Please visit website: http://cxyroad.com

在物联网项目中Flink实时计算设备上报数据的应用
==========

1. 业务背景
=======

物联网项目中，设备会上报的实时数据（例如心率、呼吸率），这些数据IOT平台接收之后会发送到Kafka。然后一般物联网项目中会存在大数据计算系统，大数据计算系统订阅接收到设备上报的数据（我们的设备上报的数据是心率、呼吸率等数据）之后，会进行`流式计算各个设备每5分钟的心率、呼吸率的平均值`，然后发送到Kafka，业务后端订阅获取设备心率呼吸率平均值落库保存。

大数据系统在订阅接收到设备上报的数据还需要每10分钟发送设备上报的N条心率呼吸率数据给算法平台，算法平台会根据这些数据计算识别出睡眠状态（清醒状态、浅睡状态、深睡状态）。

最后`定时调度系统`在第二天早上，调用`算法平台获取返睡眠状态，整合生成初始睡眠报告`发送到Kafka。业务后端从Kafka订阅接收到报告，进一步`处理加工成最终睡眠报告`，展示给用户，如下图。

![大数据计算.png](https://p9-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/b25b772918c348a0b518b862151b0574~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=729&h=469&s=28506&e=png&a=1&b=fefdfd)

所以，接下来，我们需要基于`流式计算生成各个设备5分钟心率、呼吸率的平均值`。

2. 技术选型
=======

Flink可以进行流处理，我们项目要使用流处理计算各个设备5分钟心率、呼吸率的平均值。同时Flink将中间结果存储在内存中，因此性能高。

技术选择	数据是否精确处理	处理性能	函数以及api	吞吐量	延迟
Storm	不能保证仅处理一次，数据会重复处理	一般	一般	一般	一般
Spark Streaming	对数据仅处理一次，数据不会丢失，不会重复处理	一般	一般	一般	秒级
Flink	对数据仅处理一次，数据不会丢失，不会重复处理	高	丰富的函数以及api	高	毫秒级

综上所述，因此我们的大数据选择Flink。

3. 计算心率呼吸率的平均值

1. 创建Flink工程

使用mvn命令创建Flink工程，flink版本为1.9.0

```
mvn archetype:generate -DarchetypeGroupId=org.apache.flink  -DarchetypeArtifactId=flink-quickstart-java -DarchetypeVersion=1.9.0 -DgroupId=testFlink  -DartifactId=testFlink
```

pom中依赖如下：

```
<properties>
    <flink.version>1.9.0</flink.version>
    <scala.binary.version>2.11</scala.binary.version>
</properties>

<!--Flink依赖-->
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-java</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
```

```
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-streaming-java_${scala.binary.version}</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-connector-kafka-
0.10_${scala.binary.version}</artifactId>
    <version>${flink.version}</version>
</dependency>
```

2. Flink数据计算逻辑编写

我们计算最近5分钟心率、呼吸率的平均值。为了便于尽快演示，我们设置计算最近5秒的心率、呼吸率的平均值。

主要流程：

* 设置数据源（从kafka中接收设备数据）
* 按照设备mac地址对设备分组。
* 设置时间窗口（指定最近5秒，即收集各个设备5秒之内的数据，参与后面的窗口计算）。
* 使用窗口函数计算各个设备在窗口期内的设备心率平均值。
* 将计算结果添加到Kafka连接器，从而发送到Kafka
* 启动flink作业。

```
/**
 * @description: 计算心率service
 * @author：xg
 * @date: 2024/5/10
 * @Copyright：
 */
@Component
@Slf4j
public class ComputeHeartRateRunner implements ApplicationRunner {

    @Value(”${kafka.topic.device.data}”)
    private String topicDeviceData;
```

```
    @Value("${kafka.topic.device.avg}")
    private String topicDeviceAvg;

    @Resource
    private KafkaProducer kafkaProducer;


    @Override
    public void run(ApplicationArguments args) throws Exception {

        // 定义kafka相关属性
        Properties kafkaProps = new Properties();
        kafkaProps.setProperty("bootstrap.servers",
"192.168.56.200:9092,192.168.56.202:9092,192.168.56.203:9092");
        kafkaProps.setProperty("group.id", "flink.group.device.data");
        kafkaProps.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        kafkaProps.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        kafkaProps.put("auto.offset.reset", "latest");
        StreamExecutionEnvironment senv =
StreamExecutionEnvironment.getExecutionEnvironment();
        // 设置使用数据的eventTime（即数据的时间）来处理数据
        senv.setStreamTimeCharacteristic(TimeCharacteristic.EventTime);

        // 设置数据源（从kafka中接收设备数据）
        DataStreamSource<String> dataStreamSource = senv
                .addSource(new FlinkKafkaConsumer010<>(topicDeviceData,
new SimpleStringSchema(), kafkaProps));


        // 转换成DeviceData对象
        DataStream<DeviceData> deviceDataStream = dataStreamSource
                .map(string -> JSON.parseObject(string, DeviceData.class));

        // 设置数据时间戳和水位线
        // 水位线作用:解决数据乱序，导致的延时，设置延时一段时间(例如：下
面指定延迟1分钟)接收数据，为了便于演示指定1秒钟，并参与本次窗口计算
        DataStream<DeviceData> marksSource = deviceDataStream
                .assignTimestampsAndWatermarks(
                        new
BoundedOutOfOrdernessTimestampExtractor<DeviceData>(Time.second
s(1)){
                            @Override
                            public long extractTimestamp(DeviceData d) {
                                return d.getDateTime().getTime();
                            }
                        });
```

```
        // 按照设备mac地址分组（按照设备分组，设置窗口之后，得到各个设备在窗口期内产生的N条数据）
        SingleOutputStreamOperator<String> windowedStream = marksSource.keyBy(DeviceData::getMacAdd)
        // 时间窗口设置为不重叠窗口
        .window(TumblingEventTimeWindows.of(Time.seconds(5)))
        // 使用窗口函数计算设备心率平均值
        .process(new HeartRateAverageProcessFunction());

        // 创建Kafka连接器
        Properties properties = new Properties();
properties.setProperty(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "192.168.56.200:9092,192.168.56.202:9092,192.168.56.203:9092");
        FlinkKafkaProducer09<String> kafkaProducer = new FlinkKafkaProducer09<>(
                topicDeviceAvg,
                new SimpleStringSchema(),
                properties);
        // 将计算结果添加到Kafka连接器，从而发送到Kafka
        windowedStream.addSink(kafkaProducer);

        // 执行flink作业
        senv.execute("deviceHeartRateJob");
    }


    @Data
    @AllArgsConstructor
    private static class DeviceAvg {

        private String macAdd;

        private Double heartRateAvg;

    }


    public class HeartRateAverageProcessFunction extends ProcessWindowFunction<DeviceData, String, String, TimeWindow> {
        @Override
        public void process(String macAdd, Context context, Iterable<DeviceData> elements, Collector<String> out) throws Exception {
            int count = 0;
            double sum = 0.0;
            // 针对设备在窗口范围内的N条数据，计算平均心率
            for (DeviceData data : elements) {
```

```
            count++;
            sum += data.getHeartRate();
        }
        double average = sum / count;
        // 封装并输出设备平均心率
        out.collect(new DeviceAvg(macAdd, average).toString());
    }
  }

}
```

当然，我们先要创建topic，不然启动会报错topic不存在。

```
kafka-topics.sh --create --topic device_data_topic --bootstrap-server
192.168.56.200:9092
```

3. 往Kafka中不断的发送设备心率数据

```
/**
 * 往kafka的topic中发布设备心率数据
 */
@Test
public void sendHeartRate() throws InterruptedException {
    for(int i = 0; i<20; i++) {
        // 设备心率
        DeviceData deviceData = new DeviceData();
        deviceData.setMacAdd("m" + RandomUtil.randomInt(1,4));
        deviceData.setHeartRate(RandomUtil.randomInt(0, 100));
        deviceData.setDateTime(new Date());
        kafkaProducer.send(JSON.toJSONString(deviceData),
topicDeviceData);
        Thread.sleep(500);
    }

}
```

4. 业务后端订阅获取设备心率平均值落库保存

```
/**
 * 业务后端订阅获取设备心率平均值落库保存
 * @param record
 * @param ack
 * @param topic
 */
@KafkaListener(topics = ”device_avg_topic”, groupId =
”device_avg_group”)
public void handle2(ConsumerRecord<?, ?> record, Acknowledgment
ack, @Header(KafkaHeaders.RECEIVED_TOPIC) String topic) {
    Optional message = Optional.ofNullable(record.value());
    if (message.isPresent()) {
        Object msg = message.get();
        log.info(”消费者组：{} 从topic:{} 订阅接收处理设备心率平均值:{}”,
”device_avg_group”, topic, msg);
        // TODO 落库保存设备平均心率

        ack.acknowledge();
    }
}

```

最后，我们观察到业务消费端收到的各个设备的心率平均值：

```
2024-05-19 11:43:11.689  INFO 73516 --- [ntainer#1-2-C-1]
o.e.r.d.c.service.KafkaConsumer          : 消费者组：device_avg_group 从
topic:device_avg_topic 订阅接收处理设备心率呼吸率平均值
:ComputeHeartRateRunner.DeviceAvg(macAdd=m3, heartRateAvg=61.5)
2024-05-19 11:43:11.705  INFO 73516 --- [ntainer#1-2-C-1]
o.e.r.d.c.service.KafkaConsumer          : 消费者组：device_avg_group 从
topic:device_avg_topic 订阅接收处理设备心率呼吸率平均值
:ComputeHeartRateRunner.DeviceAvg(macAdd=m1,
heartRateAvg=21.6666666666666668)
2024-05-19 11:43:11.709  INFO 73516 --- [ntainer#1-2-C-1]
o.e.r.d.c.service.KafkaConsumer          : 消费者组：device_avg_group 从
topic:device_avg_topic 订阅接收处理设备心率呼吸率平均值
:ComputeHeartRateRunner.DeviceAvg(macAdd=m2, heartRateAvg=80.5)
2024-05-19 11:43:16.330  INFO 73516 --- [ntainer#1-2-C-1]
o.e.r.d.c.service.KafkaConsumer          : 消费者组：device_avg_group 从
```

```
topic:device_avg_topic 订阅接收处理设备心率呼吸率平均值
:ComputeHeartRateRunner.DeviceAvg(macAdd=m3,
heartRateAvg=59.3333333333333336)
2024-05-19 11:43:16.379  INFO 73516 --- [ntainer#1-2-C-1]
o.e.r.d.c.service.KafkaConsumer          : 消费者组：device_avg_group 从
topic:device_avg_topic 订阅接收处理设备心率呼吸率平均值
:ComputeHeartRateRunner.DeviceAvg(macAdd=m1,
heartRateAvg=64.33333333333333)
2024-05-19 11:43:16.384  INFO 73516 --- [ntainer#1-2-C-1]
o.e.r.d.c.service.KafkaConsumer          : 消费者组：device_avg_group 从
topic:device_avg_topic 订阅接收处理设备心率呼吸率平均值
:ComputeHeartRateRunner.DeviceAvg(macAdd=m2, heartRateAvg=38.25)

```
原文链接: https://juejin.cn/post/7370197993678733362