21377062-王悦扬-第10周作业

1. Kafaka安装配置

1.1 Kafka安装

访问Kafka官网下载页面（https://kafka.apache.org/downloads），下载 kafka_2.13-3.7.0.tgz 到虚拟机“~/Downloads”目录下。

执行如下命令完成Kafka的安装：

```
cd ~/Downloads
sudo tar -zxf  kafka_2.13-3.7.0.tgz -C /usr/local
cd /usr/local
sudo mv kafka_2.13-3.7.0 kafka
sudo chown -R hadoop ./kafka
```

```
hadoop@21377062-VirtualBox:~/Downloads$ cd /usr/local
hadoop@21377062-VirtualBox:/usr/local$ sudo mv kafka_2.13-3.7.0 kafka
hadoop@21377062-VirtualBox:/usr/local$ sudo chown -R hadoop ./kafka
```

1.2 部署Kafka伪分布式集群

进入kafka目录，在此目录下建立一个目录etc，将config文件夹中的zookeeper.properties复制到etc文件目录中。

```
cd /usr/local/kafka
sudo mkdir etc
cd /usr/local/kafka/config
sudo mv zookeeper.properties /usr/local/kafka/etc
```

```
hadoop@21377062-VirtualBox:/usr/local$ cd /usr/local/kafka
hadoop@21377062-VirtualBox:/usr/local/kafka$ sudo mkdir etc
hadoop@21377062-VirtualBox:/usr/local/kafka$ cd /usr/local/kafka/config
hadoop@21377062-VirtualBox:/usr/local/kafka/config$ ls
connect-console-sink.properties    connect-mirror-maker.properties  server.properties
connect-console-source.properties  connect-standalone.properties    tools-log4j.properties
connect-distributed.properties     consumer.properties              trogdor.conf
connect-file-sink.properties       kraft                            zookeeper.properties
connect-file-source.properties     log4j.properties
connect-log4j.properties           producer.properties
hadoop@21377062-VirtualBox:/usr/local/kafka/config$ sudo mv zookeeper.properties /usr/local/kafka/etc
hadoop@21377062-VirtualBox:/usr/local/kafka/config$ cd /usr/local/kafka/etc
hadoop@21377062-VirtualBox:/usr/local/kafka/etc$ ls
zookeeper.properties
```

将config文件夹中的server.properties复制三份至etc文件目录中，分别命名为server-0.properties、server-1.properties、server-2.properties

```
cd /usr/local/kafka/config
sudo cp server.properties /usr/local/kafka/etc/server-0.properties
sudo cp server.properties /usr/local/kafka/etc/server-1.properties
sudo cp server.properties /usr/local/kafka/etc/server-2.properties
```

```
hadoop@21377062-VirtualBox:/usr/local/kafka/etc$ cd /usr/local/kafka/config
hadoop@21377062-VirtualBox:/usr/local/kafka/config$ sudo cp server.properties /usr/local/kafka/etc/server-0.pr
operties
hadoop@21377062-VirtualBox:/usr/local/kafka/config$ sudo cp server.properties /usr/local/kafka/etc/server-1.pr
operties
hadoop@21377062-VirtualBox:/usr/local/kafka/config$ sudo cp server.properties /usr/local/kafka/etc/server-2.pr
operties
hadoop@21377062-VirtualBox:/usr/local/kafka/config$ cd /usr/local/kafka/etc
hadoop@21377062-VirtualBox:/usr/local/kafka/etc$ ls
server-0.properties  server-1.properties  server-2.properties  zookeeper.properties
```

1.3 配置server-X.properties文件

分别编辑三个broker配置server-X.properties文件中的以下信息：

```
broker.id = X
listeners = PLAINTEXT://:9092(9093,9094)
log.dirs.=/tmp/kafka-logsX
```

操作命令：

```
cd /usr/local/kafka/etc
sudo vim ../etc/server-0.properties
sudo vim ../etc/server-1.properties
sudo vim ../etc/server-2.properties
```

```
hadoop@21377062-VirtualBox:/usr/local/kafka/config$ cd /usr/local/kafka/etc
hadoop@21377062-VirtualBox:/usr/local/kafka/etc$ ls
s VBox_GAs_7.0.14 ties  server-1.properties  server-2.properties  zookeeper.properties
hadoop@21377062-VirtualBox:/usr/local/kafka/etc$ cd /usr/local/kafka/etc
hadoop@21377062-VirtualBox:/usr/local/kafka/etc$ ls
server-0.properties  server-1.properties  server-2.properties  zookeeper.properties
hadoop@21377062-VirtualBox:/usr/local/kafka/etc$ sudo vim ../etc/server-0.properties
hadoop@21377062-VirtualBox:/usr/local/kafka/etc$ sudo vim ../etc/server-1.properties
hadoop@21377062-VirtualBox:/usr/local/kafka/etc$ sudo vim ../etc/server-2.properties
```

对于3个伪分布式集群分别进行修改

1.4 启动zookeeper服务器和kafka集群

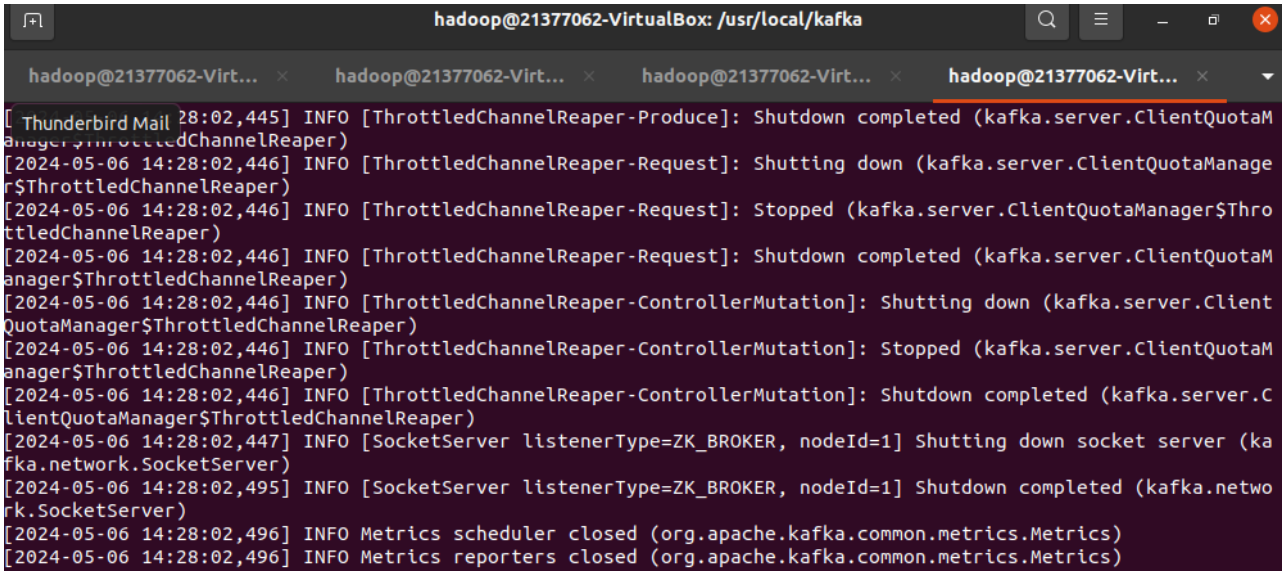
首先启动zookeeper：

```
cd /usr/local/kafka
./bin/zookeeper-server-start.sh  etc/zookeeper.properties
```

```
[2024-05-06 14:26:06,769] INFO Snapshotting: 0x0 to /tmp/zookeeper/version-2/snapshot.0 (org.apache.zookeeper.
server.persistence.FileTxnSnapLog)
[2024-05-06 14:26:06,775] INFO Snapshot loaded in 19 ms, highest zxid is 0x0, digest is 1371985504 (org.apache
.zookeeper.server.ZKDatabase)
[2024-05-06 14:26:06,803] INFO Snapshotting: 0x0 to /tmp/zookeeper/version-2/snapshot.0 (org.apache.zookeeper.
server.persistence.FileTxnSnapLog)
[2024-05-06 14:26:06,804] INFO Snapshot taken in 1 ms (org.apache.zookeeper.server.ZooKeeperServer)
[2024-05-06 14:26:06,823] INFO zookeeper.request_throttler.shutdownTimeout = 10000 ms (org.apache.zookeeper.se
rver.RequestThrottler)
[2024-05-06 14:26:06,823] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeep
er.server.PrepRequestProcessor)
[2024-05-06 14:26:06,855] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.ap
ache.zookeeper.server.ContainerManager)
[2024-05-06 14:26:06,857] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
```

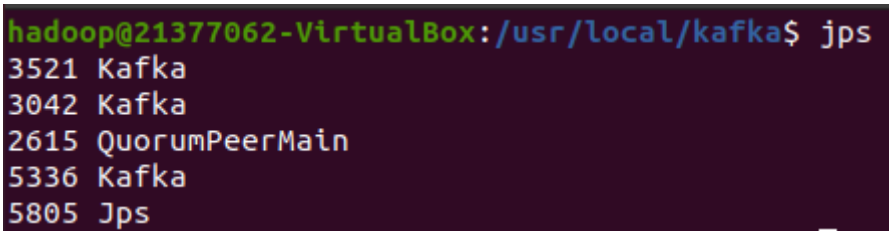
启动kafka集群

```
cd /usr/local/kafka
./bin/kafka-server-start.sh  etc/server-0.properties
./bin/kafka-server-start.sh  etc/server-1.properties
./bin/kafka-server-start.sh  etc/server-2.properties
```



启动集群看是否成功

```
1  jps
```



2. Java API实现数据统计到JSON

2.1 配置Topic

创建3个Topic以完成任务，分别是`comments`,`likes`,和`shares`

```
cd /usr/local/kafka
./bin/kafka-topics.sh --create --topic comments --partitions 3 --replication-factor 2 --bootstrap-server localhost:9092,localhost:9093,localhost:9094

./bin/kafka-topics.sh --create --topic likes --partitions 3 --replication-factor 2 --bootstrap-server localhost:9092,localhost:9093,localhost:9094

./bin/kafka-topics.sh --create --topic shares --partitions 3 --replication-factor 2 --bootstrap-server localhost:9092,localhost:9093,localhost:9094
```

```
hadoop@21377062-VirtualBox:/usr/local/kafka$ cd /usr/local/kafka
hadoop@21377062-VirtualBox:/usr/local/kafka$ ./bin/kafka-topics.sh --create --topic comments --partitions 3 --
replication-factor 2 --bootstrap-server localhost:9092,localhost:9093,localhost:9094
Created topic comments.
hadoop@21377062-VirtualBox:/usr/local/kafka$ ./bin/kafka-topics.sh --create --topic likes --partitions 3 --rep
lication-factor 2 --bootstrap-server localhost:9092,localhost:9093,localhost:9094
Created topic likes.
hadoop@21377062-VirtualBox:/usr/local/kafka$ ./bin/kafka-topics.sh --create --topic shares --partitions 3 --re
plication-factor 2 --bootstrap-server localhost:9092,localhost:9093,localhost:9094
Created topic shares.
```

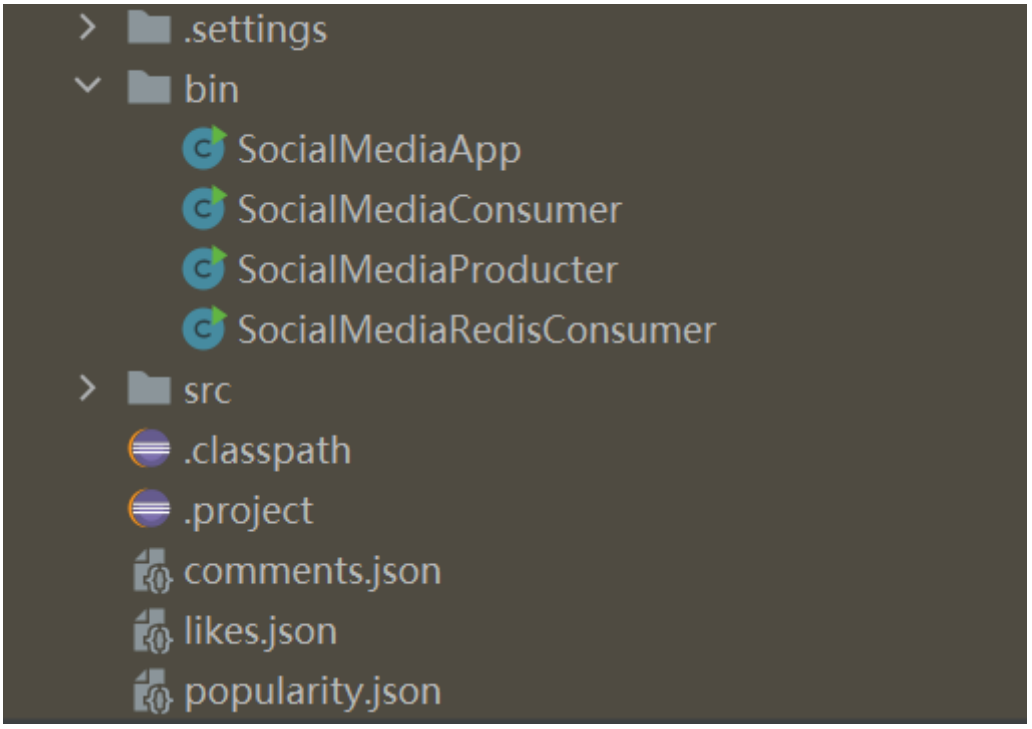
查看创建的Topics

```
cd /usr/local/kafka
./bin/kafka-topics.sh --describe --topic comments --bootstrap-server localhost:9092,localhost:9093,localhost:9094
./bin/kafka-topics.sh --describe --topic likes,shares --bootstrap-server localhost:9092,localhost:9093,localhost:9094
```

```
hadoop@21377062-VirtualBox:/usr/local/kafka$ cd /usr/local/kafka
hadoop@21377062-VirtualBox:/usr/local/kafka$ ./bin/kafka-topics.sh --describe --topic comments --bootstrap-ser
ver localhost:9092,localhost:9093,localhost:9094
Topic: comments TopicId: 0X_GRQGJT26AzivLznVgNQ PartitionCount: 3       ReplicationFactor: 2    Configs:
        Topic: comments Partition: 0    Leader: 1       Replicas: 1,2   Isr: 1,2
        Topic: comments Partition: 1    Leader: 0       Replicas: 0,1   Isr: 0,1
        Topic: comments Partition: 2    Leader: 2       Replicas: 2,0   Isr: 2,0
h Terminal 377062-VirtualBox:/usr/local/kafka$ ./bin/kafka-topics.sh --describe --topic likes,shares --bootstrap
-server localhost:9092,localhost:9093,localhost:9094
Topic: shares   TopicId: onUDO_79QCOEib81r2LZCA PartitionCount: 3       ReplicationFactor: 2    Configs:
        Topic: shares   Partition: 0    Leader: 0       Replicas: 0,2   Isr: 0,2
        Topic: shares   Partition: 1    Leader: 2       Replicas: 2,1   Isr: 2,1
        Topic: shares   Partition: 2    Leader: 1       Replicas: 1,0   Isr: 1,0
Topic: likes    TopicId: Ar2LUKXiR6KAOjdyZmBG4w PartitionCount: 3       ReplicationFactor: 2    Configs:
        Topic: likes    Partition: 0    Leader: 0       Replicas: 0,1   Isr: 0,1
        Topic: likes    Partition: 1    Leader: 2       Replicas: 2,0   Isr: 2,0
        Topic: likes    Partition: 2    Leader: 1       Replicas: 1,2   Isr: 1,2
```

2.2 项目创建与环境配置

创建java项目，导入kafka/libs下的所有jar包，并创建3个文件（SocialMediaApp.class, SocialMediaConsumer.class, SocialMediaProducter.class)



2.3 程序编写

SocialMediaConsumer.class

```
1  import com.fasterxml.jackson.databind.ObjectMapper;
2  import com.fasterxml.jackson.databind.node.ObjectNode;
3  import java.io.FileWriter;
4  import java.util.Arrays;
5  import java.util.HashMap;
6  import java.util.HashSet;
7  import java.util.Iterator;
8  import java.util.Map;
9  import java.util.Properties;
10 import java.util.Set;
11 import org.apache.kafka.clients.consumer.ConsumerRecord;
12 import org.apache.kafka.clients.consumer.ConsumerRecords;
13 import org.apache.kafka.clients.consumer.KafkaConsumer;
14 import org.apache.kafka.common.serialization.StringDeserializer;
15
16 public class SocialMediaConsumer {
```

```
17      private static final ObjectMapper mapper = new ObjectMapper();
18
19      public SocialMediaConsumer() {
20      }
21
22      public static void main(String[] args) {
23          Properties props = new Properties();
24          props.put("bootstrap.servers",
    "localhost:9092,localhost:9093,localhost:9094");
25          props.put("group.id", "social-media-group");
26          props.put("key.deserializer",
    StringDeserializer.class.getName());
27          props.put("value.deserializer",
    StringDeserializer.class.getName());
28          props.put("auto.offset.reset", "earliest");
29          KafkaConsumer<String, String> consumer = new
    KafkaConsumer(props);
30          consumer.subscribe(Arrays.asList("likes", "comments",
    "shares"));
31          Map<String, Set<String>> userComments = new HashMap();
32          Map<String, Map<String, Integer>> userLikes = new
    HashMap();
33          Map<String, Integer> userPopularity = new HashMap();
34          int giveUp = true;
35          int noRecordsCount = 0;
36
37          try {
38              while(true) {
39                  ConsumerRecords<String, String> records =
    consumer.poll(100L);
40                  if (records.count() == 0) {
41                      noRecordsCount += 100;
42                      if (noRecordsCount > 10000) {
43                          return;
44                      }
45                  } else {
46                      noRecordsCount = 0;
47                      System.out.println("Received " +
    records.count() + " records");
48                  }
49
50                  Iterator var10 = records.iterator();
```

```
51
52                 while(var10.hasNext()) {
53                     ConsumerRecord<String, String> record =
(ConsumerRecord)var10.next();
54                     processRecord(record, userComments, userLikes,
userPopularity);
55                 }
56
57                 if (!records.isEmpty()) {
58                     try {
59                         writeToJsonFiles(userComments, userLikes,
userPopularity);
60                     } catch (Exception var14) {
61                     }
62                 }
63             }
64         } finally {
65             consumer.close();
66             System.out.println("Consumer closed");
67         }
68     }
69
70     private static void processRecord(ConsumerRecord<String,
String> record, Map<String, Set<String>> userComments, Map<String,
Map<String, Integer>> userLikes, Map<String, Integer>
userPopularity) {
71         System.out.println((String)record.value());
72         String[] parts = ((String)record.value()).split(" ");
73         String userWhoPosted = parts[1];
74         String postId = parts[2];
75         System.out.println(topic);
76         switch (topic) {
77             case "shares":
78                 int shareCount = parts.length - 3;
79                 userPopularity.merge(userWhoPosted, 20 *
shareCount, Integer::sum);
80                 break;
81             case "comments":
82                 String comment = parts[3];
83                 ((Set)userComments.computeIfAbsent(userWhoPosted,
(k) -> {
84                     return new HashSet();
```

```
85                  })).add(comment);
86                  userPopularity.merge(userWhoPosted, 5,
    Integer::sum);
87                  break;
88              case "likes":
89                  ((Map)userLikes.computeIfAbsent(userWhoPosted, (k)
    -> {
90                      return new HashMap();
91                  })).put(postId, (Integer)
    ((Map)userLikes.getOrDefault(userWhoPosted, new
    HashMap())).getOrDefault(postId, 0) + 1);
92                  userPopularity.merge(userWhoPosted, 1,
    Integer::sum);
93          }
94
95      }
96
97      private static void writeToJsonFiles(Map<String, Set<String>>
    userComments, Map<String, Map<String, Integer>> userLikes,
    Map<String, Integer> userPopularity) throws Exception {
98          Throwable var3 = null;
99          Object var4 = null;
100
101         try {
102             FileWriter commentsWriter = new
    FileWriter("comments.json");
103
104             try {
105                 FileWriter likesWriter = new
    FileWriter("likes.json");
106
107                 try {
108                     FileWriter popularityWriter = new
    FileWriter("popularity.json");
109
110                     try {
111                         mapper.writeValue(commentsWriter,
    userComments);
112                         mapper.writeValue(likesWriter, userLikes);
113                         ObjectNode popularityJson =
    mapper.createObjectNode();
```

```
114                         userPopularity.forEach((user, popularity) -
    > {
115                             popularityJson.put(user,
    (double)popularity / 1000.0);
116                         });
117                         mapper.writeValue(popularityWriter,
    popularityJson);
118                     } finally {
119                         if (popularityWriter != null) {
120                             popularityWriter.close();
121                         }
122 
123                     }
124                 } catch (Throwable var26) {
125                     if (var3 == null) {
126                         var3 = var26;
127                     } else if (var3 != var26) {
128                         var3.addSuppressed(var26);
129                     }
130 
131                     if (likesWriter != null) {
132                         likesWriter.close();
133                     }
134 
135                     throw var3;
136                 }
137 
138                 if (likesWriter != null) {
139                     likesWriter.close();
140                 }
141             } catch (Throwable var27) {
142                 if (var3 == null) {
143                     var3 = var27;
144                 } else if (var3 != var27) {
145                     var3.addSuppressed(var27);
146                 }
147 
148                 if (commentsWriter != null) {
149                     commentsWriter.close();
150                 }
151 
152                 throw var3;
```

```
            }

            if (commentsWriter != null) {
                commentsWriter.close();
            }

        } catch (Throwable var28) {
            if (var3 == null) {
                var3 = var28;
            } else if (var3 != var28) {
                var3.addSuppressed(var28);
            }

            throw var3;
        }
    }
}
```

SocialMediaProducter.class

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Properties;
import org.apache.kafka.clients.producer.Callback;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.clients.producer.RecordMetadata;

public class SocialMediaProducter {
    public SocialMediaProducter() {
    }

    public static void main(String[] args) {
        String inputFile =
"/home/hadoop/Downloads/datasets/student_dataset.txt";
        Properties props = new Properties();
        props.put("bootstrap.servers",
"localhost:9092,localhost:9093,localhost:9094");
```

```
        props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        Producer<String, String> producer = new
KafkaProducer(props);

        try {
            Throwable var4 = null;
            Object var5 = null;

            try {
                BufferedReader br = new BufferedReader(new
FileReader(inputFile));

                try {
                    while(true) {
                        String line;
                        while((line = br.readLine()) != null) {
                            String[] parts = line.split(" ", 2);
                            if (parts.length < 2) {
                                System.err.println("Skipping
malformed line: " + line);
                            } else {
                                String type;
                                label295: {
                                    type = parts[0].toLowerCase();
                                    final String message =
parts[1];

                                    final String topic;
                                    switch (type.hashCode()) {
                                        case 3321751:
                                            if
(!type.equals("like")) {
                                                break label295;
                                            }

                                            topic = "likes";
                                            break;
                                        case 109400031:
                                            if
(!type.equals("share")) {
```

```
                                                    break label295;
                                                }

                                                topic = "shares";
                                                break;
                                            case 950398559:
                                                if
(type.equals("comment")) {
                                                    topic = "comments";
                                                    break;
                                                }
                                            default:
                                                break label295;
                                        }

                                        producer.send(new
ProducerRecord(topic, (Object)null, message), new Callback() {
                                            public void
onCompletion(RecordMetadata metadata, Exception exception) {
                                                if (exception != null)
{

 System.err.println("Failed to send message: " + message + " to
topic: " + topic);

 exception.printStackTrace();
                                                } else {

 System.out.println("Sent message: " + message + " to topic: " +
topic);
                                                }

                                            }
                                        });
                                        continue;
                                    }

                                    System.err.println("Unknown type: "
+ type + " in line: " + line);
                                }
                            }

```

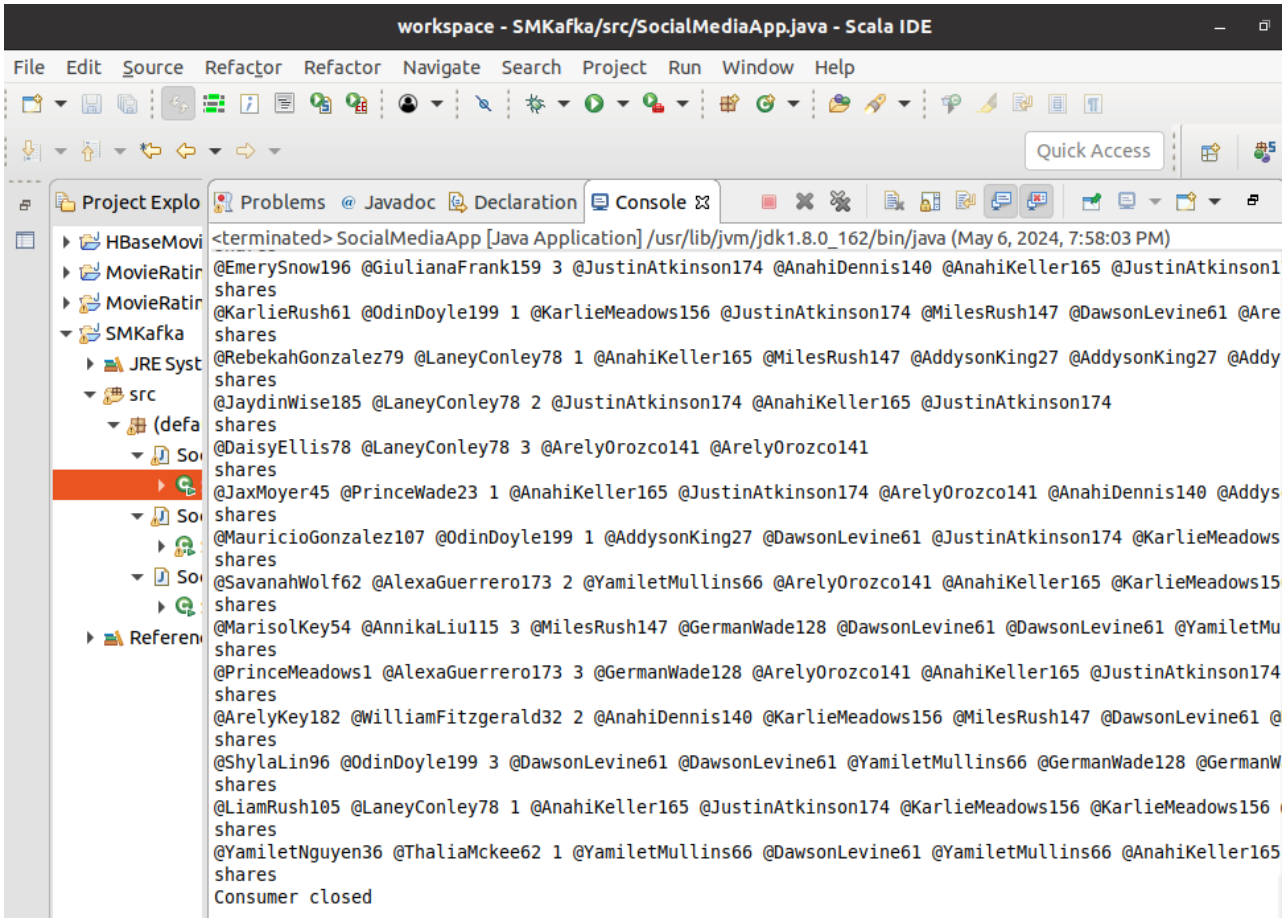
```
                        return;
                    }
                } finally {
                    if (br != null) {
                        br.close();
                    }

                }
            } catch (Throwable var27) {
                if (var4 == null) {
                    var4 = var27;
                } else if (var4 != var27) {
                    var4.addSuppressed(var27);
                }

                throw var4;
            }
        } catch (IOException var28) {
            var28.printStackTrace();
        } finally {
            producer.close();
        }

    }
}
```

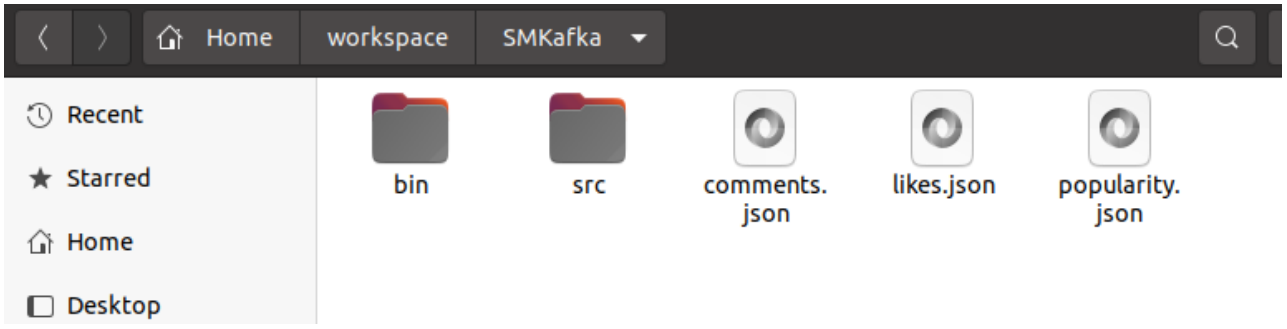
SocialMediaApp.class

```
import java.io.IOException;

public class SocialMediaApp {
    public static void main(String[] args) {
        Thread consumerThread = new Thread(() -> {
            SocialMediaRedisConsumer.main(new String[]{});
        });

        Thread producerThread = new Thread(() -> {
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
```

```
            }
            SocialMediaProducer.main(new String[]{});
        });

        consumerThread.start();
        producerThread.start();
    }
}
```

2.4 运行结果





3. Java API数据统计到Redis

3.1 安装配置Redis数据库

```
sudo apt-get install redis-server
```

```
Setting up libjemalloc2:amd64 (5.2.1-1ubuntu1) ...
Setting up lua-cjson:amd64 (2.1.0+dfsg-2.1) ...
Setting up lua-bitop:amd64 (1.0.2-5) ...
Setting up liblua5.1-0:amd64 (5.1.5-8.1build4) ...
Setting up libhiredis0.14:amd64 (0.14.0-6) ...
Setting up redis-tools (5:5.0.7-2ubuntu0.1) ...
Setting up redis-server (5:5.0.7-2ubuntu0.1) ...
Created symlink /etc/systemd/system/redis.service → /lib/systemd/system/redis-server.service.
Created symlink /etc/systemd/system/multi-user.target.wants/redis-server.service → /lib/systemd/system/
erver.service.
Processing triggers for systemd (245.4-4ubuntu3.23) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.14) ...
```

查看redis是否正常启动

```
service redis-server status
```

```
hadoop@21377062-VirtualBox:~$ service redis-server status
● redis-server.service - Advanced key-value store
     Loaded: loaded (/lib/systemd/system/redis-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-05-06 19:01:56 CST; 1h 5min ago
       Docs: http://redis.io/documentation,
             man:redis-server(1)
   Main PID: 33265 (redis-server)
      Tasks: 4 (limit: 9429)
     Memory: 1.8M
     CGroup: /system.slice/redis-server.service
             └─33265 /usr/bin/redis-server 127.0.0.1:6379
Show Applications
```

3.2 程序编写

SocialMediaRedisConsumer.class

```
import redis.clients.jedis.Jedis;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.common.serialization.StringDeserializer;

import java.util.*;

public class SocialMediaRedisConsumer {
```

```
    private static Jedis jedis;

    public static void main(String[] args) {
        jedis = new Jedis("localhost", 6379);
        System.out.println("Connected to Redis");

        Properties props = new Properties();
        props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
"localhost:9092,localhost:9093,localhost:9094");
        props.put(ConsumerConfig.GROUP_ID_CONFIG, "social-media-
group");
        props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
        props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
        props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,
"earliest");

        KafkaConsumer<String, String> consumer = new KafkaConsumer<>
(props);
        consumer.subscribe(Arrays.asList("likes", "comments",
"shares"));

        final int giveUp = 10000; // 10 seconds timeout
        int noRecordsCount = 0;

        try {
            while (true) {
                ConsumerRecords<String, String> records =
consumer.poll(100);
                if (records.count() == 0) {
                    noRecordsCount += 100;
                    if (noRecordsCount > giveUp) break;
                } else {
                    noRecordsCount = 0;
                    System.out.println("Received " + records.count()
+ " records");
                }

                for (ConsumerRecord<String, String> record :
records) {
                    processRecord(record);
```

```
                }
            }
        } finally {
            consumer.close();
            jedis.close();
            System.out.println("Consumer and Redis client closed");
        }
    }

    private static void processRecord(ConsumerRecord<String, String>
record) {
        String[] parts = record.value().split(" ");
        String topic = record.topic();
        String userWhoPosted = parts[1];
        String postId = parts[2];

        switch (topic) {
            case "likes":
                String likesKey = "likes:" + userWhoPosted + ":" +
postId;
                jedis.hincrBy(userWhoPosted, postId, 1);
                break;
            case "comments":
                String comment = parts[3];
                jedis.rpush("comments:" + userWhoPosted, comment);
                break;
            case "shares":
                int shareCount = parts.length - 3;
                jedis.incrBy("popularity:" + userWhoPosted, 20 *
shareCount);
                break;
        }
    }
}
```

3.3 运行结果

```
<terminated> SocialMediaApp [Java Application] /usr/lib/jvm/jdk1.8.0_162/bin/java (May 6, 2024, 8:10:49 PM)
Sent message: @MarisolKey54 @AnnikaLiu115 3 "vsLiWl " to topic: comments
Sent message: @PrinceMeadows1 @AlexaGuerrero173 3 "oQJhHUGDUwCZVoFCTLWtnpTq TixwSnUYnllBFBAImmIO UIQvYoKmPql HbIRQGQeAMSwqWIX
Sent message: @ArelyKey182 @WilliamFitzgerald32 2 "HdKgygfvxnarNTk KDTSJIaIAyhNGKw vNLzLTzamyNYtygyvFfBtbrR " to topic: comme
Sent message: @ShylaLin96 @OdinDoyle199 3 "bbkqfKHhOrp MNFGtMBHWdkHqrASZEIYNFMv iMmvSDcpGgWOL " to topic: comments
Sent message: @LiamRush105 @LaneyConley78 1 "rwcBRyUWxOZqptM NIWMJbzX OkvOs jvdvVcLMrLIAwXZ " to topic: comments
Sent message: @YamiletNguyen36 @ThaliaMckee62 1 "ldGBEEBMreowPLet nIgxLtFbLyCAxgf tYjMMKXgaIpqmfRjRxHhunR JDLefmrQGUxg " to t
Sent message: @DelaneyMorris61 @LaneyConley78 2 to topic: likes
Sent message: @VeronicaBruce37 @AlexaGuerrero173 3 to topic: likes
Sent message: @EnriqueMorrow21 @PrinceKing174 3 to topic: likes
Sent message: @LiamStephenson62 @AlexaGreen177 1 to topic: likes
Sent message: @AzulDeleon137 @OdinDoyle199 1 to topic: likes
Sent message: @GradyMckee131 @PrinceKing174 1 to topic: likes
Sent message: @JaydinRussell77 @WilliamFitzgerald32 1 to topic: likes
Sent message: @RhysBrooks25 @PrinceWade23 2 to topic: likes
Sent message: @MosheHuynh12 @OdinDoyle199 3 to topic: likes
Sent message: @GiselleMeadows58 @AlexaGreen177 2 to topic: likes
Sent message: @MosheAtkinson53 @AlexaGuerrero173 3 to topic: likes
Sent message: @DominickLevine117 @WilliamFitzgerald32 1 to topic: likes
Sent message: @KaleyStevenson112 @OdinDoyle199 2 to topic: likes
Sent message: @DannaKey152 @AlexaGreen177 3 to topic: likes
Sent message: @EmerySnow196 @GiulianaFrank159 3 to topic: likes
Sent message: @KarlieRush61 @OdinDoyle199 1 to topic: likes
Sent message: @RebekahGonzalez79 @LaneyConley78 1 to topic: likes
Sent message: @JaydinWise185 @LaneyConley78 2 to topic: likes
Sent message: @DaisyEllis78 @LaneyConley78 3 to topic: likes
Sent message: @JaxMoyer45 @PrinceWade23 1 to topic: likes
Sent message: @MauricioGonzalez107 @OdinDoyle199 1 to topic: likes
Sent message: @SavanahWolf62 @AlexaGuerrero173 2 to topic: likes
Sent message: @MarisolKey54 @AnnikaLiu115 3 to topic: likes
Sent message: @PrinceMeadows1 @AlexaGuerrero173 3 to topic: likes
Sent message: @ArelyKey182 @WilliamFitzgerald32 2 to topic: likes
Sent message: @ShylaLin96 @OdinDoyle199 3 to topic: likes
Sent message: @LiamRush105 @LaneyConley78 1 to topic: likes
Sent message: @YamiletNguyen36 @ThaliaMckee62 1 to topic: likes
Received 8 records
Received 99 records
Consumer and Redis client closed
```

查看运行结果

```
redis-cli
KEYS *
```

```
hadoop@21377062-VirtualBox:~/workspace$ cd SocialMediaKa
hadoop@21377062-VirtualBox:~/workspace/SocialMediaKa$ redis-cli
127.0.0.1:6379> KEYS *
 1) "@ThaliaMckee62"
 2) "@LaneyConley78"
 3) "popularity:@PrinceKing174"
 4) "popularity:@LaneyConley78"
 5) "comments:@OdinDoyle199"
 6) "popularity:@OdinDoyle199"
 7) "comments:@AlexaGuerrero173"
 8) "comments:@WilliamFitzgerald32"
 9) "comments:@AlexaGreen177"
10) "@WilliamFitzgerald32"
11) "@PrinceKing174"
12) "popularity:@AlexaGreen177"
13) "@OdinDoyle199"
14) "popularity:@PrinceWade23"
15) "@PrinceWade23"
16) "popularity:@ThaliaMckee62"
17) "popularity:@AlexaGuerrero173"
18) "comments:@ThaliaMckee62"
19) "comments:@AnnikaLiu115"
20) "@AlexaGuerrero173"
21) "@GiulianaFrank159"
22) "comments:@LaneyConley78"
23) "@AlexaGreen177"
24) "popularity:@GiulianaFrank159"
25) "comments:@GiulianaFrank159"
26) "popularity:@AnnikaLiu115"
27) "popularity:@WilliamFitzgerald32"
28) "comments:@PrinceWade23"
29) "comments:@PrinceKing174"
30) "@AnnikaLiu115"
```

4. 问题及解决方案

4.1 启动Kafka集群时报错

检查发现配置文件时未修改broker id，但修改完成后仍然报错：

```
[2024-05-06 14:31:49,712] ERROR Exiting Kafka due to fatal exception during startup. (kafka.Kafka$)
java.lang.RuntimeException: Stored node id 1 doesn't match previous node id 3 in /tmp/kafka-logs2/meta.propert
ies. If you moved your data, make sure your configured node id matches. If you intend to create a new node, yo
u should remove all data in your data directories.
        at org.apache.kafka.metadata.properties.MetaPropertiesEnsemble.verify(MetaPropertiesEnsemble.java:526)
        at kafka.server.KafkaServer.startup(KafkaServer.scala:250)
        at kafka.Kafka$.main(Kafka.scala:112)
        at kafka.Kafka.main(Kafka.scala)
[2024-05-06 14:31:49,715] INFO shutting down (kafka.server.KafkaServer)
```

查询日志文件的存储位置：

```
# A comma separated list of directories under which to store log files
log.dirs=/tmp/kafka-logs2
```

```
hadoop@21377062-VirtualBox:/usr/local/kafka/bin$ sudo find / -name "kafka-logs2"
[sudo] password for hadoop:
find: ‘/run/user/1001/gvfs’: Permission denied
/tmp/kafka-logs2
hadoop@21377062-VirtualBox:/usr/local/kafka/bin$ cd /tmp/kafka-logs2
hadoop@21377062-VirtualBox:/tmp/kafka-logs2$ ll
total 24
drwxrwxr-x  2 hadoop hadoop 4096 5月   6 14:28 ./
drwxrwxrwt 22 root   root   4096 5月   6 14:28 ../
-rw-rw-r--  1 hadoop hadoop    0 5月   6 14:28 cleaner-offset-checkpoint
-rw-rw-r--  1 hadoop hadoop   30 5月   6 14:28 .kafka_cleanshutdown
-rw-rw-r--  1 hadoop hadoop    0 5月   6 14:28 .lock
-rw-rw-r--  1 hadoop hadoop    4 5月   6 14:28 log-start-offset-checkpoint
-rw-rw-r--  1 hadoop hadoop   88 5月   6 14:28 meta.properties
-rw-rw-r--  1 hadoop hadoop    4 5月   6 14:28 recovery-point-offset-checkpoint
-rw-rw-r--  1 hadoop hadoop    0 5月   6 14:28 replication-offset-checkpoint
hadoop@21377062-VirtualBox:/tmp/kafka-logs2$ vim meta.properties
```

编辑meta.properties的broker id，与配置文件一致后，成功解决报错。