



青训营结业项目答辩汇报--Hello, Flink组（项目二）

一、项目介绍

我们组选择的是项目二，目前已经完成了大约80%，实现了WordCount小程序初级功能和Watermarks处理乱序延迟数据的中级功能，Retract和自动故障恢复由于时间比较紧张所以还没有完成，然后Exactly Once高级功能只写好了相应的配置，还没有设计相应的案例进行测试。通过测试，目前已写好的功能基本都能实现，满足题目所提的要求。

项目服务地址-必须

Github 地址，权限设置为 public- 必须

二、项目分工

好的团队协作可以酌情加分哟～请组长和组员做好项目分工与监督。

团队成员	主要贡献
朱骏凯	作为组长，参与项目初级功能和中级功能的讨论，设计，编码开发以及测试工作（贡献度30%）
伍莞秋	参与项目初级功能的讨论，设计，编码开发以及测试工作（贡献度25%）
房哲	参与项目中级功能的讨论，设计，编码开发以及测试工作（贡献度25%）
谢立国	参与项目的文档撰写和整理（贡献度10%）
郭世哲	参与项目的测试（贡献度10%）

三、项目实施

3.1 技术选型与相关开发文档

可以补充场景分析环节，明确要解决的问题和前提假设，比如按当前选型和架构总体预计需要xxx存储空间，xxx台服务器.....。

本次的项目均在Win10本地电脑进行开发和测试，后期会考虑打包上传到Linux服务器进行测试，毕竟Kafka更适用于Linux上进行配置。

3.2 架构设计

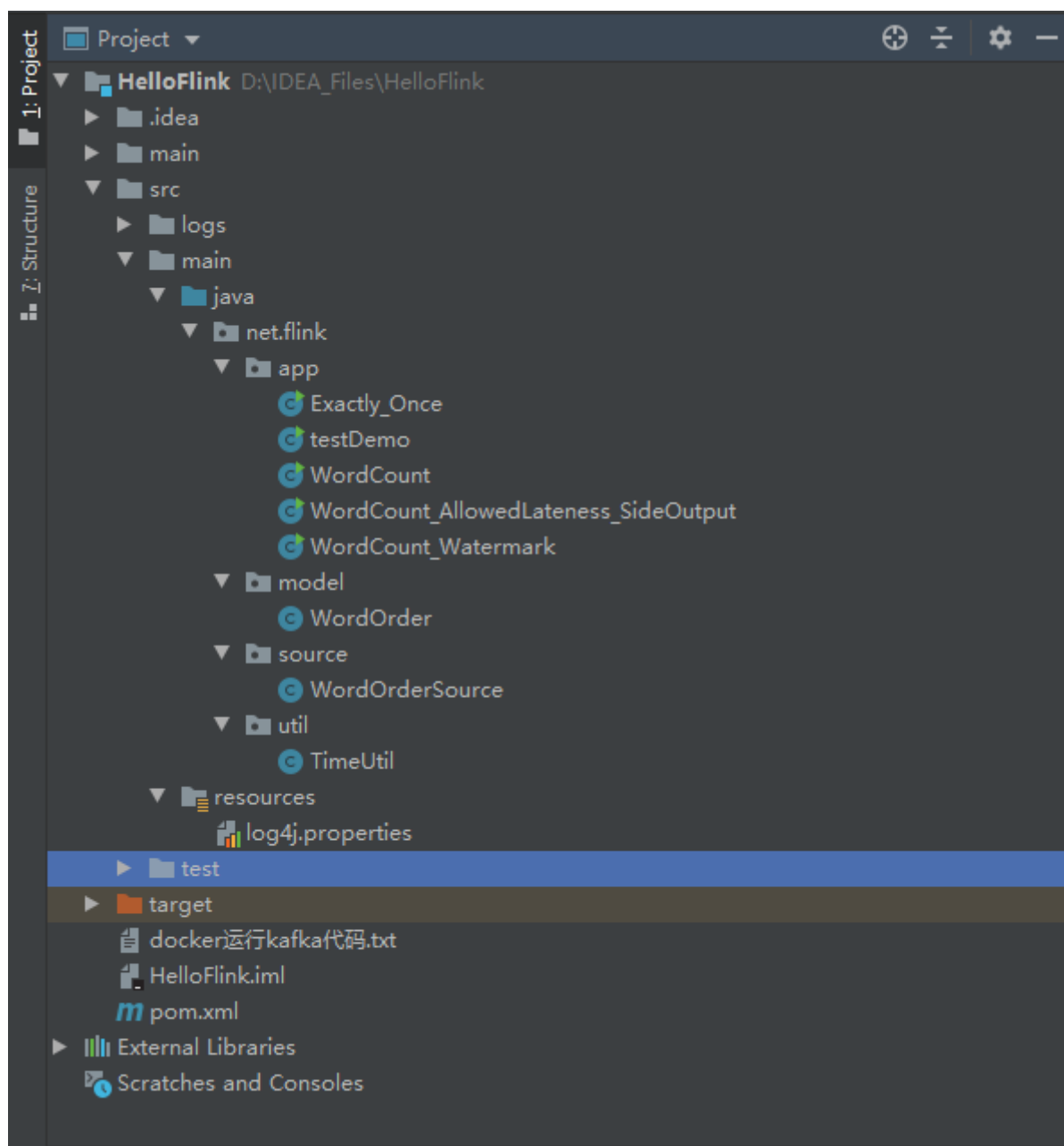
可以补充场景分析环节，明确要解决的问题和前提假设，比如预计0.5%的用户属于大V，粉丝很多，也会经常上传视频，当前架构的解决方案是xxx。

Flink的开发我们都按照标准的“Source -> Transformation -> Sink”的有向无环图DAG格式进行设计和开发。其中Source用了多种形式进行实现，比如说预定义的

fromElements，自己撰写了一个可以源源不断产生数据的类，Socket，Kafka等；Transformation使用了flatMap、Map、keyBy、sum、reduce等多种算子，同时设置了滚动窗口，将全量和增量窗口结合进行使用，并设置了Watermarks处理延迟乱序数据；Sink同时开发了输出到控制台以及输出到.txt两种输出方式。

总体来看，整个开发的架构还是很有层次的，而且代码处都有相应的注释。

3.3 项目代码介绍



上图为整个项目的代码结构：

pom.xml：添加相应的依赖

log4j.properties：日志管理

TimeUtil：时间格式转换类，包含Date类型转String类型、Long类型转String类型以及String类型转Date类型共三种应用场景

WordOrder：构造一个类存放输入的数据时间、单词以及个数

WordOrderSource：构造一个类产生源源不断随机产生输入数据

WordCount：统计5mins内每个单词的数量，我这边做了一个改进：就是不局限于一条记录一个单词只产生一次，而是让用户自己定义单词的数量。使用增量和全量窗口相结合的方式，在节省资源提高效率的同时还返回了窗口的时间信息

WordCount_Watermark：设置Watermarks机制处理延迟乱序数据，其中设置窗口大小为10s，延迟时间为3s

WordCount_AllowedLateness_SideOutput：在Watermarks的基础上设置allowedLateness二次兜底延迟数据处理和SideOutput兜底延迟数据处理，二次兜底时间为1min

Exactly_Once：写好了Exactly Once Checkpoint的相关配置，但相应的测试案例还没设计好

备注：

1.Kafka作为Source的代码我也写好了，但是由于其更适合用于Linux上进行测试，因此我在WordCount小程序中使用自己定义类源源不断地产生数据流代替进行测试，然后Watermarks那里采用Socket和nc方法代替进行测试，效果都是一样的

2."testDemo"是一个实现字符串切割的flink测试demo,与本项目无关

五、测试结果

建议从功能测试和性能测试两部分分析，其中功能测试补充测试用例，性能测试补充性能分析报告、可优化点等内容。

(1). 功能测试

①.WordCount

测试用例：

WordOrderSource源源不断随机产生的WordOrder数据记录（每隔1s产生1条数据）

输出结果：

每隔5mins统计一次单词数量

Time: 2022/08/20 16:05:00, Word: apple, Count: 1759
Time: 2022/08/20 16:05:00, Word: pie, Count: 1464
Time: 2022/08/20 16:05:00, Word: water, Count: 1884
Time: 2022/08/20 16:05:00, Word: orange, Count: 1375
Time: 2022/08/20 16:05:00, Word: paper, Count: 2114
Time: 2022/08/20 16:10:00, Word: pie, Count: 2833
Time: 2022/08/20 16:10:00, Word: water, Count: 2826
Time: 2022/08/20 16:10:00, Word: orange, Count: 2586
Time: 2022/08/20 16:10:00, Word: paper, Count: 2872
Time: 2022/08/20 16:10:00, Word: apple, Count: 2440
Time: 2022/08/20 16:15:00, Word: paper, Count: 2260
Time: 2022/08/20 16:15:00, Word: apple, Count: 3569
Time: 2022/08/20 16:15:00, Word: water, Count: 2751
Time: 2022/08/20 16:15:00, Word: orange, Count: 2634
Time: 2022/08/20 16:15:00, Word: pie, Count: 2982
Time: 2022/08/20 16:20:00, Word: paper, Count: 3502
Time: 2022/08/20 16:20:00, Word: pie, Count: 2674
Time: 2022/08/20 16:20:00, Word: water, Count: 1868
Time: 2022/08/20 16:20:00, Word: orange, Count: 2375
Time: 2022/08/20 16:20:00, Word: apple, Count: 3440
Time: 2022/08/20 16:25:00, Word: orange, Count: 3067
Time: 2022/08/20 16:25:00, Word: water, Count: 2335
Time: 2022/08/20 16:25:00, Word: pie, Count: 3072
Time: 2022/08/20 16:25:00, Word: apple, Count: 3151
Time: 2022/08/20 16:25:00, Word: paper, Count: 3193
Time: 2022/08/20 16:30:00, Word: orange, Count: 3348
Time: 2022/08/20 16:30:00, Word: paper, Count: 3114
Time: 2022/08/20 16:30:00, Word: water, Count: 3014
Time: 2022/08/20 16:30:00, Word: pie, Count: 2364
Time: 2022/08/20 16:30:00, Word: apple, Count: 2180

②.Watermarks

测试用例（在nc中依次输入）：

2023/01/01 11:24:07,java,15 //窗口1计算

2023/01/01 11:24:11,java,15 //窗口2计算

2023/01/01 11:24:08,java,15 //窗口1计算

2023/01/01 11:24:13,java,15 //窗口2计算,同时watermarks触发窗口1计算

2023/01/01 11:24:17,java,15 //窗口2计算

2023/01/01 11:24:09,java,15 //窗口计算结束,该记录将被丢弃不输出

2023/01/01 11:24:20,java,15 //左闭右开,窗口3计算

2023/01/01 11:24:22,java,15 //窗口3计算

2023/01/01 11:24:23,java,15 //窗口3计算,同时watermarks触发窗口2计算

2023/01/01 11:24:25,java,15 //窗口3计算

2023/01/01 11:24:40,java,15 //窗口4计算,同时watermarks触发窗口3计算

输出结果：

```
wordCountWatermarks.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Word:java,Count:30,WindowTime:[2023/01/01 11:24:00~2023/01/01 11:24:10],EventTime:[2023/01/01 11:24:07, 2023/01/01 11:24:08]
Word:java,Count:45,WindowTime:[2023/01/01 11:24:10~2023/01/01 11:24:20],EventTime:[2023/01/01 11:24:11, 2023/01/01 11:24:13, 2023/01/01 11:24:17]
Word:java,Count:60,WindowTime:[2023/01/01 11:24:20~2023/01/01 11:24:30],EventTime:[2023/01/01 11:24:20, 2023/01/01 11:24:22, 2023/01/01 11:24:23, 2023/01/01 11:24:25]
```

③.AllowedLateness & SideOutput

测试用例（在nc中依次输入）：

2023/01/01 11:24:07,java,15 //窗口1计算

2023/01/01 11:24:11,java,15 //窗口2计算

2023/01/01 11:24:08,java,15 //窗口1计算

2023/01/01 11:24:13,java,15 //窗口2计算,同时watermarks触发窗口1计算

2023/01/01 11:24:23,java,15 //窗口3计算,同时watermarks触发窗口2计算

2023/01/01 11:24:09,java,15 //allowedLateness窗口1重新计算

2023/01/01 11:24:12,java,15 //allowedLateness窗口2重新计算

2023/01/01 11:24:02,java,15 //allowedLateness窗口1重新计算

2023/01/01 11:25:24,java,15 //延迟超过allowedLateness设定的1min

2023/01/01 11:24:05,java,15 //SideOutput输出

2023/01/01 11:24:08,java,15 //SideOutput输出

2023/01/01 11:24:18,java,15 //SideOutput输出

输出结果：

```
wordCountWatermarksAllowedLateness.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Word:java,Count:30,WindowTime:[2023/01/01 11:24:00~2023/01/01 11:24:10],EventTime:[2023/01/01 11:24:07, 2023/01/01 11:24:08]
Word:java,Count:30,WindowTime:[2023/01/01 11:24:10~2023/01/01 11:24:20],EventTime:[2023/01/01 11:24:11, 2023/01/01 11:24:13]
Word:java,Count:45,WindowTime:[2023/01/01 11:24:00~2023/01/01 11:24:10],EventTime:[2023/01/01 11:24:07, 2023/01/01 11:24:08, 2023/01/01 11:24:09]
Word:java,Count:45,WindowTime:[2023/01/01 11:24:10~2023/01/01 11:24:20],EventTime:[2023/01/01 11:24:11, 2023/01/01 11:24:13, 2023/01/01 11:24:12]
Word:java,Count:60,WindowTime:[2023/01/01 11:24:00~2023/01/01 11:24:10],EventTime:[2023/01/01 11:24:07, 2023/01/01 11:24:08, 2023/01/01 11:24:09, 2023/01/01 11:24:02]
Word:java,Count:15,WindowTime:[2023/01/01 11:24:20~2023/01/01 11:24:30],EventTime:[2023/01/01 11:24:23]
```

```
wordCountWatermarksSideOutput.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
(2023/01/01 11:24:05,java,15)
(2023/01/01 11:24:08,java,15)
(2023/01/01 11:24:18,java,15)
```


(2). 性能测试

经测试，该程序响应速度满足预期设计，运行效果良好

六、演示 Demo（必须）

视频演示：

链接：<https://pan.baidu.com/s/1GxvlpcyF-eCIXJweyKQ6sw>

提取码：q9wr

七、项目总结与反思

1. 目前仍存在的问题

(1).Retract计算和故障恢复由于时间问题还没有完成

(2).代码和程序性能的优化

2. 已识别出的优化项

代码和程序性能以及功能

3. 架构演进的可能性

DAG流程设计进一步优化，减少资源消耗，提高效率

4. 项目过程中的反思与总结

通过这个项目，小组成员能够在总体上了解Flink流式计算的实现原理，通过动手实践掌握基本的Flink流式计算框架的编写，实现特定的功能，了解窗口和Watermarks的机制并动手实际和测试，了解Exactly Once的原理和代码配置等等。

总的来说，通过这次课程和项目实践，大家都收获颇丰，希望以后能有更多的机会参加这样类似的课程和项目实践。

参考资料：

1. 青训营课程
2. 项目提供的参考文档以及网上的一些参考资料
3. 小滴课堂“新一代流式计算框架Flink 1.13”视频教程