Project report

小组成员：欧阳山立、李岩

**目录**

1. 问题概述
2. 任务理解
3. 难点分析
4. 整体技术方案
5. 代码的模块化设计思路
6. 优化与改进
7. 可视化

正文

1. **问题概述**

该项目名为股票市场的主力资金流向计算，背景是股票市场的交易环境，大体需求为根据给定的股票市场数据（包括委托表和成交表），在给定的计算规则下，算出指定股票的输出数据（主力净流入，主力流入，主力流出，超大买单成交量，超大买单成交额，超大卖单成交量，

超大卖单成交额，大买单成交量，大买单成交额，大卖单成交量，大卖单成交额，中买单

成交量，中买单成交额，中卖单成交量，中卖单成交额，小买单成交量，小买单成交额，

小卖单成交量，小卖单成交额。），并根据输入的参数Twindow进行分组呈现。

**二、任务理解**

1、输入数据：输入数据包含四个csv文件，分别是am\_order.txt、pm\_order.txt、am\_trade.txt、pm\_trade.txt，其中各表均包含十几个字段，我们需要根据计算逻辑需要，筛选出关键数据，如委托表中的委托索引、委托时间和买卖方标记以及成交表中的卖方委托索引、买方委托索引、成交时间、成交价格、成交量、成交类别和证券代码。对于每笔成交单，我们将会根据其买卖方委托索引去匹配买卖委托时间并进行比较，以将其分类为主动买单或主动卖单（买方委托时间大即为主动买单，卖方委托时间大即为主动卖单）。

2.Twindow：根据Twindow将目标时间区间（9:30 - 11:30、13:00 - 15:00)均分,例如：若Twindow为十分钟，则目标时间区间将会被分为24个小时间区间，第一个小时间区间会是9:30 - 9:40。最后的输出结果将以小时间区间为单位。

3.成交类别：F表示确实成交。

4.区分大小单类别：成交单类别包括超大单、大单、中单、小单，分类规则如下：



（来自project介绍文档project-v1.pdf）

可以预料到这部分的逻辑在代码实现时容易出现疏漏，需重点关注。

1. 输出结果：根据Twindow划分时间区间进行输出，输出内容见“问题概述”。

**三、难点分析**

1.将成交单分为主动买单或主动卖单：在这个过程中，涉及到成交表和委托表的join（或类似join的操作），初步设想是通过缓存委托表或mapjoin的方式实现。然而，可以预想这一过程将消耗大量时间，将成为后续优化的难点。

2.Twindow：Twindow是一个可变参数，根据Twindow对时间进行分区的过程中，区间边界的数据很可能分类有误，在具体实现过程中将需要若干次调试。

3.分类大小单的逻辑可能出现理解错误。

1. **整体技术方案**

am\_hq\_trade\_spot.txt; pm\_hq\_trade\_spot.txt

Cache

am\_hq\_order\_spot.txt

pm\_hq\_order\_spot.txt

PriceMapper

TimeGrouping

PriceReducer

ouputNotGroup.txt

PriceGroupMapper

PriceGroupReducer

outputFinal.txt

在整体方案设计中，有两个job。

在job1中，我们先将两个委托表的数据载入缓存，然后通过PriceMapper解析两个成交表的数据，并完成与缓存中的委托表的join，获取每笔成交单对应的卖单委托时间和买单委托时间，由此我们便可以判断每笔成交单为主动买单或主动卖单。同时，PriceMapper还会根据每笔成交单的tradeTime和Twindow判断每笔成交单所属于的时间区间，并写入进行记录。这个过程调用了TimeGrouping类中的方法，该方法的输入参数为tradeTime，输出参数为为一个键值对，由所属时间区间的顺序序号和对应时间组成。

接着PriceReduce会对每笔成交单进行超大单、大单、中单、小单的划分并根据主动买卖的信息，对每笔主动单进行最终结果（问题概述中的结果）的计算，输出文件outputNotGroup.txt包括每笔成交单所属时间区间以及以每单为单位的计算结果。

在job2中，我们实现的是一个分组求和的过程，根据job1的输出，我们以时间区间为key进行map的输出，在PriceGroupReducer中对属于同一时间区间的成交单计算结果进行求和，便可以得到最终所需要的输出结果。

1. **代码的模块化设计思路**

在代码实现的过程中，我们发现，买方与卖方的委托索引并无交集，其满足“索引越大，实际委托时间越晚的规律”。因此我们尝试舍弃委托表，仅用成交表来实现项目，并预期这种思路上的转变将大大提升计算效率。

1. PriceMapper：这是job1的mapper类，起到为成交表数据标记所属时间区间以及主动买或卖单的作用。首先，在setup（）中接收两个参数，分布是将要查询的股票代码securityID和Twindow。接着，在map（）中，我们解析成交表的数据，筛选其中将要用到的字段：交易时间、股票代码、买方委托索引、卖方委托索引、成交价格和成交量。然后我们判断买卖方委托索引均不为0后完成刚刚提及的逻辑，“若买方委托索引大于卖方委托索引，则该成交为主动买单，否则相反”。

此map的输入key、value为两个成交表数据，输出key为所属组号,所属时间区间,卖（买）单索引,1(0);输出值为成交价格和成交量。

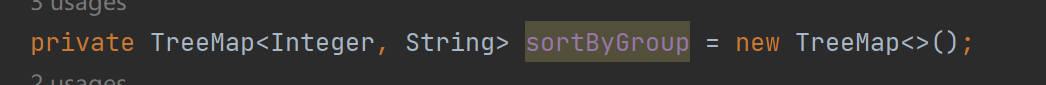
1. TineGrouping：TimeGrouping是我们为了根据交易时间标记所属时间区间而专门设计的列，包含一个属性和一个方法，属性即Twindow，由外界传入，方法名为findTimeGroup（），能够返回一个键值对，由所属时间区间对应的升序序号以及该区间组成。
2. PriceReducer: 经过shuffle之后，每一行输入values包含了某组别下某一主动买（卖）单的各个组成委托所贡献的部分，PriceReducer可以将各部分的成交量和成交额求和，得到该主动单的总成交量和成交额，接着，便可以对此主动单进行分类（超大、大、中、小），并对每个主动单进行目标结果的计算（主力净流入、超大主动卖单成交量等）。

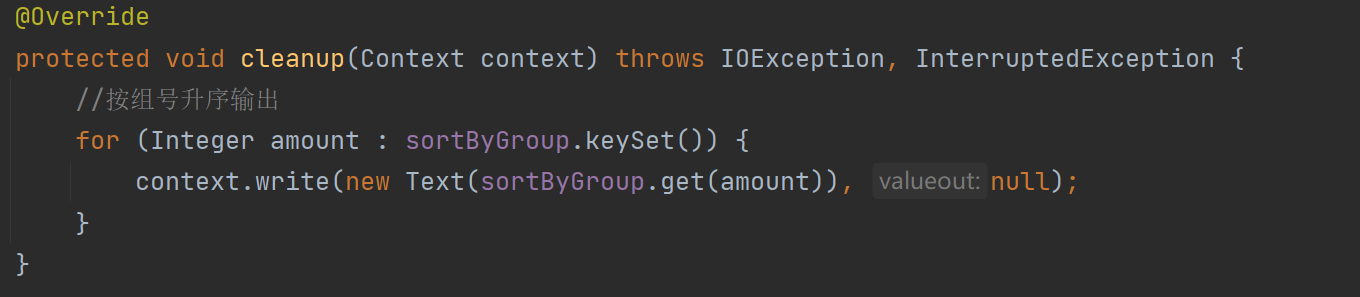
这个reducer的输出key为时间组号，时间区间，输出value为每笔主动单的目标计算结果，输出文件为outputNotGroup.txt。

1. PriceGroupMapper：输入数据是outputNotGroup.txt,输出key是组别信息（组号、时间区间），输出value是对每个主动单计算出的目标结果。
2. PriceGroupReducer：经过shuffle，PriceGroupReducer的values中包含某一组号下所有主动单的目标结果，对这些结果进行求和，便可输出符合需求的最终结果。

同时，PriceGroupReducer中的setup（）为输出文件设置了表头。我们也使用了TreeMap来保证结果按时间顺序输出。







1. **优化与改进**

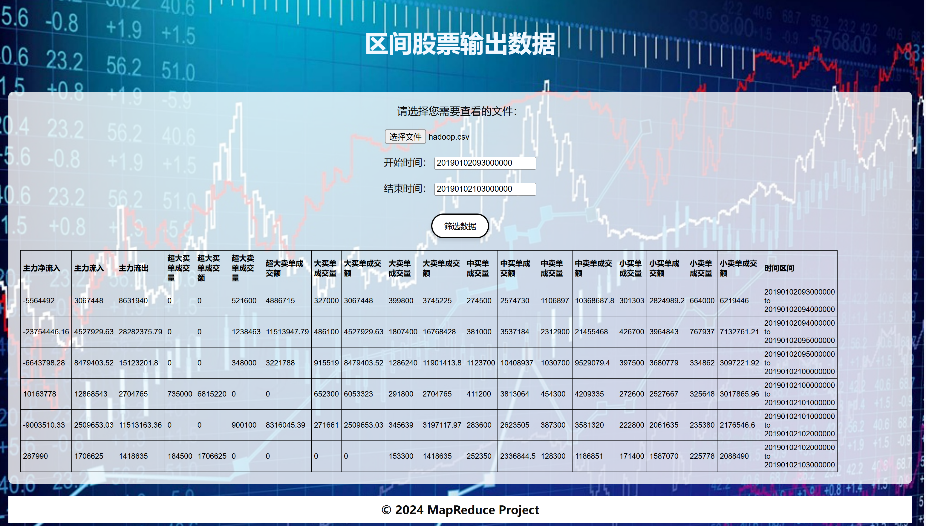
在最初的方案中，我们使用cashe缓存委托表数据，这导致存入缓存以及在PriceMapper中setup读取缓存都会消耗大量时间，在后续逻辑保持前文所述的情况下，计算示例题目（000001、十分钟）所需时间约为三分十秒。

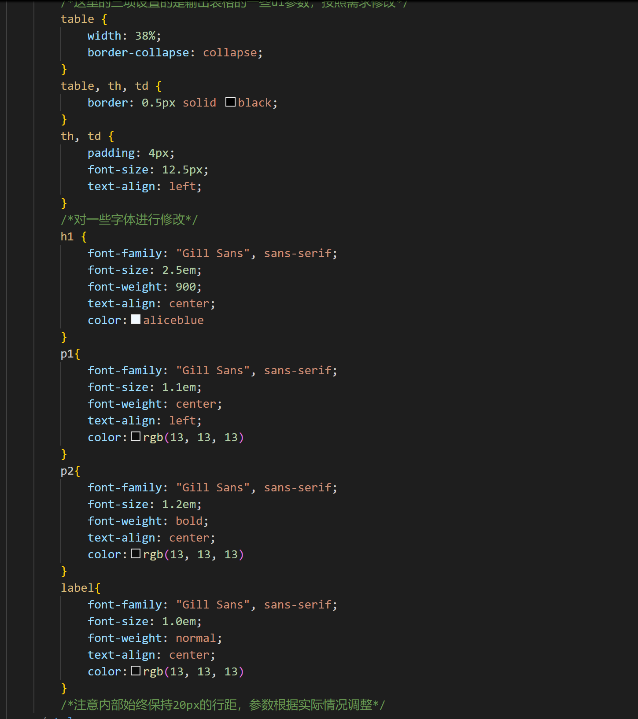
然后，我们希望在保持代码主体逻辑的前提下，在技术方面进行效率提升。于是我们通过自主学习，在力所能及的范围内尝试将两个委托表的缓存及读取过程修改为并行线程，效果并不显著，于是考虑通过观察原始数据的具体特性以求发现新的取巧思路。

最终，我们发现了买卖方委托索引的特征，便利用这一特征成功舍去了从委托表中获取委托时间的步骤，更加高效得得出主动单集合，于是有了上述代码实现过程，运行例题时间约为一分十五秒。

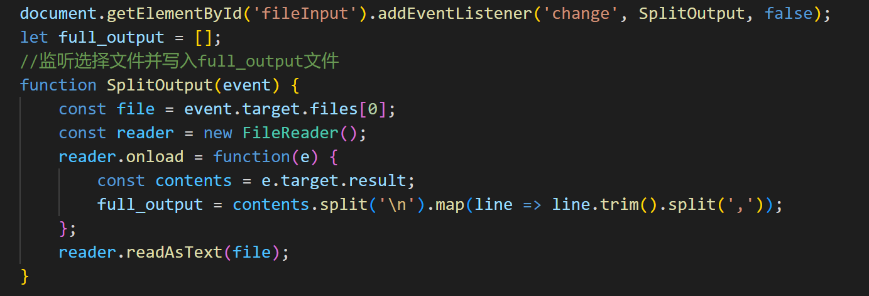
七、可视化

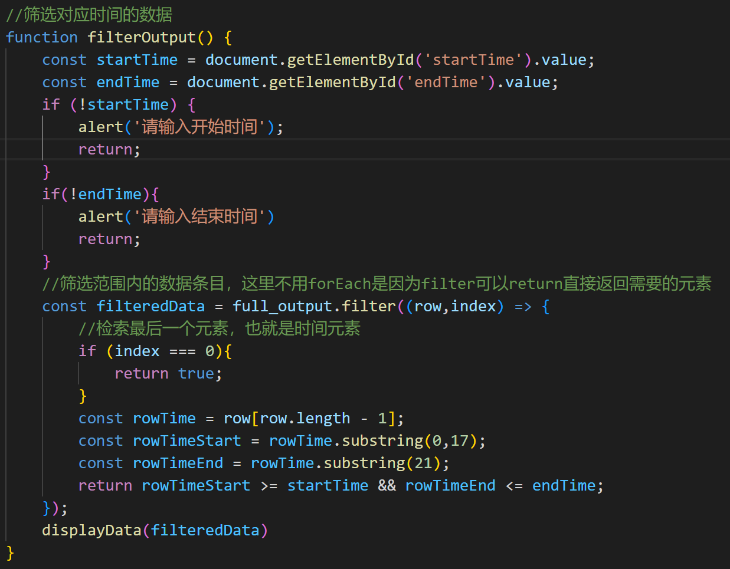
在可视化的部分中，我们采用了写CSS、Java Script的方式，用一个html文件实现了我们的结果展示。在主程序运行得到结果并输出为csv文件后，通过网页打开文件，并根据选择的时间段展示对应时间段内的结果数据。效果示例如下：



第一个部分为CSS编辑页面组件，通过不断调整参数常数得到了比较理想的页面布局和艺术效果：

第二部分向页面添加组件，通过div划分出不同的行，上下保持20px的行距：

最后在script部分，通过三个方法，分别是splitOutput：将读入的csv按“,”分割得到一个二维数组；

filterOutput：对split后的output进行处理，选取在所需时间范围内的数据，并传入最后的plotOut函数；

plotOut：通过循环所有filteredData内的数据并设置为表格内成分（表头、主表），将表格在网页页面显示出来。

经测试展示效果良好。