Kafka优化

1、并发优化

（1）利用消费者组，可以开启多个消费者进行消费

利用producer生产10条数据（生产到两个分区），利用消费者组开启2个消费者可以进行不重复消费

```
package com.bupt.comsumer;

import org.apache.kafka.clients.consumer.*;

import java.util.Arrays;

import java.util.Properties;

public class MyConsumer {
    public static void main(String[] args) {
        //1 创建消费者配置信息
        Properties properties = new Properties();
        //2 给配置信息复制
            //连接集群

properties.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,"hadoop101:9092");
            //消费者组
        properties.put(ConsumerConfig.GROUP_ID_CONFIG,"test");
            //开启自动提交
        properties.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "true");
       // properties.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
            //自动提交的延时
        properties.put(ConsumerConfig.AUTO_COMMIT_INTERVAL_MS_CONFIG, "1000");
            //key value的反序列化
        properties.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,"earliest");

properties.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,"org.apache.kafka.co
mmon.serialization.StringDeserializer");

properties.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,"org.apache.kafka.
common.serialization.StringDeserializer");
        //3 创建消费者
        KafkaConsumer<String, String> consumer = new KafkaConsumer<String,
String>(properties);
        //4 订阅主题
        consumer.subscribe(Arrays.asList("first","second"));

        while(true){
            //获取数据
            ConsumerRecords<String, String> consumerRecords =
consumer.poll(100);
            //解析并打印
            for(ConsumerRecord<String, String> Records:consumerRecords){
                System.out.println(Records.offset()+"--------"+Records.key()+"--
----"+Records.value());
            }
```

```
            // consumer.commitSync();
        }

    }
}
```

consumer1控制台

```
188--------null------hello yk 0
189--------null------hello yk 2
190--------null------hello yk 4
191--------null------hello yk 6
192--------null------hello yk 8
```

consumer2控制台

```
162--------null------hello yk 1
163--------null------hello yk 3
164--------null------hello yk 5
165--------null------hello yk 7
166--------null------hello yk 9
```

(2) 利用多线程，开启多线程进行消费

利用多线程开启不同的消费者进行消费

```
package com.bupt.comsumer;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;

import java.util.Arrays;
import java.util.Properties;
public class MyTestThread {

    public static void main(String[] args) {
        new Thread(new MyConsumerThread(),"a").start();

        new Thread(new MyConsumerThread(),"b").start();
    }
}

class MyConsumerThread implements Runnable {
    KafkaConsumer<String, String> consumer;
    public  MyConsumerThread(){
        Properties properties = new Properties();

properties.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,"hadoop101:9092");
        //消费者组
        properties.put(ConsumerConfig.GROUP_ID_CONFIG,"test");
        //开启自动提交
        properties.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "true");
        // properties.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
```

```
        //自动提交的延时
        properties.put(ConsumerConfig.AUTO_COMMIT_INTERVAL_MS_CONFIG, "1000");
        //key value的反序列化
        //properties.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,"earliest");

properties.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,"org.apache.kafka.co
mmon.serialization.StringDeserializer");

properties.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,"org.apache.kafka.
common.serialization.StringDeserializer");
        //3 创建消费者
        consumer = new KafkaConsumer<String, String>(properties);
    }
    @Override
    public void run() {
        consumer.subscribe(Arrays.asList("first"));
        while(true){
            ConsumerRecords<String,String> records = consumer.poll(1000);
            for (ConsumerRecord<String, String> record : records) {
                System.out.println(Thread.currentThread().getName()+"进行了消
费"+record.value()+"分区是"+record.partition()+"偏移量是"+record.offset());
            }

        }

    }
}
```

测试结果:

```
a进行了消费hello yk 0分区是0偏移量是198
a进行了消费hello yk 2分区是0偏移量是199
a进行了消费hello yk 4分区是0偏移量是200
a进行了消费hello yk 6分区是0偏移量是201
b进行了消费hello yk 1分区是1偏移量是172
a进行了消费hello yk 8分区是0偏移量是202
b进行了消费hello yk 3分区是1偏移量是173
b进行了消费hello yk 5分区是1偏移量是174
b进行了消费hello yk 7分区是1偏移量是175
b进行了消费hello yk 9分区是1偏移量是176
```