



**分布式程序设计期末报告**

**2023~2024第一学期**

报告题目：Flink技术学习及应用

学生姓名：杨程锦

专 业：计算机科学与技术

班 级：21-1班

学 号：2021214710

指导教师：张 赞

完成日期：2024年1月19日

**一、Flink基本介绍**

**1.1 Flink介绍**

任何类型的数据都可以形成一种事件流。信用卡交易、传感器测量、机器日志、网站或移动应用程序上的用户交互记录，所有这些数据都形成一种流。数据可以被作为无界或者有界流来处理，广泛用于大数据和实时数据处理。它具有高吞吐量、低延迟、Exactly-Once语义等特性，

**1.2 Flink的应用场景**

实时数据处理： Flink可用于实时处理大规模数据流，支持复杂的事件处理和数据转换。这在需要快速而准确地对实时数据进行分析和操作的场景中非常有用，比如金融交易监控、网络日志分析等。

事件驱动应用： Flink提供了强大的事件时间处理机制，使其非常适用于构建事件驱动的应用程序。这包括处理实时事件流、实时监控、报警系统等。

批处理： 虽然Flink主要是流处理框架，但它也支持批处理。这使得它能够在同一个框架中处理批处理和实时处理任务，简化了系统架构。

机器学习： Flink可以与机器学习库集成，用于实时的机器学习任务。这在需要实时更新模型或进行在线学习的应用中非常有用。

复杂事件处理（CEP）： Flink支持复杂事件处理，能够检测和处理复杂的事件模式。这在需要对一系列事件进行模式匹配和分析的应用中非常实用，如欺诈检测、实时监控等。

数据湖分析： Flink可以与数据湖集成，用于实时分析和查询数据湖中的数据。

日志和指标分析： Flink可以用于实时分析系统日志和性能指标，帮助识别和解决问题。

**1.3 Flink的流处理特性**

高吞吐、低延迟、高性能的流处理： Flink提供高效的流处理能力，具备高吞吐率和低延迟，以满足实时数据处理需求。

带有事件时间的窗口操作： Flink支持基于事件时间的窗口操作，使得处理延迟和乱序数据时能够确保准确性。

有状态计算的Exactly-Once语义： Flink支持有状态计算，并提供Exactly-Once语义，确保在故障发生时不会发生数据重复或丢失。

灵活的窗口操作： Flink支持多种窗口操作，包括基于时间、计数、会话以及数据驱动的窗口操作，满足不同应用场景的需求。

具有背压功能的持续流模型： Flink支持背压机制，能够处理流量过大时的流控制，确保系统稳定运行。

基于轻量级分布式快照的容错： Flink通过Checkpoint机制实现容错，使用轻量级分布式快照保存应用程序状态，以便在故障发生时进行恢复。

Batch on Streaming处理和Streaming处理： Flink运行时支持同时进行批处理和流处理，使得开发者能够在同一框架内处理不同类型的任务。

自身内存管理： Flink在JVM内部实现了自己的内存管理，优化了内存使用，提高了系统性能。

迭代计算： Flink支持迭代计算，适用于需要多次迭代的计算任务。

程序自动优化： Flink具备自动优化功能，避免在特定情况下进行昂贵的Shuffle、排序等操作，并对中间结果进行必要的缓存，提升执行效率。

**1.4 Flink的优势**

**相对于Hadoop、MapReduce批处理的第一代**：

Flink引入了流处理，不仅支持批处理还支持实时流处理，极大地提高了数据处理的实时性。

Flink自带DAG，简化了任务的构建和执行过程，降低了开发难度。

性能表现更优，具备高吞吐、低延迟的特性，适用于更广泛的应用场景。

**相对于DAG框架（Oozie、Tez）,Tez+MapReduce批处理的第二代**：

Flink在支持批处理的同时，更强调流处理，具备更灵活的窗口操作和事件时间处理，提供更全面的实时计算能力。

Flink的有状态计算和Exactly-Once语义提供了更可靠的数据处理保障。

内部实现了自己的内存管理，进一步优化了性能。

**相对于Spark批处理、流处理、SQL高层API支持的第三代：**

Flink在流处理性能上更具优势，支持具有背压功能的持续流模型，提高了系统的稳定性。

Flink的流式计算性能更高、可靠性更强，使其在实时大数据处理中更为出色。

**相对于Flink批处理、流处理、SQL高层API支持的第四代**：

Flink在流式计算性能方面进一步提升，更适用于需要快速而可靠的实时数据处理场景。

Flink的SQL高层API支持更加完善，使得开发者能够更轻松地进行数据处理和分析。

**1.5 Flink的四大基石**

**Checkpoint（检查点）：**

基于Chandy-Lamport算法实现分布式计算任务的一致性语义。

提供了容错机制，通过定期保存应用程序状态，确保在发生故障时能够进行准确的恢复，实现Exactly-Once语义。

**State（状态）：**

Flink中的状态机制，天生支持状态的处理，状态可以被视为程序的中间计算结果或历史计算结果。

允许在处理流数据时维护和管理状态，为有状态计算提供了强大的支持。

**Time（时间）：**

Flink支持基于事件时间和处理时间进行计算。

基于事件时间的计算有助于解决数据迟到和乱序等问题，提高对实时数据的准确性和可靠性。

**Window（窗口）：**

Flink提供丰富的窗口操作，包括基于时间、数量、会话等多种窗口类型。

支持滚动窗口和滑动窗口的计算，使开发者能够更灵活地处理不同窗口类型的数据。

**二、Flink架构**

**2.1 Flink的重要组件**

**JobManager（作业管理器）**：

职责： JobManager负责整个作业的管理和协调。

具体功能：

接收客户端提交的作业（Job）并进行解析。

对作业进行优化、分配任务给TaskManager，并生成执行图。

协调任务的调度和执行。

负责作业的整体状态管理、容错机制的实现，包括生成和处理检查点（Checkpoints）。

与Client通信，接收作业的配置和控制指令。

**TaskManager（任务管理器）：**

职责： TaskManager负责具体任务的执行。

具体功能：

接收来自JobManager的任务，并执行其中的子任务（Task）。

在本地维护任务所需的状态和数据。

与其他TaskManager进行通信，进行数据的交换和协同计算。

处理任务的生命周期，包括任务的初始化、执行和结束。

与JobManager通信，报告任务的状态、心跳等信息。

**2.2 无界数据流和有界数据流**

**无界数据流（Unbounded Data Streams）**：

代表实时流数据，流数据的产生是持续不断的，没有明确的结束点。

通常用于实时数据处理场景，如实时日志处理、实时监控等。

数据流是无限的，不会在处理过程中终止。

**有界数据流（Bounded Data Streams）**：

代表有限批量的数据，数据有一个明确的开始和结束。

通常用于离线批处理场景，如对历史数据的分析、离线报表生成等。

数据流在处理过程中有一个确定的终止点。

**2.3 Flink编程模型**

**最底层的有状态流抽象：**

提供了底层的有状态流处理，通过过程函数（Process Function）嵌入到DataStream API中。

允许用户处理来自一个或多个数据流的时间，使用一致的容错状态，并注册事件时间和处理时间回调。

**核心API（DataStream API和DataSet API）：**

针对大多数应用，开发者通常使用核心API进行编程，如DataStream API（用于无界或有界流数据）和DataSet API（用于有界数据集）。

提供通用的构建模块，包括用户定义的转换、连接、聚合、窗口操作等。

数据类型以类的形式表示，适用于各种编程语言。

**Table API：**

以表为中心的声明式编程，支持动态变化的表达式，特别适用于处理流数据。

遵循关系模型，提供可比较的操作，如select、project、join、group\_by、aggregate等。

程序声明式地定义逻辑操作，不需关注具体操作代码的实现。

**SQL：**

最高层次的抽象，与Table API交互密切，以SQL查询表达式的形式表示程序。

SQL抽象在语法和表达能力上与Table API相似，允许直接在Table API定义的表上执行查询。

SQL查询可以在保留内置优化器的同时，简化了程序的表达。

**三、Flink集群搭建**

步骤：

1、搭建Hadoop集群，这一步省略，可使用在大数据处理技术课程中搭建的Hadoop集群。

2、下载flink-1.7.2-bin-hadoop27-scala\_2.11.tgz



3、集群规划

node1（master+slave）、node2（slave）、node3（slave）

4、将Flink压缩包上传至node1结点根目录export/software下。

5、解压flink-1.7.2-bin-hadoop27-scala\_2.11.tgz

tar -zxvf flink-1.7.2-bin-hadoop27-scala\_2.11.tgz -C /export/server/

文本

描述已自动生成

6、修改文件名

mv flink-1.7.2/ flink/

cd flink/

文本, 表格

描述已自动生成

7、修改用户权限

cd ..

chown -R root:root flink/

文本

低可信度描述已自动生成

8、启动本地模式Flink

bin/start-scala-shell.sh local

文本

中度可信度描述已自动生成

9、提交任务用于统计词频

benv.readTextFile("/root/words.txt").flatMap(\_.split(" ")).map((\_,1)).groupBy(0).sum(1).print()

电视游戏的萤幕截图

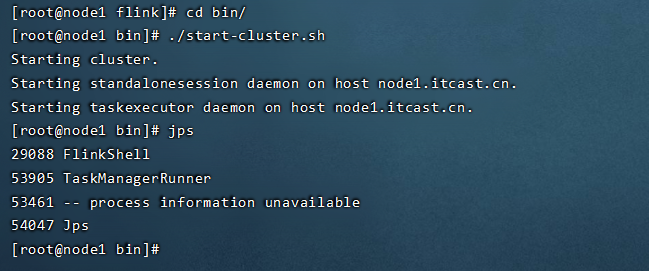
中度可信度描述已自动生成

10、启动集群（单节点）

cd bin/

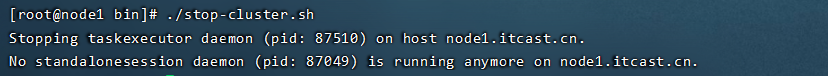
./start-cluster.sh

Jps



11、停止集群（单节点）

./stop-cluster.sh



12、在网页输入node1:8081可以查看webUI界面

社交网站的手机截图

描述已自动生成

13、安装standalone集群模式

修改配置文件 flink-conf.yaml

vim /conf/flink-conf.yaml

修改内容：

# jobManager 的IP地址

jobmanager.rpc.address: node1

# 每个 TaskManager 提供的任务 slots 数量大小

taskmanager.numberOfTaskSlots: 2

#是否进行预分配内存，默认不进行预分配，这样在我们不使用flink集群时候不会占用集群资源

taskmanager.memory.preallocate: false

# 程序默认并行计算的个数

parallelism.default: 1

#JobManager的Web界面的端口（默认：8081）

jobmanager.web.port: 8081



文本

描述已自动生成

蓝色的球

描述已自动生成

屏幕上有字

描述已自动生成



14、修改masters

vim /conf/masters



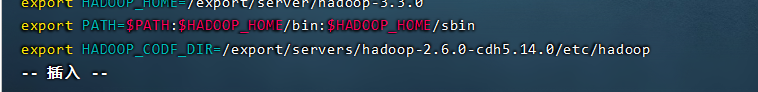
15、修改slaves文件

vim /conf/slaves

蓝色的天空

描述已自动生成

16、添加HADOOP\_CONF\_DIR环境变量



17、把flink发到两个结点

cd /export/server/

scp -r flink/ node2:$PWD

scp -r flink/ node3:$PWD



18、启动集群，在node1下执行

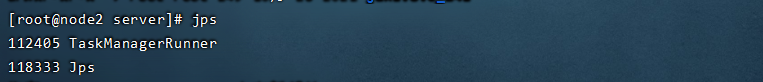
cd flink/

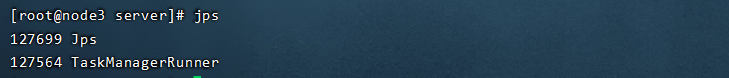
bin/start-cluster.sh

图片包含 文本

描述已自动生成

可以看到在node1上运行了jobmanger,taskmanger、node2上运行了jtaskmanger、node3上运行了taskmanger。



查看webUI

图形用户界面, 文本, 应用程序, 电子邮件, 网站

描述已自动生成

电脑屏幕截图

描述已自动生成

在Web UI 页面可以看到集群的具体配置 3个taskmanger,6个 task slots。

19、测试

上传清洗的数据到HDFS

（1）启动Hadoop

start-all.sh

hadoop fs -ls hdfs://node1:8020/

手机屏幕的截图

描述已自动生成

上传words.txt到HDFS中。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

cd /export/server/flink/

bin/flink run examples/batch/WordCount.jar --input

hdfs://node1:8020/wordcount/input/words.txt --output

hdfs://node1:8020/wordcount/output

海上有许多船

中度可信度描述已自动生成

在HDFS页面中就可以查看了

图形用户界面, 文本, 应用程序

描述已自动生成

下载到本地查看 输出结果

图形用户界面, 应用程序

描述已自动生成

在Flink的WebUI界面查看执行任务

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图示

描述已自动生成

Flink standalone模式已完全搭建成功。

**四、Flink集群搭建**

**4.1 构建工程导入maven依赖**

文本

描述已自动生成

**4.2 创建log4j.properties文件**

log4j.rootLogger=INFO, console

log4j.appender.console=org.apache.log4j.ConsoleAppender

log4j.appender.console.layout=org.apache.log4j.PatternLayout

log4j.appender.console.layout.ConversionPattern=%d{HH:mm:ss,SSS} %-5p %-60c %x - %m%n

图形用户界面, 文本

描述已自动生成

**4.3 添加scala插件**

电脑屏幕截图

描述已自动生成

**4.4 用scala语言编写wordcount程序**

package cn.itcast.flink.batch

import org.apache.flink.api.scala.ExecutionEnvironment

import org.apache.flink.api.scala.\_

/\*

\*  使用Flink批处理进行单词计数

\* \*/

object Word {

  def main(args: Array[String]): Unit = {

    /\*

      获得一个 execution envirtionment

      加载、创建初始数据

      指定数据的转换

      指定将计算结果放在哪里

      触发程序执行

     \*/

    val env: ExecutionEnvironment = ExecutionEnvironment.getExecutionEnvironment

    env.setParallelism(1)

    val sourceDs: DataSet[String]=env.fromElements(data = "Apache Flink is anopen source platform for distributed " +

      "stream and batch dataprocessing",

      "Flink’s core is a streaming dataflow engine that provides data distribution")

    val wordsDs: DataSet[String] = sourceDs.flatMap(fun = \_.split(" "))

    val wordAndOneDs:DataSet[(String,Int)] = wordsDs.map(fun = (\_,1))

    val groupDs:GroupedDataSet[(String,Int)] = wordAndOneDs.groupBy(fields = 0)

    val aggDs:AggregateDataSet[(String,Int)]=groupDs.sum(field = 1)

    aggDs.writeAsText(filePath = "./output.txt")

    env.execute()

  }

}

图片包含 图形用户界面

描述已自动生成

输出结果：

文本

描述已自动生成

**五、问题及解决方案**

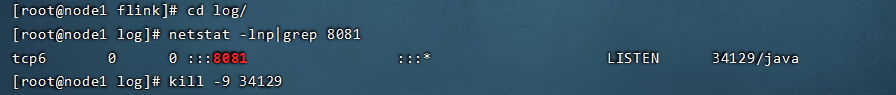
**5.1 Flink local模式集群搭建成功，无法刷新webUI**

解决方案：8081端口被占用

cd log/

netstat -lnp|grep 8081

kill -9 34129



**5.2 程序并行度**

程序输出结果保存在当地或者HDFS时，如果不设置程序并行度，其默认并行度是PC机的核数，因此会造成一份输出数据可能有输出文件，解决这一问题就得设置程序并行度，添加下方代码，使用setParallelise() API 设置并行度为1 这样最终输出文件就是一个。

env.setParallelism(1)

文本

描述已自动生成

**六、感想体会**

选择Flink大数据技术介绍及应用作为期末报告的原因是在深入学习Spark技术后，我了解到Flink作为第四代大数据技术目前非常热门，因此想要深入了解和学习。与Spark相比，Flink的学习过程对我来说稍显曲折，主要是因为Spark现在支持使用Python语言进行学习，而Flink仅支持Java和Scala两种语言，而我对这两种语言的基础知识较为欠缺，导致在学习过程中遇到了许多小问题，增加了难度。不过，由于之前掌握了Spark，我在学习Flink时能够发现其中很多与Spark相似的地方，使得其思想理解并不太困难。

在学习过程中，我遇到了许多问题，其中最令人头疼的是环境配置。虽然网上有相关的博客和教程，但实际配置过程中总是出现意外问题。在一段时间里，我曾考虑放弃并选择其他课题。例如，在成功搭建Flink环境后，启动时发现TaskManager数目不对，Web UI页面打不开。经过漫长的搜索和尝试，我发现是之前有进程占用了8081端口。最令人头疼的是配置IDEA环境时，pom.xml文件报错不知道多少次，解决后程序运行又出现连接不到HDFS等问题。这些问题花费了我两三天的时间，期间曾一度想要放弃，但最终通过查阅无数博客的解决方法，终于成功了。

总体而言，整个学习过程虽然痛苦，但我确实从中收获了很多。对于大数据技术，我有了更清晰的认识。