPLICATION\_ID\_CONFIG***, "app");//应用程序的ID

config.put(StreamsConfig.***BOOTSTRAP\_SERVERS\_CONFIG***, "192.168.0.101:9092");//kafka 的broker主机,如果是多个中间用","号间隔

config.put(StreamsConfig.***KEY\_SERDE\_CLASS\_CONFIG***, Serdes.*String*().getClass());//

config.put(StreamsConfig.***COMMIT\_INTERVAL\_MS\_CONFIG***, "60000"); //多长时间进行一次提交,这个时间应大于windows窗口期的时间间隔,例如每一个windows窗口1分钟时间作为一个计算时间片段，如果10分钟作一次提交的话，我们一次(批)可以获得10个计算结果,

config.put(StreamsConfig.***VALUE\_SERDE\_CLASS\_CONFIG***, Serdes.*String*().getClass()); //序列化反序列化类

config.put(org.apache.kafka.clients.consumer.ConsumerConfig.***AUTO\_OFFSET\_RESET\_CONFIG***, "latest");//offset 开始,latest表示只消费最新的记录

KStreamBuilder builder = **new** KStreamBuilder();

KStream<String, String> testLines = builder.stream("test");//消费test主题

KStream<String, String> testLines2 = testLines.flatMapValues(**new** ValueMapper<String, Iterable<String>>() {

**public** Iterable<String> apply(String value) {

String[] strs = value.split("\\s+"); //按空格拆分单词

List<String> list = **new** ArrayList<String>();

**for** (String str : strs) {

list.add(str);

}

**return** list;

}

});

//每个单词算 一个值

KStream<String, Long> testLines3 = testLines2.map(**new** KeyValueMapper<String, String, KeyValue<String, Long>>() {

**public** KeyValue<String, Long> apply(String key, String value) {

**return** **new** KeyValue<String, Long>(value, 1L);

}

});

//按单词名称进行一次分组

KGroupedStream<String, Long> gstream = testLines3.groupByKey(Serdes.*String*(), Serdes.*Long*());

//计算每个组的单词数量

KTable<Windowed<String>, Long> table2 = gstream.count(TimeWindows.*of*(10000L), "aa"); //窗口为10秒钟

table2.foreach(**new** ForeachAction<Windowed<String>, Long>() {

**public** **void** apply(Windowed<String> key, Long value) {

System.***out***.println(key.key() + "..............." + value + " " + **new** Date() + " "

+ **new** Date(key.window().start()) + "<---->" + **new** Date(key.window().end()));

}

});

//将计算结果重新写入到消息队列中去

table2.to(**new** MySerde(), Serdes.*Long*(), "result");

KafkaStreams streams = **new** KafkaStreams(builder, config);

streams.start();

}

}

下面是7组计算结果：

mmmmmm...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:04:40 CST 2017<---->Sat Jan 28 14:04:50 CST 2017

nnnn...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:04:40 CST 2017<---->Sat Jan 28 14:04:50 CST 2017

gggg...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:04:40 CST 2017<---->Sat Jan 28 14:04:50 CST 2017

mmmmmm...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:04:50 CST 2017<---->Sat Jan 28 14:05:00 CST 2017

nnnn...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:04:50 CST 2017<---->Sat Jan 28 14:05:00 CST 2017

gggg...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:04:50 CST 2017<---->Sat Jan 28 14:05:00 CST 2017

mmmmmm...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:00 CST 2017<---->Sat Jan 28 14:05:10 CST 2017

nnnn...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:00 CST 2017<---->Sat Jan 28 14:05:10 CST 2017

gggg...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:00 CST 2017<---->Sat Jan 28 14:05:10 CST 2017

mmmmmm...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:10 CST 2017<---->Sat Jan 28 14:05:20 CST 2017

nnnn...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:10 CST 2017<---->Sat Jan 28 14:05:20 CST 2017

gggg...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:10 CST 2017<---->Sat Jan 28 14:05:20 CST 2017

mmmmmm...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:20 CST 2017<---->Sat Jan 28 14:05:30 CST 2017

nnnn...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:20 CST 2017<---->Sat Jan 28 14:05:30 CST 2017

gggg...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:20 CST 2017<---->Sat Jan 28 14:05:30 CST 2017

mmmmmm...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:30 CST 2017<---->Sat Jan 28 14:05:40 CST 2017

nnnn...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:30 CST 2017<---->Sat Jan 28 14:05:40 CST 2017

gggg...............10 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:30 CST 2017<---->Sat Jan 28 14:05:40 CST 2017

mmmmmm...............1 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:40 CST 2017<---->Sat Jan 28 14:05:50 CST 2017

nnnn...............1 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:40 CST 2017<---->Sat Jan 28 14:05:50 CST 2017

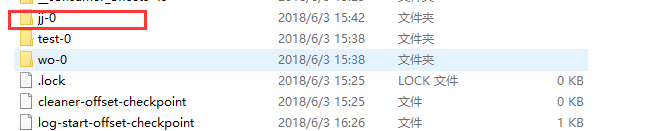
gggg...............1 Sat Jan 28 14:05:40 CST 2017 Sat Jan 28 14:05:40 CST 2017<---->Sat Jan 28 14:05:50 CST 2017

**下面我们分析一下上面的结果，我们生产者每一秒钟发送一条”mmmmmm nnnn gggg” 信息，流计算统计每10秒钟(窗口)中每个单词出现的次数,计算结果每60秒钟(提交时间)提交一次,按理说我们应该获得60/10=6个结果，这里怎么多了一个呢？这个我们不必纠结，kafka的划分片断没有那么精确，如果刚好在临界点之前来了一条记录，那我们也就刚好多了一条记录，不过不必担心，虽说第7组记录统计是不准确的，不过在下一批提交的结果中，第7组就会作为下一批的第一个结果，而且结果是正常的。这个是kafka 流式计算的时候我们需要理解的**

**说明：我在单机上做了kafka性能的测试，结果是生产者发送一百万条数据的数据的时间仅为1S，生产者消费的速度也为一秒钟10万条记录，这说明对于分布式系统的日志收集来说，kafka完全胜任。**

**例如我有三台broker，然后将分区数配置为3,即num.partitions=3，那么我们创建的新的topic就会在每个broker下面都有一个副本，如下图，新建一人topic：jj,各broker对应的log目录下面就有**

**第一个broker下面**



第二个broker下面



第三个broker下面

