任务4-2 构建流处理应用

|  |  |
| --- | --- |
| 任务书 | |
| 一、任务描述与要求 | |
| 任务描述 | 分析并明确流处理应用的具体需求，包括数据源、处理逻辑、输出要求等。设计应用的整体架构，确定使用的技术栈和处理流程。 |
| 具体要求 | 1.选择和配置适合的数据源；  2.进行流处理应用的需求分析和架构设计。 |
| 二、任务目标 | |
| 知识目标 | * 了解流处理的基本概念和原理； * 掌握 Spark Streaming 的核心组件。 |
| 能力目标 | * 能够使用 Spark Streaming 构建简单的流处理应用； * 具备对流处理应用进行性能优化的能力。 |
| 三、知识储备 | |
| 1.创建和配置StreamingQuery  创建和配置StreamingQuery是Spark结构化流处理中的核心步骤。这个过程主要涉及设置如何输出处理后的数据流，以及如何控制查询的执行。  首先需要选择输出模式，Spark提供了Complete、Append和Update三种主要模式，根据数据处理需求选择适当的模式。接着指定输出格式和目标，可以是控制台、文件系统或外部数据库等。  触发器配置决定了流处理作业的执行频率，可以基于时间间隔或数据量来设置。此外，还可以设置查询名称和检查点位置等参数，这些有助于查询管理和故障恢复。  最后通过调用start()方法启动查询，返回的StreamingQuery对象可用于监控和控制查询的执行。  IMG_256  2. 使用DataFrame和Dataset API处理流数据  使用DataFrame和Dataset API处理流数据是Spark结构化流处理的核心功能，它为开发者提供了一种强大而灵活的方式来处理实时数据流。  首先，通过SparkSession的readStream()方法，开发者可以从多种数据源创建流式DataFrame，包括文件系统、Kafka、Socket等。这个流式DataFrame本质上是一个无界表，随着时间推移不断有新数据加入。  创建流式DataFrame后，可以应用各种转换操作，如select、filter、map、flatMap等。这些操作与处理静态数据的方式非常相似，使得开发者可以轻松地将批处理代码迁移到流处理环境。此外，还支持groupBy、join等复杂操作，但在流处理中这些操作可能有一些特殊的限制或行为。 | |

|  |
| --- |
| 操作指南 |
| 一、制定计划 |
| |  |  |  |  | | --- | --- | --- | --- | | **序号** | **作业项目** | **序号** | **作业项目** | | 1 |  | 5 |  | | 2 |  | 6 |  | | 3 |  | 7 |  | | 4 |  | 8 |  | | 计划审核 | 审核意见： | | | |
| 二、实施方案 |
| 1. 导入必要的库  通过该案例可以展示如何使用Spark结构化流处理实时数据流，包括数据生成、聚合操作、状态管理，以及查询配置和输出，这里模拟分析订单数据，对订单信息和状态进行跟踪。  首先，导入必要的库和初始化SparkSession，便于后续生成数据以及分析使用。具体代码如下：   |  | | --- | | import org.apache.spark.sql.SparkSession  import org.apache.spark.sql.functions.\_  import org.apache.spark.sql.streaming.{GroupState, GroupStateTimeout}  val spark = SparkSession.builder()  .appName("SalesStreamingApp")  .master("local[\*]")  .getOrCreate()  import spark.implicits.\_ |   2. 定义数据模型  这里需要使用rate源的数据，这里需要定义数据模型，具体代码如下：   |  | | --- | | case class SaleEvent(productId: String, amount: Double, timestamp: Long) |   3.创建模拟的输入流  使用 rate 源生成数据，然后转换为 SaleEvent，具体代码如下：   |  | | --- | | val rawDataStream = spark.readStream  .format("rate")  .option("rowsPerSecond", 5)  .load()  val saleEventStream = rawDataStream.select(  (rand() \* 3).cast("int").as("productId"),  (rand() \* 100).cast("double").as("amount"),  $"timestamp"  ).as[SaleEvent] |   4.进行流数据的聚合操作  这边数据已经有了，就对流数据进行处理，这里先完成聚合操作，具体代码如下：   |  | | --- | | val salesAggregates = saleEventStream  .groupBy(  window($"timestamp", "1 minute"),  $"productId"  )  .agg(sum("amount").alias("total\_sales")) |   5.使用状态管理跟踪产品销售历史  使用 rate 源生成数据，然后转换为 SaleEvent，具体代码如下：   |  | | --- | | case class ProductHistory(productId: String, totalSales: Double, lastSaleTime: Long)  def updateProductHistory(productId: String, events: Iterator[SaleEvent], state: GroupState[ProductHistory]): ProductHistory = {  val oldHistory = state.getOption.getOrElse(ProductHistory(productId, 0, 0))  val newHistory = events.foldLeft(oldHistory) { (history, event) =>  ProductHistory(productId, history.totalSales + event.amount, event.timestamp)  }  state.update(newHistory)  newHistory  }  val productHistories = saleEventStream  .groupByKey(\_.productId)  .mapGroupsWithState(GroupStateTimeout.NoTimeout())(updateProductHistory) |   6.创建和配置StreamingQuery  在上述工作完成后，接下来可以通过StreamingQuery完成查找，具体代码如下：   |  | | --- | | val aggregateQuery = salesAggregates.writeStream  .outputMode("complete")  .format("console")  .option("truncate", false)  .start()  val historyQuery = productHistories.writeStream  .outputMode("update")  .format("console")  .option("truncate", false)  .start()  // 等待查询终止  spark.streams.awaitAnyTermination() |   在上述代码完成后，将两个查询的结果都输出到控制台，这样可以直接在终端中看到结果，运行这个应用程序，你会看到控制台中不断输出更新的销售聚合结果和产品历史记录。 |
| 三、实施记录 |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **序号** | **关键步骤记录** |  | **序号** | **解决问题记录** | | 1 |  |  | 1 |  | | 2 |  |  | 2 |  | | 3 |  |  | 3 |  | | 4 |  |  | 4 |  | | 5 |  |  | 5 |  | | 6 |  |  | 6 |  | | 7 |  |  | 7 |  | | 8 |  |  | 8 |  | | 9 |  |  | 9 |  | |
| 四、任务拓展 |
| 在数据处理完成后，系统将把实时的销售统计结果和销售历史记录输出到控制台，以便于用户进行实时监控和分析。最终，系统将提供准确的销售数据，帮助用户了解当前的销售状况。 |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 评价标准 | | | | | | | | |
| 一、基本信息 | | | | | | | | |
| 项目名称 | | 单元四 结构化流处理 | | | | | | |
| 任务名称 | | 任务4-2 构建流处理应用 | | | | | | |
| 班级 |  | | 学号 | |  | 姓名 | |  |
| 组名 |  | | 学时 | |  | 日期 | |  |
| 组员  分工 |  | | |  | | |  | |
|  | | |  | | |  | |
| 二、任务检查评价单 | | | | | | | | |
| |  |  |  | | --- | --- | --- | | **评价内容** | **评价标准** | **得分** | | 知识运用  （20分） | 掌握相关理论知识，理解本次任务要求，制定详细计划，计划条理清晰，逻辑正确（20分） |  | | 理解相关理论知识，能根据本次任务要求、制定合理计划（15分） | | 了解相关理论知识，有制定计划（10分） | | 无制定计划（0分） | | 专业技能  （40分） | 结果验证全部满足。（40分） |  | | 结果验证只有一个功能不能实现，其它功能全部实现（30分） | | 结果验证只有一个功能实现，其它功能全部没有实现（20分） | | 结果验证功能均未实现（0分） | | 核心素养  （20分） | 具有良好的自主学习能力、分析解决问题的能力、整个任务过程中有指导他人（20分） |  | | 具有较好的学习能力和分析解决问题的能力，任务过程中无指导他人（15分） | | 能够主动学习并收集信息，有请教他人进行解决问题的能力（10分） | | 不主动学习（0分） | | 课堂纪律  （20分） | 设备无损坏、设备摆放整齐、工位区域内保持整洁、无干扰课堂秩序（20分） |  | | 设备无损坏、无干扰课堂秩序（15分） | | 无干扰课堂秩序（10分） | | 干扰课堂秩序（0分） | | 得分（满分100） | |  | | 综合评价 | ☆☆☆☆☆ | | | | | | | | | | |