**赛题03**

**背景描述**

工业互联网是工业全要素、全产业链、全价值链的全面连接，是人、机、物、工厂互联互通的新型工业生产制造服务体系，是互联网从消费领域向生产领域、从虚拟经济向实体经济拓展的核心载体，是建设现代化经济体系、实现高质量发展和塑造全球产业竞争力的关键支撑，工业大数据则是工业互联网实现工业要素互联之后的核心价值创造者｡随着大数据行业的发展,工业数据收集呈现时间维度不断延长、数据范围不断扩大、数据粒度不断细化的趋势｡以上三个维度的变化使得企业所积累的数据量以加速度的方式在增加,最终构成了工业大数据的集合｡

为完成工业大数据分析工作，你所在的小组将应用大数据技术，以Scala作为整个项目的基础开发语言，基于大数据平台综合利用

Hudi、Spark、Flink、Vue.js等技术，对数据进行处理、分析及可视化呈现，你们作为该小组的技术人员，请按照下面任务完成本次工作。

## 任务A：大数据平台搭建（容器环境）（15分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**宿主机可通过Asbru工具或SSH客户端进行SSH访问；  相关软件安装包在宿主机的/opt目录下，请选择对应的安装包进行安装，用不到的可忽略；  所有任务中应用命令必须采用绝对路径；  进入Master节点的方式为  docker exec -it master /bin/bash  进入Slave1节点的方式为  docker exec -it slave1 /bin/bash  进入Slave2节点的方式为  docker exec -it slave2 /bin/bash  三个容器节点的root密码均为123456 |

### 子任务一：Hadoop 完全分布式安装配置

本任务需要使用root用户完成相关配置，安装Hadoop需要配置前置环境。命令中要求使用绝对路径，具体要求如下:

1. 从宿主机/opt目录下将文件hadoop-3.1.3.tar.gz、jdk-8u212-linux-x64.tar.gz复制到容器Master中的/opt/software路径中（若路径不存在，则需新建），将Master节点JDK安装包解压到/opt/module路径中(若路径不存在，则需新建)，将JDK解压命令复制并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
2. 修改容器中/etc/profile文件，设置JDK环境变量并使其生效，配置完毕后在Master节点分别执行“java -version”和“javac”命令，将命令行执行结果分别截图并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
3. 请完成host相关配置，将三个节点分别命名为master、slave1、slave2，并做免密登录，用scp命令并使用绝对路径从Master复制JDK解压后的安装文件到slave1、slave2节点（若路径不存在，则需新建），并配置slave1、slave2相关环境变量，将全部scp复制JDK的命令复制并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
4. 在Master将Hadoop解压到/opt/module(若路径不存在，则需新建)目录下，并将解压包分发至slave1、slave2中，其中master、slave1、slave2节点均作为datanode，配置好相关环境，初始化Hadoop环境namenode，将初始化命令及初始化结果截图（截取初始化结果日志最后20行即可）粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
5. 启动Hadoop集群（包括hdfs和yarn），使用jps命令查看Master节点与slave1节点的Java进程，将jps命令与结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下。

### 子任务二：Spark on Yarn安装配置

本任务需要使用root用户完成相关配置，已安装Hadoop及需要配置前置环境，具体要求如下：

1. 从宿主机/opt目录下将文件spark-3.1.1-bin-hadoop3.2.tgz复制到容器Master中的/opt/software（若路径不存在，则需新建）中，将Spark包解压到/opt/module路径中(若路径不存在，则需新建)，将完整解压命令复制粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
2. 修改容器中/etc/profile文件，设置Spark环境变量并使环境变量生效，在/opt目录下运行命令spark-submit --version，将命令与结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
3. 完成on yarn相关配置，使用spark on yarn 的模式提交$SPARK\_HOME/examples/jars/spark-examples\_2.12-3.1.1.jar 运行的主类为org.apache.spark.examples.SparkPi，将运行结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下（截取Pi结果的前后各5行）。

（运行命令为：spark-submit --master yarn --class org.apache.spark.examples.SparkPi $SPARK\_HOME/examples/jars/spark-examples\_2.12-3.1.1.jar）

### 子任务三：Hudi安装配置

本任务需要使用root用户完成相关配置，具体要求如下：

1. 从宿主机/opt目录下将maven相关安装包复制到容器Master中的/opt/software（若路径不存在，则需新建）中，将maven相关安装包解压到/opt/module/目录下（若路径不存在，则需新建）并配置maven本地库为/opt/software/RepMaven/，远程仓库使用阿里云镜像，配置maven的环境变量，并在/opt/下执行mvn -v，将运行结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；

<mirror>

<id>nexus-aliyun</id>

<mirrorOf>central</mirrorOf>

<name>Nexus aliyun</name>

<url>http://maven.aliyun.com/nexus/content/groups/public</url>

</mirror>

1. 从宿主机/opt目录下将Hudi相关安装包复制到容器Master中的/opt/software（若路径不存在，则需新建）中，将Hudi相关安装包解压到/opt/module/目录下（若路径不存在，则需新建），将命令复制并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
2. 完成解压安装及配置后使用maven对Hudi进行构建（spark3.1,scala-2.12），编译完成后与Spark集成，集成后使用spark-shell操作Hudi，将spark-shell启动使用spark-shell运行下面给到的案例，并将最终查询结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下。

（**提示**：编译需要替换以下内容：

1.将父模块pom.xml替换；

2.hudi-common/src/main/java/org/apache/hudi/common/table/log/block/HoodieParquetDataBlock.java替换；

1. 将packaging/hudi-spark-bundle/pom.xml替换；

3.将packaging/hudi-utilities-bundle/pom.xml替换）

import org.apache.hudi.QuickstartUtils.\_

import scala.collection.JavaConversions.\_

import org.apache.spark.sql.SaveMode.\_

import org.apache.hudi.DataSourceReadOptions.\_

import org.apache.hudi.DataSourceWriteOptions.\_

import org.apache.hudi.config.HoodieWriteConfig.\_

import org.apache.hudi.common.model.HoodieRecord

val tableName = "hudi\_trips\_cow"

val basePath = "file:///tmp/hudi\_trips\_cow"

val dataGen = new DataGenerator

val inserts = convertToStringList(dataGen.generateInserts(10))

val df = spark.read.json(spark.sparkContext.parallelize(inserts, 2))

df.write.format("hudi").

options(getQuickstartWriteConfigs).

option(PRECOMBINE\_FIELD\_OPT\_KEY, "ts").

option(RECORDKEY\_FIELD\_OPT\_KEY, "uuid").

option(PARTITIONPATH\_FIELD\_OPT\_KEY, "partitionpath").

option(TABLE\_NAME, tableName).

mode(Overwrite).

save(basePath)

val tripsSnapshotDF = spark.read.format("hudi").load(basePath + "/\*/\*/\*/\*")

tripsSnapshotDF.createOrReplaceTempView("hudi\_trips\_snapshot")

spark.sql("select fare, begin\_lon, begin\_lat, ts from hudi\_trips\_snapshot where fare > 20.0").show()

## 任务B：离线数据处理（25分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**各节点可通过Asbru工具或SSH客户端进行SSH访问；  主节点MySQL数据库用户名/密码：root/123456（已配置远程连接）；  Spark任务在Yarn上用Client运行，方便观察日志。  若Hudi中无数据，正常抽取 |

### 子任务一：数据抽取

编写Scala代码，使用Spark将MySQL库中表ChangeRecord，BaseMachine，MachineData， ProduceRecord全量抽取到**Hudi**的hudi\_gy\_ods库（路径为/user/hive/warehouse/hudi\_gy\_ods.db）中对应表changerecord，basemachine， machinedata，producerecord中。

1. 抽取MySQL的shtd\_industry库中ChangeRecord表的全量数据进入Hudi的hudi\_gy\_ods库中表changerecord，字段排序、类型不变，分区字段为etldate，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。PRECOMBINE\_FIELD使用ChangeEndTime，ChangeID和ChangeMachineID作为联合主键。使用spark-sql的cli执行select count（\*） from hudi\_gy\_ods.changerecord命令，将spark-sql的cli执行结果分别截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
2. 抽取MySQL的shtd\_industry库中BaseMachine表的全量数据进入Hudi的hudi\_gy\_ods库中表basemachine，字段排序、类型不变，分区字段为etldate，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。PRECOMBINE\_FIELD使用MachineAddDate，BaseMachineID为主键。使用spark-sql的cli执行show partitions hudi\_gy\_ods.basemachine命令，将cli的执行结果分别截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
3. 抽取MySQL的shtd\_industry库中ProduceRecord表的全量数据进入Hudi的hudi\_gy\_ods库中表producerecord，剔除ProducePrgCode字段，其余字段排序、类型不变，分区字段为etldate，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。PRECOMBINE\_FIELD使用ProduceCodeEndTime，ProduceRecordID和ProduceMachineID为联合主键。使用spark-sql的cli执行show partitions hudi\_gy\_ods.producerecord命令，将spark-sql的cli的执行结果分别截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
4. 抽取MySQL的shtd\_industry库中MachineData表的全量数据进入Hudi的hudi\_gy\_ods库中表machinedata，字段排序、类型不变，分区字段为etldate，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。PRECOMBINE\_FIELD使用MachineRecordDate，MachineRecordID为主键。使用spark-sql的cli执行show partitions hudi\_gy\_ods.machinedata命令，将cli的执行结果分别截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下。

### 子任务二：数据清洗

编写Scala代码，使用Spark将hudi\_gy\_ods库中相应表数据全量抽取到**Hudi**的hudi\_gy\_dwd库（路径为/user/hive/warehouse/hudi\_gy\_dwd.db）中对应表中。表中有涉及到timestamp类型的，均要求按照yyyy-MM-dd HH:mm:ss，不记录毫秒数，若原数据中只有年月日，则在时分秒的位置添加00:00:00，添加之后使其符合yyyy-MM-dd HH:mm:ss。

1. 抽取hudi\_gy\_ods库中changerecord的全量数据进入Hudi的hudi\_gy\_dwd库中表fact\_change\_record，分区字段为etldate且值与hudi\_gy\_ods库的相对应表该值相等，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”，dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。dwd\_modify\_time作为preCombineField，change\_id和change\_machine\_id作为联合primaryKey。使用spark-sql的cli按照change\_machine\_id、change\_id均为降序排序，查询前1条数据，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
2. 抽取hudi\_gy\_ods库中basemachine的全量数据进入Hudi的hudi\_gy\_dwd库中表dim\_machine。分区字段为etldate且值与hudi\_gy\_ods库的相对应表该值相等，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”，dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。dwd\_modify\_time作为preCombineField，Base\_machine\_id作为primaryKey。使用spark-sql的cli按照Base\_machine\_id升序排序，查询dim\_machine前2条数据，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
3. 抽取hudi\_gy\_ods库中producerecord的全量数据进入Hudi的hudi\_gy\_dwd库中表fact\_produce\_record,分区字段为etldate且值与hudi\_gy\_ods库的相对应表该值相等，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”，dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。dwd\_modify\_time作为preCombineField，produce\_record\_id和produce\_machine\_id作为联合primaryKey。使用spark-sql的cli按照produce\_machine\_id、produce\_record\_id均为升序排序，查询fact\_produce\_record前1条数据，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
4. 抽取hudi\_gy\_ods库中machinedata的全量数据进入Hudi的hudi\_gy\_dwd库中表fact\_machine\_data。分区字段为etldate且值与hudi\_gy\_ods库的相对应表该值相等，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”，dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。dwd\_modify\_time作为preCombineField，machine\_record\_id作为primaryKey。使用spark-sql的cli按照machine\_id、machine\_record\_id均为降序排序，查询前1条数据，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下。

### 子任务三：指标计算

1. 本任务基于以下2、3、4小题完成，使用DolphinScheduler完成第2、3、4题任务代码的调度。工作流要求，使用shell输出“开始”作为工作流的第一个job（job1），2、3、4题任务为并行任务且它们依赖job1的完成（命名为job2、job3、job4），job2、job3、job4完成之后使用shell输出“结束”作为工作流的最后一个job（endjob），endjob依赖job2、job3、job4，并将最终任务调度完成后的工作流截图，将截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；
2. 编写Scala代码，使用Spark根据hudi\_gy\_dwd层的fact\_change\_record表统计每个月（change\_start\_time的月份）、每个设备、每种状态的时长，若某状态当前未结束（即change\_end\_time值为空）则该状态不参与计算。计算结果存入ClickHouse数据库shtd\_industry的machine\_state\_time表中（表结构如下），然后在Linux的**ClickHouse**命令行中根据设备id、状态持续时长均为降序排序，查询出前10条，将SQL语句复制粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下;

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_id | int | 设备id |  |
| change\_record\_state | varchar | 状态 |  |
| duration\_time | varchar | 持续时长（秒） | 当月该状态的时长和 |
| year | int | 年 | 状态产生的年 |
| month | int | 月 | 状态产生的月 |

1. 编写Scala代码，使用Spark根据hudi\_gy\_dwd层的fact\_change\_record表关联dim\_machine表统计每个车间中所有设备运行时长（即设备状态为“运行”）的中位数在哪个设备（为偶数时，两条数据原样保留输出），若某个设备运行状态当前未结束（即change\_end\_time值为空）则该状态不参与计算，计算结果存入ClickHouse数据库shtd\_industry的machine\_running\_median表中（表结构如下），然后在Linux的**ClickHouse**命令行中根据所属车间、设备id均为降序排序，查询出前5条数据，将SQL语句复制粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下;

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_id | int | 设备id |  |
| machine\_factory | int | 所属车间 |  |
| total\_running\_time | int | 运行总时长 | 结果以秒为单位 |

1. 编写Scala代码，使用Spark根据hudi\_gy\_dwd层的fact\_produce\_record表，基于全量历史数据计算各设备生产一个产品的平均耗时，produce\_code\_end\_time值为1900-01-01 00:00:00的数据为脏数据，需要剔除（注：fact\_produce\_record表中，一条数据代表加工一个产品，produce\_code\_start\_time字段为开始加工时间，produce\_code\_end\_time字段为完成加工时间），将设备每个产品的耗时与该设备平均耗时作比较，保留耗时高于平均值的产品数据，将得到的数据写入**ClickHouse**数据库shtd\_industry的machine\_produce\_per\_avgtime表中（表结构如下），然后在Linux的ClickHouse命令行中根据设备id降序排序查询前3条数据，将SQL语句复制粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下。

machine\_produce\_per\_avgtime：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| produce\_record\_id | int | 生产记录id | 每生产一件产品产生一条数据 |
| produce\_machine\_id | int | 设备id |  |
| producetime | int | 该产品耗时 |  |
| produce\_per\_avgtime | int | 设备生产一个产品平均耗时 | 单位：秒 |

## 任务C：数据挖掘（10分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**各节点可通过Asbru工具或SSH客户端进行SSH访问；  主节点MySQL数据库用户名/密码：root/123456（已配置远程连接）；  Spark任务在Yarn上用Client运行，方便观察日志。  该任务均使用Scala编写，利用Spark相关库完成。 |

### 子任务一：特征工程

1. 根据hudi\_gy\_dwd库中fact\_machine\_data表（或MySQL的shtd\_industry库中MachineData表），根据以下要求转换：获取最大分区（MySQL不用考虑）的数据后，首先解析列machine\_record\_data（MySQL中为MachineRecordData）的数据（数据格式为xml，采用dom4j解析，解析demo在客户端/home/ubuntu/Documents目录下），并获取每条数据的主轴转速，主轴倍率，主轴负载，进给倍率，进给速度，PMC程序号，循环时间，运行时间，有效轴数，总加工个数，已使用内存，未使用内存，可用程序量，注册程序量等相关的值（若该条数据没有相关值，则按下表设置默认值），同时转换machine\_record\_state字段的值，若值为报警，则填写1，否则填写0，以下为表结构，将数据保存在Hive的dwd.fact\_machine\_learning\_data，使用cli按照machine\_record\_id升序排序，查询dwd.fact\_machine\_learning\_data前1条数据，将结果截图粘贴至客户端桌面【Release\任务C提交结果.docx】中对应的任务序号下。

dwd.fact\_machine\_learning\_data表结构：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_record\_id | int | 主键 |  |
| machine\_id | double | 设备id |  |
| machine\_record\_state | double | 设备状态 | 默认0.0 |
| machine\_record\_mainshaft\_speed | double | 主轴转速 | 默认0.0 |
| machine\_record\_mainshaft\_multiplerate | double | 主轴倍率 | 默认0.0 |
| machine\_record\_mainshaft\_load | double | 主轴负载 | 默认0.0 |
| machine\_record\_feed\_speed | double | 进给倍率 | 默认0.0 |
| machine\_record\_feed\_multiplerate | double | 进给速度 | 默认0.0 |
| machine\_record\_pmc\_code | double | PMC程序号 | 默认0.0 |
| machine\_record\_circle\_time | double | 循环时间 | 默认0.0 |
| machine\_record\_run\_time | double | 运行时间 | 默认0.0 |
| machine\_record\_effective\_shaft | double | 有效轴数 | 默认0.0 |
| machine\_record\_amount\_process | double | 总加工个数 | 默认0.0 |
| machine\_record\_use\_memory | double | 已使用内存 | 默认0.0 |
| machine\_record\_free\_memory | double | 未使用内存 | 默认0.0 |
| machine\_record\_amount\_use\_code | double | 可用程序量 | 默认0.0 |
| machine\_record\_amount\_free\_code | double | 注册程序量 | 默认0.0 |
| machine\_record\_date | timestamp | 记录日期 |  |
| dwd\_insert\_user | string |  |  |
| dwd\_insert\_time | timestamp |  |  |
| dwd\_modify\_user | string |  |  |
| dwd\_modify\_time | timestamp |  |  |

### 子任务二：报警预测

1. 根据子任务一的结果，建立随机森林（随机森林相关参数可自定义，不做限制），使用子任务一的结果训练随机森林模型，然后再将dwd.fact\_machine\_learning\_data\_test（该表字段含义与dwd.fact\_machine\_learning\_data表相同，machine\_record\_state列值为空，表结构自行查看）转成向量，预测其是否报警将结果输出到MySQL数据库shtd\_industry中的ml\_result表中（表结构如下）。在Linux的MySQL命令行中查询出machine\_record\_id为1、8、20、28和36的5条数据，将SQL语句复制并粘贴至客户端桌面【Release\任务C提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务C提交结果.docx】中对应的任务序号下。

ml\_result表结构：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_record\_id | int | 主键 |  |
| machine\_record\_state | double | 设备状态 | 报警为1，其他状态则为0 |

## 任务D：数据采集与实时计算（20分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**各节点可通过Asbru工具或SSH客户端进行SSH访问；  Flink任务在Yarn上用per job模式（即Job分离模式，不采用Session模式），方便Yarn回收资源。 |

### 子任务一：实时数据采集

1. 在主节点使用Flume采集/data\_log目录下实时日志文件中的数据（实时日志文件生成方式为在/data\_log目录下执行./make\_data\_file\_v1命令即可生成日志文件，如果没有权限，请执行授权命令chmod 777 /data\_log/make\_data\_file\_v1），将数据存入到Kafka的Topic中（Topic名称分别为ChangeRecord、ProduceRecord和EnvironmentData，分区数为4），将Flume采集ChangeRecord主题的配置截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下；
2. 编写新的Flume配置文件，将数据备份到HDFS目录/user/test/flumebackup下，要求所有主题的数据使用同一个Flume配置文件完成，将Flume的配置截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下。

### 子任务二：使用Flink处理Kafka中的数据

编写Scala工程代码，使用Flink消费Kafka中的数据并进行相应的数据统计计算。

1. 使用Flink消费Kafka中ProduceRecord主题的数据，统计在已经检验的产品中，各设备每5分钟生产产品总数，将结果存入Redis中，key值为“totalproduce”，value值为“设备id，最近五分钟生产总数”。使用redis cli以HGETALL key方式获取totalproduce值，将结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下，需两次截图，第一次截图和第二次截图间隔5分钟以上，第一次截图放前面，第二次截图放后面；

注：ProduceRecord主题，生产一个产品产生一条数据；

change\_handle\_state字段为1代表已经检验，0代表未检验；

时间语义使用Processing Time。

1. 使用Flink消费Kafka中ChangeRecord主题的数据，当某设备30秒状态连续为“预警”，输出预警信息。当前预警信息输出后，最近30秒不再重复预警（即如果连续1分钟状态都为“预警”只输出两次预警信息），将结果存入Redis中，key值为“warning30sMachine”，value值为“设备id，预警信息”。使用redis cli以HGETALL key方式获取warning30sMachine值，将结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下，需两次截图，第一次截图和第二次截图间隔1分钟以上，第一次截图放前面，第二次截图放后面；

注：时间使用change\_start\_time字段，忽略数据中的change\_end\_time不参与任何计算。忽略数据迟到问题。

Redis的value示例：115,2022-01-01 09:53:10:设备115 连续30秒为预警状态请尽快处理！

(2022-01-01 09:53:10 为change\_start\_time字段值，中文内容及格式必须为示例所示内容。)

1. 使用Flink消费Kafka中ChangeRecord主题的数据，统计每3分钟各设备状态为“预警”且未处理的数据总数，将结果存入MySQL数据库shtd\_industry的threemin\_warning\_state\_agg表中（追加写入，表结构如下）。请将任务启动命令复制粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下，启动且数据进入后按照设备id升序排序查询threemin\_warning\_state\_agg表进行截图，第一次截图后等待3分钟再次查询并截图,将结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下。

threemin\_warning\_state\_agg表：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 中文含义 |
| change\_machine\_id | int | 设备id |
| totalwarning | int | 未被处理预警的数据总数 |
| window\_end\_time | varchar | 窗口结束时间（yyyy-MM-dd HH:mm:ss） |

注：时间语义使用Processing Time。

## 任务E：数据可视化（15分）

**环境说明：**

|  |
| --- |
| **数据接口地址及接口描述详见各任务服务端说明。**  **注：所有数据排序按照接口返回数据顺序处理即可，不用特意排序。** |

### 子任务一：用柱状图展示设备历史各个状态持续时长

编写Vue工程代码，根据接口，用柱状图展示接口所有数据中各设备各个状态持续时长（秒），同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

### 子任务二：用柱状图展示每日所有车间各设备平均运行时长

编写Vue工程代码，根据接口，用柱状图展示每日所有车间各设备平均运行时长（秒，四舍五入保留两位小数），每日里的所有设备按照接口返回的数据顺序展示即可，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

子任务三：用折线图展示某设备每日运行时长

编写Vue工程代码，根据接口，用折线图展示设备OP160每日的运行时长（秒），同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

子任务四：用条形图展示每日各设备产量

编写Vue工程代码，根据接口，用条形图展示每日各设备产量，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

子任务五：用折柱混合图展示设备日均产量和所在车间日均产量

编写Vue工程代码，根据接口，用折柱混合图展示设备日均产量（四舍五入保留两位小数）和所在车间日均产量（四舍五入保留两位小数），其中柱状图展示各设备的日均产量，折线图展示该设备所在车间的日均产量，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

## 任务F：综合分析（10分）

### 子任务一：请简述ZooKeeper的选举策略，leader和follower的区别。

请概述ZooKeeper的选举策略、leader和follower的区别，将内容编写至客户端桌面【Release\任务F提交结果.docx】中对应的任务序号下。

### 子任务二：请简述MapReduce的Shuffle过程。

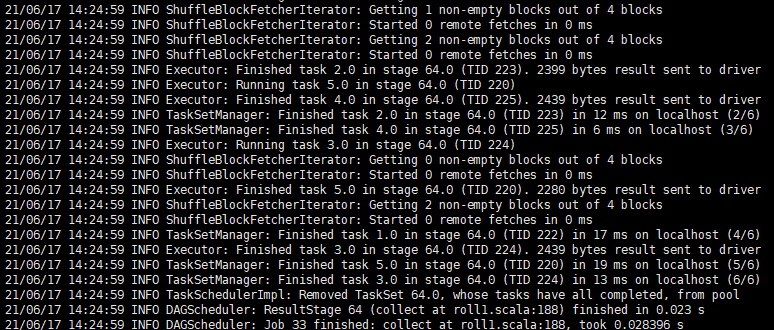
请简述MapReduce的Shuffle过程，将内容编写至客户端桌面【Release\任务F提交结果.docx】中对应的任务序号下。

### 子任务三：请根据可视化图表做以下分析。

如果你是工厂技术人员，通过可视化结果你将会对工厂设备优化（不限于该角度，可自行扩展分析角度）做出哪些建议，将内容编写至客户端桌面【Release\任务F提交结果.docx】中对应的任务序号下。

## 附录：补充说明

### 命令行截图样例（保证清晰）



### 离线数据：

#### 表BaseMachine：

|  |  |  |
| --- | --- | --- |
| 字段名 | 中文含义 | 备注 |
| BaseMachineID | 设备id |  |
| MachineFactory | 设备所在车间 |  |
| MachineNo | 设备编号 |  |
| MachineName | 设备名称 |  |
| MachineIP | 设备ip |  |
| MachinePort | 设备端口号 |  |
| MachineAddDate | 维护日期 |  |
| MachineRemarks | 备注 |  |
| MachineAddEmpID | 维护人id |  |
| MachineResponsEmpID | 设备责任人id |  |
| MachineLedgerXml | 设备附带台账信息 |  |
| ISWS | 设备采集协议 | 无实际意义 |

#### 表ChangeRecord：

|  |  |  |
| --- | --- | --- |
| 字段名 | 中文含义 | 备注 |
| ChangeID | 主键 |  |
| ChangeMachineID | 设备id | 关联BaseMachine表中的BaseMachineID |
| ChangeMachineRecordID | 设备记录id |  |
| ChangeRecordState | 设备所属状态 |  |
| ChangeStartTime | 状态开始时间 |  |
| ChangeEndTime | 状态终止时间 |  |
| ChangeRecordData | 具体信息 |  |
| ChangeHandleState | 是否处理 |  |

#### 表MachineData：

|  |  |  |
| --- | --- | --- |
| 字段名 | 中文含义 | 备注 |
| MachineRecordID | 主键 |  |
| MachineID | 设备id | 关联BaseMachine表中的BaseMachineID |
| MachineRecordState | 设备所属状态 |  |
| MachineRecordData | 设备记录数据 |  |
| MachineRecordDate | 记录日期 |  |

#### 表ProduceRecord：

|  |  |  |
| --- | --- | --- |
| 字段名 | 中文含义 | 备注 |
| ProduceRecordID | 主键 |  |
| ProduceMachineID | 设备id | 关联BaseMachine表中的BaseMachineID |
| ProduceCodeNumber | 生产程序的代码号 | 程序的编号 |
| ProduceStartWaitTime | 产品加工准备时间 | 也就是上一个产品加工的结束时间 |
| ProduceCodeStartTime | 产品开始加工时间 | 也就是员工按下启动按钮的时间 |
| ProduceCodeEndTime | 产品加工结束时间 | 也就是下一个产品的加工准备时间 |
| ProduceCodeCycleTime | 加工产品的标准时间 | 单位是毫秒 |
| ProduceEndTime | 加工产品结束时间 | 和ProduceCodeEndTime一样 |
| ProducePrgCode | 加工代码 | 加工此产品用的加工代码 |
| ProduceTotalOut | 累计加工数量 | 实时读取的设备累计加工数量 |
| ProduceInspect | 是否检验 | 是否已经检验，1代表已经检验，0代表未检验 |

#### 表EnvironmentData：

|  |  |  |
| --- | --- | --- |
| 字段名 | 中文含义 | 备注 |
| EnvoID | 主键 |  |
| BaseID | 设备id | 关联BaseMachine表中的BaseMachineID |
| CO2 | 环境检测项 |  |
| PM25 | 环境检测项 |  |
| PM10 | 环境检测项 |  |
| Temperature | 环境检测项 | 温度 |
| Humidity | 环境检测项 | 湿度 |
| TVOC | 环境检测项 |  |
| CH2O | 环境检测项 |  |
| Smoke | 环境检测项 |  |
| InPutTime | 维护时间 |  |

### 实时数据：

实时数据字段之间用逗号（“,”）分割。

#### \*\*\*\*\*\*\*-changerecord.csv：

|  |  |  |
| --- | --- | --- |
| 字段 | 中文含义 | 备注 |
| change\_id | 主键 |  |
| change\_machine\_id | 设备id |  |
| change\_machinerecord\_id | 该字段无意义 | 该字段无意义 |
| change\_record\_state | 状态信息 |  |
| change\_start\_time | 状态开始时间 |  |
| change\_end\_time | 状态结束时间 |  |
| change\_handle\_state | 当change\_record\_state为报警的时候，需要处理此报警信息 | 偶数代表未处理，奇数代表已处理 |

#### \*\*\*\*\*\*\*-producerecord.csv：

|  |  |  |
| --- | --- | --- |
| 字段 | 中文含义 | 备注 |
| produce\_record\_id | 主键 | 加工一个产品产生一条数据 |
| produce\_machine\_id | 设备id |  |
| produce\_code\_number | 程序号 |  |
| produce\_start\_wait\_time | 产品加工准备时间 | 也就是上一个产品加工的结束时间 |
| produce\_code\_start\_time | 产品开始加工时间 | 也就是员工按下启动按钮的时间 |
| produce\_code\_end\_time | 产品加工结束时间 | 也就是下一个产品的加工准备时间 |
| produce\_code\_cycle\_time | 加工产品的标准时间，单位是毫秒 |  |
| produce\_end\_time | 加工产品结束时间 | 和ProduceCodeEndTime一样 |
| produce\_totalout | 该字段无意义 | 该字段无意义 |
| produce\_inspect | 是否已经检验 | 1代表已经检验，0代表未检验 |

#### \*\*\*\*\*\*\*-environmentdata.csv：

|  |  |  |
| --- | --- | --- |
| 字段 | 中文含义 | 备注 |
| EnvoId | 主键 |  |
| BaseID | 设备id |  |
| CO2 | 环境检测项 |  |
| PM25 | 环境检测项 |  |
| PM10 | 环境检测项 |  |
| Temperature | 环境检测项 | 温度 |
| Humidity | 环境检测项 | 湿度 |
| TVOC | 环境检测项 |  |
| CH2O | 环境检测项 |  |
| Smoke | 环境检测项 | 烟雾 |
| InPutTime | 维护时间 |  |