“大数据工程”课程实验报告

题目：Kafka 综合编程实践	学号姓名：21377061 范春	日期：2024.05.06

实验环境：
虚拟机软件：VMvare
编程语言：java

实验内容与完成情况：
下载并解压：

```
hadoop@u-virtual-machine:~/下载$ tar -xvzf kafka_2.12-2.4.1.tgz -C ~/
kafka_2.12-2.4.1/
kafka_2.12-2.4.1/LICENSE
kafka_2.12-2.4.1/NOTICE
kafka_2.12-2.4.1/bin/
kafka_2.12-2.4.1/bin/kafka-delete-records.sh
kafka_2.12-2.4.1/bin/trogdor.sh
kafka_2.12-2.4.1/bin/kafka-preferred-replica-election.sh
kafka_2.12-2.4.1/bin/connect-mirror-maker.sh
kafka_2.12-2.4.1/bin/kafka-console-consumer.sh
```

伪分布式配制：

```
hadoop@u-virtual-machine:~/下载/kafka_2.12-2.4.1/config$ ls
connect-console-sink.properties     log4j.properties
connect-console-source.properties   producer.properties
connect-distributed.properties      server-1.properties
connect-file-sink.properties        server-2.properties
connect-file-source.properties      server.properties
connect-log4j.properties            tools-log4j.properties
connect-mirror-maker.properties     trogdor.conf
connect-standalone.properties       zookeeper.properties
consumer.properties
```

启动 zookeeper

```
hadoop@u-virtual-machine:~/下载/kafka_2.12-2.4.1$ bin/zookeeper-server-sta
rt.sh config/zookeeper.properties
[2024-05-03 15:03:38,914] INFO Reading configuration from: config/zookeepe
r.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-05-03 15:03:38,919] WARN config/zookeeper.properties is relative. Pr
epend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum
.QuorumPeerConfig)
```

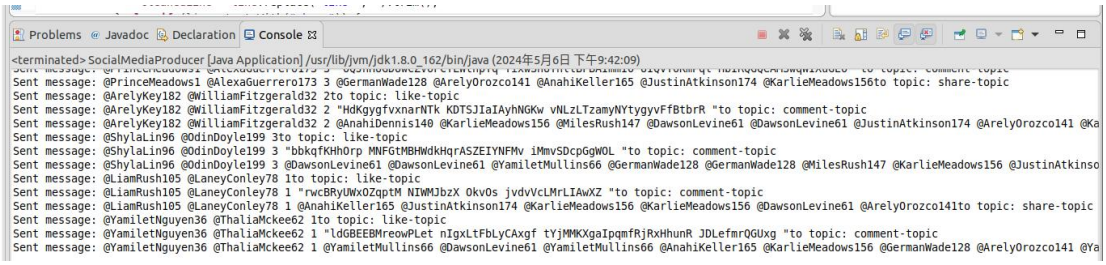
启动三个 Kafka 服务器

```
[2024-05-03 15:06:06,817] INFO Kafka startTimeMs: 1714719966804 (org.apach
e.kafka.common.utils.AppInfoParser)
[2024-05-03 15:06:06,821] INFO [KafkaServer id=0] started (kafka.server.Ka
fkaServer)
```

```
[2024-05-03 15:06:50,950] INFO Kafka startTimeMs: 1714720010944 (org.apach
e.kafka.common.utils.AppInfoParser)
[2024-05-03 15:06:50,953] INFO [KafkaServer id=1] started (kafka.server.Ka
fkaServer)
```

```
[2024-05-03 15:07:15,025] INFO Kafka startTimeMs: 1714720035017 (org.apach
e.kafka.common.utils.AppInfoParser)
[2024-05-03 15:07:15,027] INFO [KafkaServer id=2] started (kafka.server.Ka
fkaServer)
```

问题 1：
生产数据：



代码如下：

```
    package kafka;


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


    import java.io.*;
    import java.util.Properties;


    public class SocialMediaProducer {
        public static void main(String[] args) throws IOException {
            Properties props = new Properties();
            props.put("bootstrap.servers", "localhost:9092");
            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);
            BufferedReader reader = new BufferedReader(new
FileReader("/home/hadoop/桌面/dataset/student_dataset.txt"));
            String line;
            while ((line = reader.readLine()) != null) {
                String topic;
                String cleanedLine;
```

```
                if (line.startsWith("like")) {
                    topic = "like-topic";
                    cleanedLine = line.replace("like ","").trim();
                } else if (line.startsWith("share")) {
                    topic = "share-topic";
                    cleanedLine = line.replace("share ","").trim();
                } else if (line.startsWith("comment")) {
                    topic = "comment-topic";
                    cleanedLine = line.replace("comment ","").trim();
                } else {
                    continue;
                }
                producer.send(new ProducerRecord<>(topic, cleanedLine));
                System.out.println("Sent message: "+cleanedLine+"to topic:
"+topic);
            }
            producer.close();
            reader.close();
        }
}
```

处理数据并保存为 json 文件：

代码如下：

```
package kafka;

import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.time.Duration;

import java.io.FileWriter;
import java.io.IOException;
import java.util.*;

public class SocialMediaConsumer {
    public static void main(String[] args) throws IOException {
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        props.put("group.id", "consumer-group");
        props.put("auto.offset.reset", "earliest");

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

        consumer.subscribe(Arrays.asList("like-topic", "share-topic",
"comment-topic"));

        Map<String, List<String>> comments = new HashMap<>();
        Map<String, Integer> likes = new HashMap<>();
        Map<String, Set<String>> shares = new HashMap<>();
        Map<String, Double> popularity = new HashMap<>();

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

        try {
            while (true) {
                ConsumerRecords<String, String> records =
consumer.poll(Duration.ofMillis(1000));

                if (records.count() == 0) {
```

```
                noRecordsCount += 100; // poll timeout duration
                if (noRecordsCount > giveUp) break;
            } else {
                noRecordsCount = 0; // reset timeout counter
                System.out.println("Received " + records.count() + "
records");
            }

            for (ConsumerRecord<String, String> record : records) {
                String[] parts = record.value().split(" ");
                String postId = parts[1] + parts[2]; // Combine user ID and
post ID for uniqueness
                switch (record.topic()) {
                    case "comment-topic":
                        comments.computeIfAbsent(postId, k -> new
ArrayList<>()).add(record.value());
                        break;
                    case "like-topic":
                        likes.merge(postId, 1, Integer::sum);
                        break;
                    case "share-topic":
                        shares.computeIfAbsent(postId, k -> new
HashSet<>()).addAll(Arrays.asList(parts).subList(3, parts.length));
                        break;
                }
            }

            // Calculation of popularity
            popularity.forEach((key, value) -> {
                int likeCount = likes.getOrDefault(key, 0);
                int shareCount = shares.getOrDefault(key,
Collections.emptySet()).size();
                int commentCount = comments.getOrDefault(key,
Collections.emptyList()).size();
                popularity.put(key, (likeCount + 20 * shareCount + 5 *
commentCount) / 1000.0);
            });

            // Save data to JSON
            ObjectMapper mapper = new ObjectMapper();
            mapper.writeValue(new FileWriter("/home/hadoop/桌面
/comments.json"), comments);
            mapper.writeValue(new FileWriter("/home/hadoop/桌面
/likes.json"), likes);
```

```
                    mapper.writeValue(new FileWriter("/home/hadoop/桌面/popularity.json"), popularity);

                    // Optionally break after processing, depends on use case
                    break;
                }
            } finally {
                consumer.close();
                System.out.println("end");
            }
        }
}
```

问题 2：

配制 redis 数据库

```
hadoop@u-virtual-machine:~/kafka_2.12-2.4.1$ sudo apt-get install redis-server
[sudo] hadoop 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会同时安装下列软件：
  libhiredis0.14 libjemalloc2 liblua5.1-0 lua-bitop lua-cjson redis-tools
建议安装：
  ruby-redis
下列【新】软件包将被安装：
  libhiredis0.14 libjemalloc2 liblua5.1-0 lua-bitop lua-cjson redis-server redis-tools
升级了 0 个软件包，新安装了 7 个软件包，要卸载 0 个软件包，有 61 个软件包未被升级。
需要下载 915 kB 的归档。
解压缩后会消耗 4,077 kB 的额外空间。
您希望继续执行吗？ [Y/n] y
```

```
hadoop@u-virtual-machine:~/kafka_2.12-2.4.1$ 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 23:17:02 CST; 18s ago
       Docs: http://redis.io/documentation,
             man:redis-server(1)
   Main PID: 13190 (redis-server)
      Tasks: 4 (limit: 4556)
     Memory: 1.9M
     CGroup: /system.slice/redis-server.service
             └─13190 /usr/bin/redis-server 127.0.0.1:6379
```

代码如下：

```
package kafka;

import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import redis.clients.jedis.Jedis;

import java.time.Duration;
import java.util.*;

public class SocialMediaRedisConsumer {
    public static void main(String[] args) {
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("key.deserializer",
```

```
"org.apache.kafka.common.serialization.StringDeserializer");
        props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        props.put("group.id", "redis-consumer-group");
        props.put("auto.offset.reset", "earliest");

        KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props);
        consumer.subscribe(Arrays.asList("like-topic", "share-topic", "comment-topic"));


        Jedis jedis = new Jedis("localhost", 6379);

        Map<String, List<String>> comments = new HashMap<>();
        Map<String, Integer> likes = new HashMap<>();
        Map<String, Set<String>> shares = new HashMap<>();
        Map<String, Double> popularity = new HashMap<>();

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

        try {
            while (true) {
                ConsumerRecords<String, String> records =
consumer.poll(Duration.ofMillis(1000));
                if (records.count() == 0) {
                    noRecordsCount += 100;
                    if (noRecordsCount > giveUp) break;
                } else {
                    noRecordsCount = 0; // reset timeout counter
                    System.out.println("Received " + records.count() + " records");
                }

                for (ConsumerRecord<String, String> record : records) {
                    String[] parts = record.value().split(" ");
                    String postId = parts[1] + parts[2];

                    switch (record.topic()) {
                        case "comment-topic":
                            comments.computeIfAbsent(postId, k -> new ArrayList<>())
                                    .add(record.value());
                            break;
                        case "like-topic":
                            likes.merge(postId, 1, Integer::sum);
                            break;
```

```
                                case "share-topic":
                                    shares.computeIfAbsent(postId, k -> new HashSet<>())
                                            .addAll(Arrays.asList(parts).subList(3,
parts.length));
                                    break;
                            }
                        }

                        // Calculate popularity
                        popularity.forEach((key, value) -> {
                            int likeCount = likes.getOrDefault(key, 0);
                            int shareCount = shares.getOrDefault(key,
Collections.emptySet()).size();
                            int commentCount = comments.getOrDefault(key,
Collections.emptyList()).size();
                            popularity.put(key, (likeCount + 20 * shareCount + 5 *
commentCount) / 1000.0);
                        });

                        // Store data in Redis
                        comments.forEach((key, value) -> {
                            jedis.set("comments:" + key, String.join("\n", value));
                        });

                        likes.forEach((key, value) -> {
                            jedis.set("likes:" + key, String.valueOf(value));
                        });

                        popularity.forEach((key, value) -> {
                            jedis.set("popularity:" + key, String.valueOf(value));
                        });

                        // Optionally, break after processing
                        break;
                    }
                } finally {
                    consumer.close();
                    jedis.close(); // Make sure to close the Redis connection
                }
            }
        }
```

运行结果如下：

```
127.0.0.1:6379> KEYS *
 1) "likes:@LaneyConley782"
 2) "comments:@PrinceWade232"
 3) "likes:@WilliamFitzgerald322"
 4) "likes:@PrinceKing1741"
 5) "comments:@PrinceKing1741"
 6) "comments:@LaneyConley781"
 7) "comments:@PrinceKing1743"
 8) "comments:@OdinDoyle1992"
 9) "comments:@AlexaGreen1771"
10) "likes:@LaneyConley781"
11) "likes:@AlexaGuerrero1733"
12) "likes:@LaneyConley783"
13) "likes:@PrinceKing1743"
14) "comments:@GiulianaFrank1591"
15) "likes:@GiulianaFrank1592"
16) "comments:@LaneyConley782"
17) "likes:@GiulianaFrank1591"
18) "likes:@AlexaGreen1773"
19) "likes:@AlexaGuerrero1731"
20) "comments:@OdinDoyle1993"
21) "likes:@PrinceWade231"
22) "likes:@GiulianaFrank1593"
23) "comments:@AnnikaLiu1153"
24) "comments:@ThaliaMckee621"
25) "comments:@GiulianaFrank1593"
26) "likes:@AlexaGreen1772"
27) "likes:@AnnikaLiu1153"
28) "likes:@PrinceKing1742"
29) "likes:@WilliamFitzgerald321"
30) "comments:@AlexaGuerrero1732"
31) "comments:@GiulianaFrank1592"
32) "comments:@PrinceKing1742"
33) "comments:@WilliamFitzgerald321"
34) "likes:@OdinDoyle1991"
35) "likes:@AlexaGuerrero1732"
36) "likes:@ThaliaMckee621"
37) "likes:@OdinDoyle1993"
38) "likes:@OdinDoyle1992"
39) "likes:@PrinceWade232"
```