实验2实验报告

实验任务

任务一：每日资金流入流出统计

根据user\_balance\_table 表中的数据，编写MapReduce程序，统计所有用户每日的资金流入与流出情况。资金流入意味着申购行为，资金流出为赎回行为 。

注：每笔交易的资金流入和流出量分别由字段total\_purchase\_amt 和total\_redeem\_amt 表示。请注意处理数据中的缺失值，将其视为零交易。

输出格式：

<日期> TAB <资金流入量>,<资金流出量>

例如：

20130701 32488348,5525022

设计思路：

DailyCashFlowInOneFile类是一个MapReduce程序的主类，它包含了两个内部类DailyCashFlowMapper和DailyCashFlowReducer。

其中DailyCashFlowMapper类继承自Mapper类，它的作用是将输入的数据按照日期进行分组，然后将每一组的购买金额和赎回金额进行累加。DailyCashFlowMapper类的map方法接收一个键值对作为输入，键是行号，值是一行文本数据，然后将文本数据按逗号分割成多个部分，判断部分的长度是否大于10，如果大于10，则判断第5个部分和第9个部分是否为数字，如果是数字，则将第2个部分作为报告日期，将第5个部分作为购买金额，将第9个部分作为赎回金额，然后将报告日期作为键，购买金额和赎回金额作为值输出。

DailyCashFlowReducer类继承自Reducer类，它的作用是将每一组的购买金额和赎回金额进行累加，然后将结果输出。DailyCashFlowReducer类的reduce方法接收一个键值对作为输入，键是报告日期，值是购买金额和赎回金额的列表，然后将购买金额和赎回金额进行累加，然后将报告日期和累加后的购买金额和赎回金额输出。

在main方法中，首先创建一个Configuration对象conf，然后创建一个Job对象job，设置job的主类为DailyCashFlowInOneFile，设置job的Mapper类为DailyCashFlowMapper，设置job的Reducer类为DailyCashFlowReducer，设置job的输出键类型为Text，设置job的输出值类型为Text，设置job的输入路径为args[0]，设置job的输出路径为args[1]，最后调用job的waitForCompletion方法执行任务，并根据执行结果返回不同的退出码。

任务二：星期交易量统计

基于任务一的结果，编写MapReduce程序，统计一周七天中每天的平均资金流入与流出情况，并按照资金流入量从大到小排序。

输出格式：

<weekday> TAB <资金流入量>,<资金流出量>

例如：

Sunday 155914552,132427205

设计思路

使用Mapper处理输入文件的每一行，解析日期并按工作日分组，同时提取购买额和赎回额。使用Reducer对每个工作日的所有购买额和赎回额进行汇总，并计算平均值。最终输出每个工作日的平均购买额和平均赎回额。

**Mapper类（WeekdayCashFlowMapper）**：继承自Mapper类，是MapReduce任务的映射阶段。输入是文本行（LongWritable为行号，Text为行内容）。输出键值对的键（Text）是工作日（通过日期字符串转换成），（Text）是购买额和赎回额的字符串，两者通过逗号分隔。map方法：读取每一行数据，假设格式为日期和金额（购买额和赎回额），中间用制表符分隔使用SimpleDateFormat将日期字符串解析为Date对象。将日期转换为对应的星期字符串。将购买额和赎回额分割出来，并以星期字符串为键，购买额和赎回额的组合字符串为值，写入上下文（context）。

**Reducer类（WeekdayCashFlowReducer）**：继承自Reducer类，是MapReduce任务的归约阶段。输入是工作日字符串（Text）和该工作日所有记录的购买额和赎回额的列表（Iterable<Text>）。输出键值对的键是工作日字符串，值是该工作日的平均购买额和平均赎回额的字符串，两者通过逗号分隔。reduce方法：遍历所有具有相同工作日键的值。解析每个值中的购买额和赎回额，累加到总购买额和总赎回额，并记录数量。计算平均购买额和平均赎回额。将结果写入上下文

Main方法与上文一致就不过多赘述了。

任务三：用户活跃度分析

根据user\_balance\_table 表中的数据，编写MapReduce程序，统计每个用户的活跃天数，并按

照活跃天数降序排列。

当用户当日有直接购买（ direct\_purchase\_amt 字段大于0）或赎回行为（ total\_redeem\_amt

字段大于0）时，则该用户当天活跃。

输出格式：

<用户ID> TAB <活跃天数>

例如：

125 24

设计思路：

这段代码是一个Hadoop MapReduce程序，用于分析用户活动数据。程序的目的是计算每个用户的活跃天数，并对用户按照活跃天数进行排序，最后输出排序后的用户ID和活跃天数。其中

**Mapper类（ActivityMapper）**：继承自Mapper类，是MapReduce任务的映射阶段。输入是文本行（LongWritable为行号，Text为行内容）。输出键值对的键（Text）是用户ID（字符串），值（IntWritable）是活跃标记（1表示活跃，0表示不活跃）。

map方法：读取每一行数据，假设格式为以逗号分隔的字段。检查行号是否大于0，以跳过可能的标题行或空行。解析用户ID和总赎回金额。根据总赎回金额是否大于0来判断用户是否活跃输出用户ID和活跃标记。

**Reducer类（ActivityReducer）**：继承自Reducer类，是MapReduce任务的归约阶段。输入是用户ID字符串（Text）和该用户所有记录的活跃标记的列表（Iterable<IntWritable>）。输出键值对的键是用户ID和活跃天数的组合字符串，值是null。reduce方法：遍历所有具有相同用户ID键的值。累加活跃标记，得到活跃天数。将用户ID和活跃天数存储在userActivityMap中。

**cleanup方法**：在归约任务结束时调用，用于输出最终结果。将userActivityMap转换为列表并排序，以得到活跃天数最多的用户。输出排序后的用户ID和活跃天数。

值得注意的是：

在Mapper中，通过检查行号key.get() > 0来跳过可能的标题行或空行

以及在cleanup方法中，对用户按照活跃天数进行排序，我之前尝试过用键值对自动排序将键值颠倒排序但结果排出来是乱序，就用list排序了

任务四：交易行为影响因素分析

用户的交易行为（如：余额宝或银行卡的购买或赎回，用户的消费情况等）受到很多因素的影

响。例如：用户特性（参考用户信息表user\_profile\_table ），当前利率（参考支付宝收益率

表mfd\_day\_share\_interest 以及银行利率表mfd\_bank\_shibor ）。

在上面的三个任务中，我们重点研究了user\_balance\_table 表中的数据。现在，请你从其他的表中自行选取研究对象，通过MapReduce（或其他工具），根据统计结果（也即类似于上面三个任务的结果）阐述某一因素对用户交易行为的影响。

本实验中我试图分析用户的性别、城市、星座与他们一年中的总的交易频率之间的关系。我先用mapreduce统计出用户的交易频率并与用户基本信息表合并到一张表格里面并将星座换成对应的1-12的数（同时我也尝试使用python合并，python的运行速度以及代码量明显少于mapreduce），最后用python统计分析其中性别、城市、星座与其的相关系数得到：

活跃天数与性别的相关性: -0.003964657160354326

活跃天数与城市的相关性: -0.03148156207205856

活跃天数与星座对应数字的相关性: 0.007207529560090674

这三个数都比较接近0，意味着这三个量与交易频率都没有太大的相关性。

这段代码是一个使用Hadoop MapReduce框架的Java程序，目的是合并两个不同格式的数据文件，并在合并过程中对星座信息进行数字映射。以下是设计思路和注意点：

**设计思路：**

**星座映射（ConstellationMap）**：使用一个静态的HashMap来存储星座名称和对应的数字映射。提供一个静态方法getConstellationIndex来根据星座名称获取其对应的数字索引。

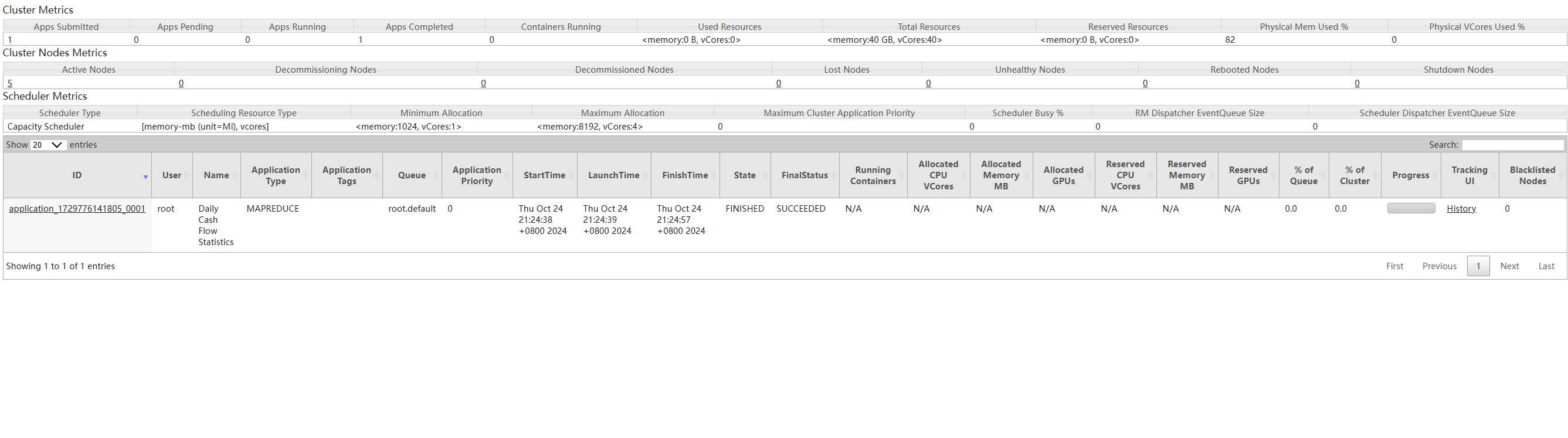
**Mapper类（UserActivityMapper）**：通过setup方法确定当前处理的是哪个文件，根据文件路径中是否包含"second"关键字来判断。根据文件类型（第一个文件或第二个文件），使用不同的分隔符（逗号或制表符）来分割每行数据。提取用户ID作为key，整行数据作为value，写入到context中。

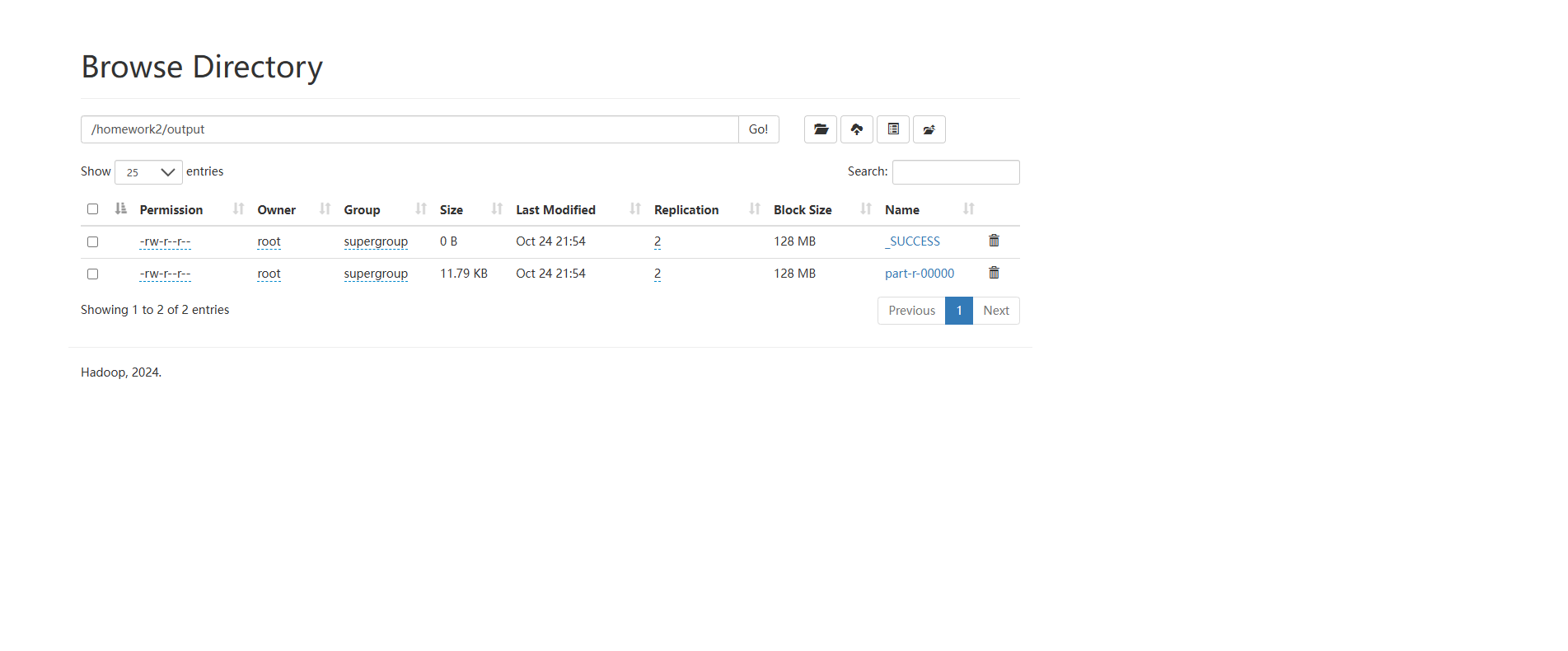
**Reducer类（UserActivityReducer）**：接收来自Mapper的输出，合并相同用户ID的数据。使用StringBuilder来拼接所有相同key的value，形成一个合并后的字符串。遍历合并后的字符串，按行分割，并尝试找到包含星座信息的行。使用ConstellationMap中的映射将星座名称替换为对应的数字。如果找到星座信息，则输出处理后的行；如果未找到，则增加一个计数器。

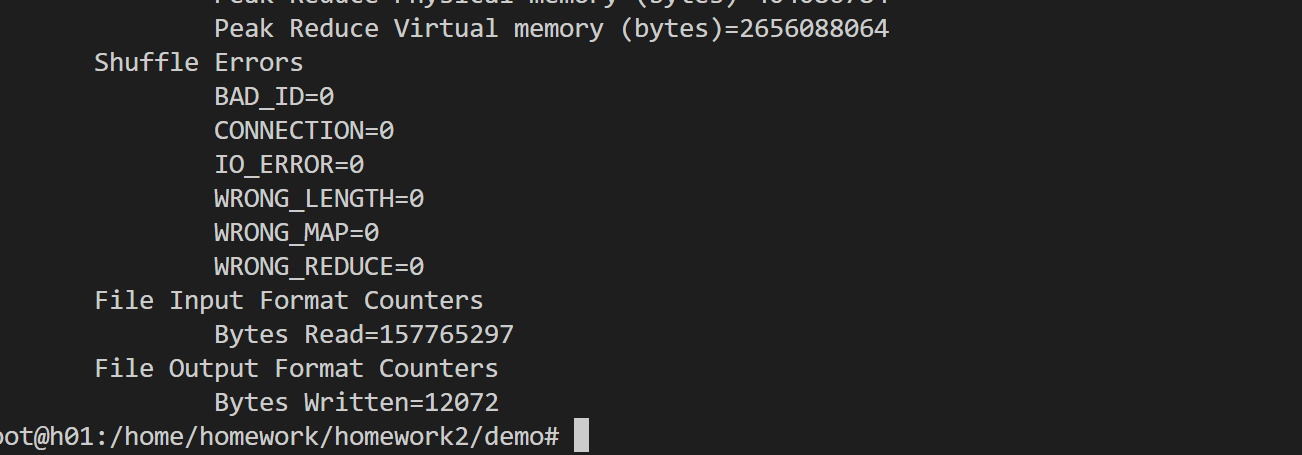
下面是一些调用代码和执行代码成功的终端截图和web截图。

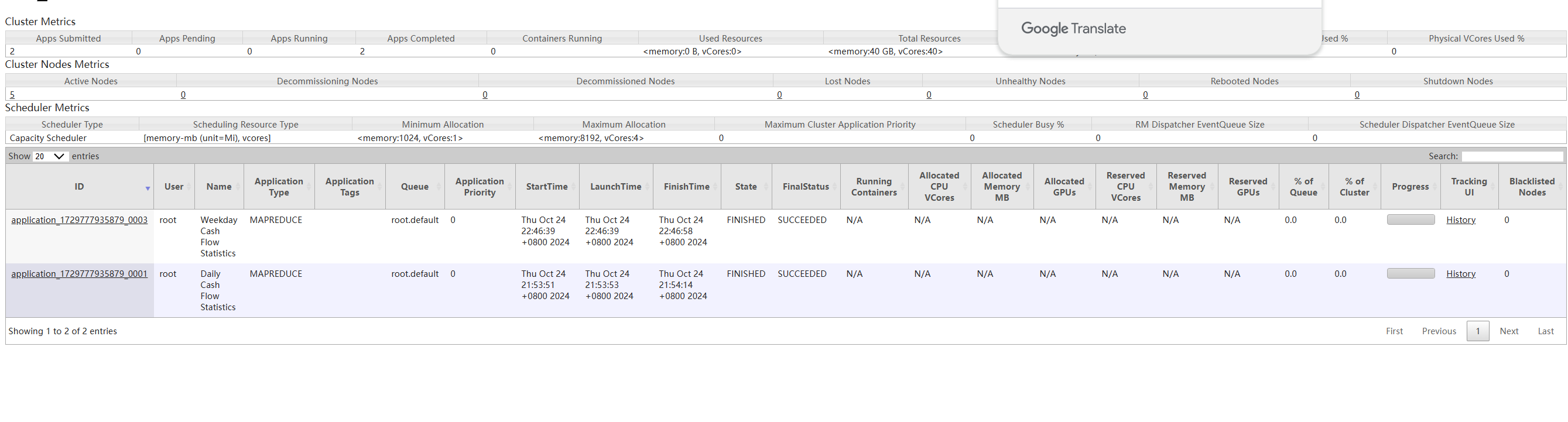
hadoop jar /home/homework/homework2/demo/target/homework1-1.0-SNAPSHOT.jar DailyCashFlowInOneFile /homework/data/user\_balance\_table.csv /home/homework/homework2/demo/src/main/java/first

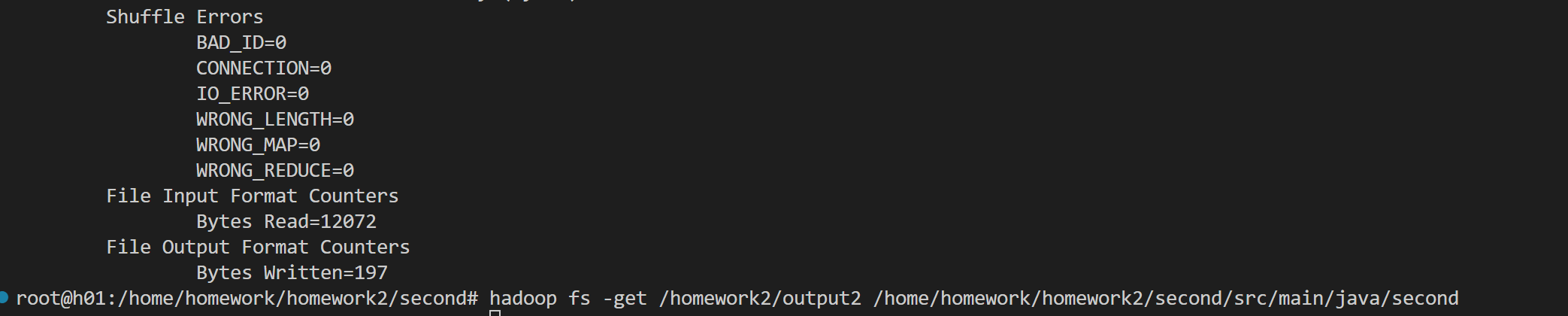
hadoop jar /home/homework/homework2/third/target/homework1-1.0-SNAPSHOT.jar third.UserActivityAnalysis /homework/data/user\_balance\_table.csv homework2/output3

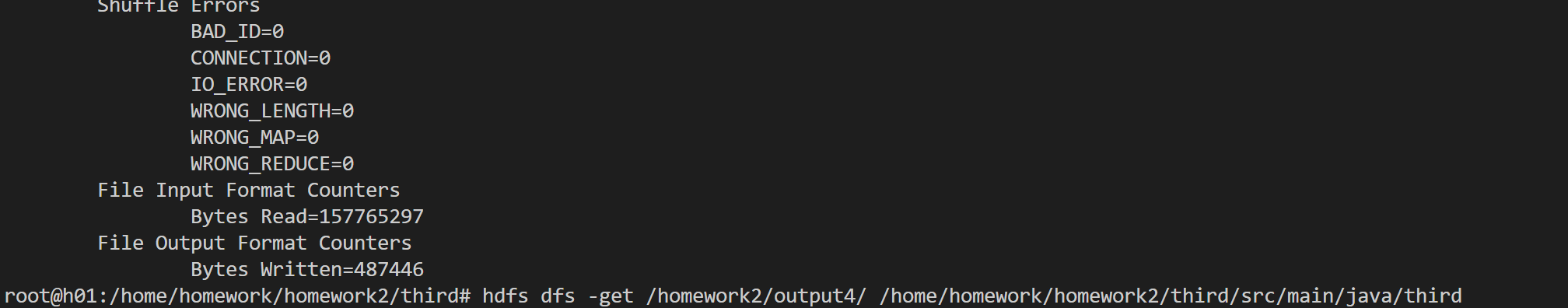


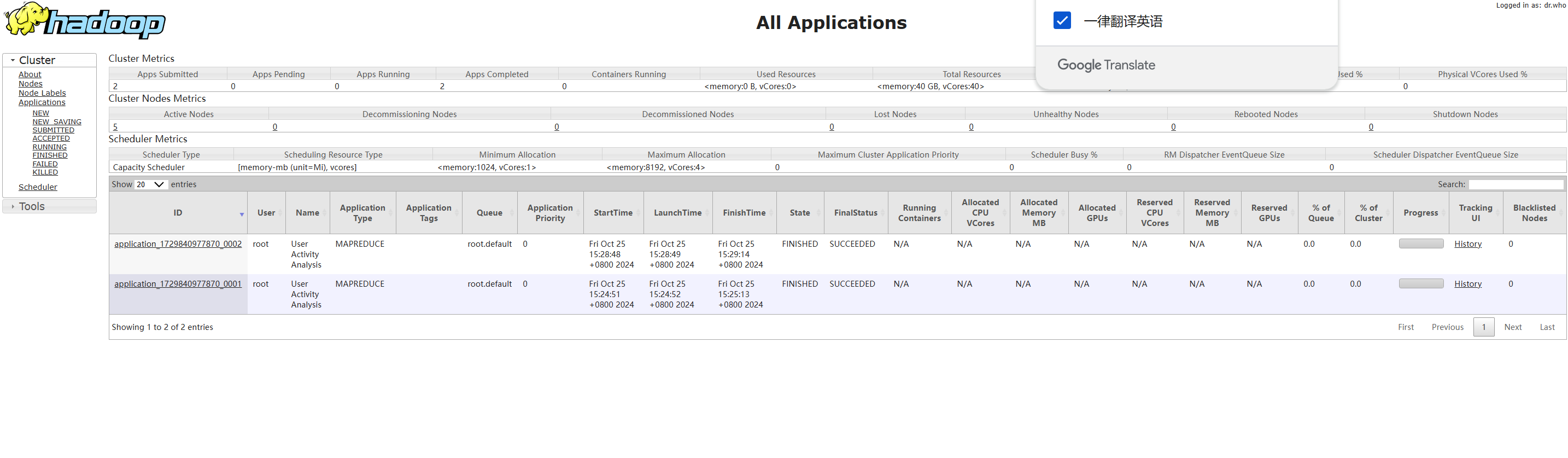












hadoop jar /home/homework/homework2/fourth/target/homework1-1.0-SNAPSHOT.jar UserProfileJoin /homework/data/trade\_times /homework/data/user\_profile\_table.csv /homework2/output5