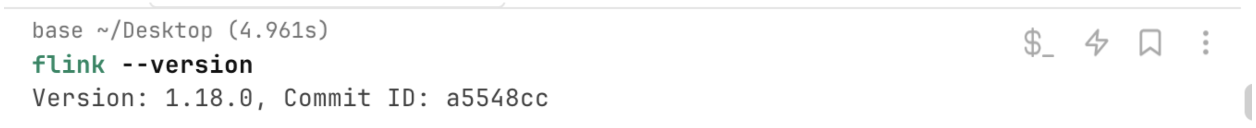
Flink 安装及使用

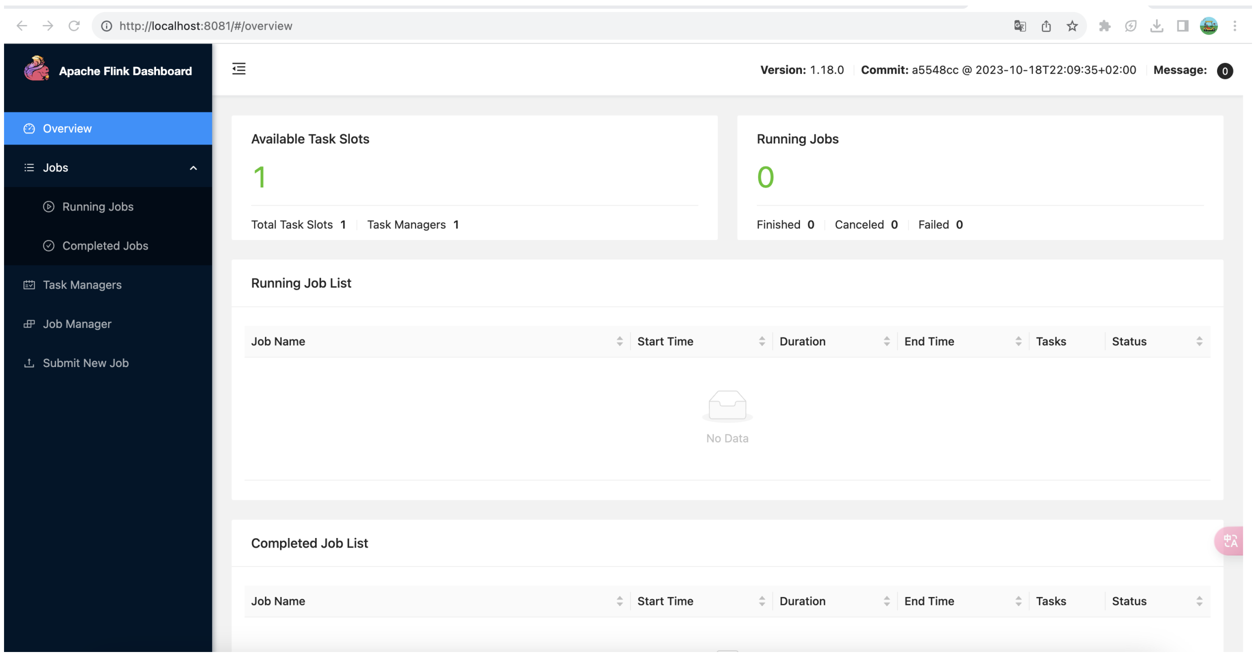
一、安装Flink

- 这里选择使用brew安装，命令为`brew install apache-flink`
- 查看flink版本



```
base ~/Desktop (4.961s)
flink --version
Version: 1.18.0, Commit ID: a5548cc
```

- 启动flink，命令为`/usr/local/Cellar/apache-flink/1.18.0/libexec/bin/start-cluster.sh`
- 检查http://localhost:8081/，可以看到flink已经成功启动



Version: 1.18.0 Commit: a5548cc @ 2023-10-18T22:09:35+02:00 Message: 0

Available Task Slots
1
Total Task Slots 1 Task Managers 1

Running Jobs
0
Finished 0 Canceled 0 Failed 0

Running Job List

Job Name	Start Time	Duration	End Time	Tasks	Status

No Data

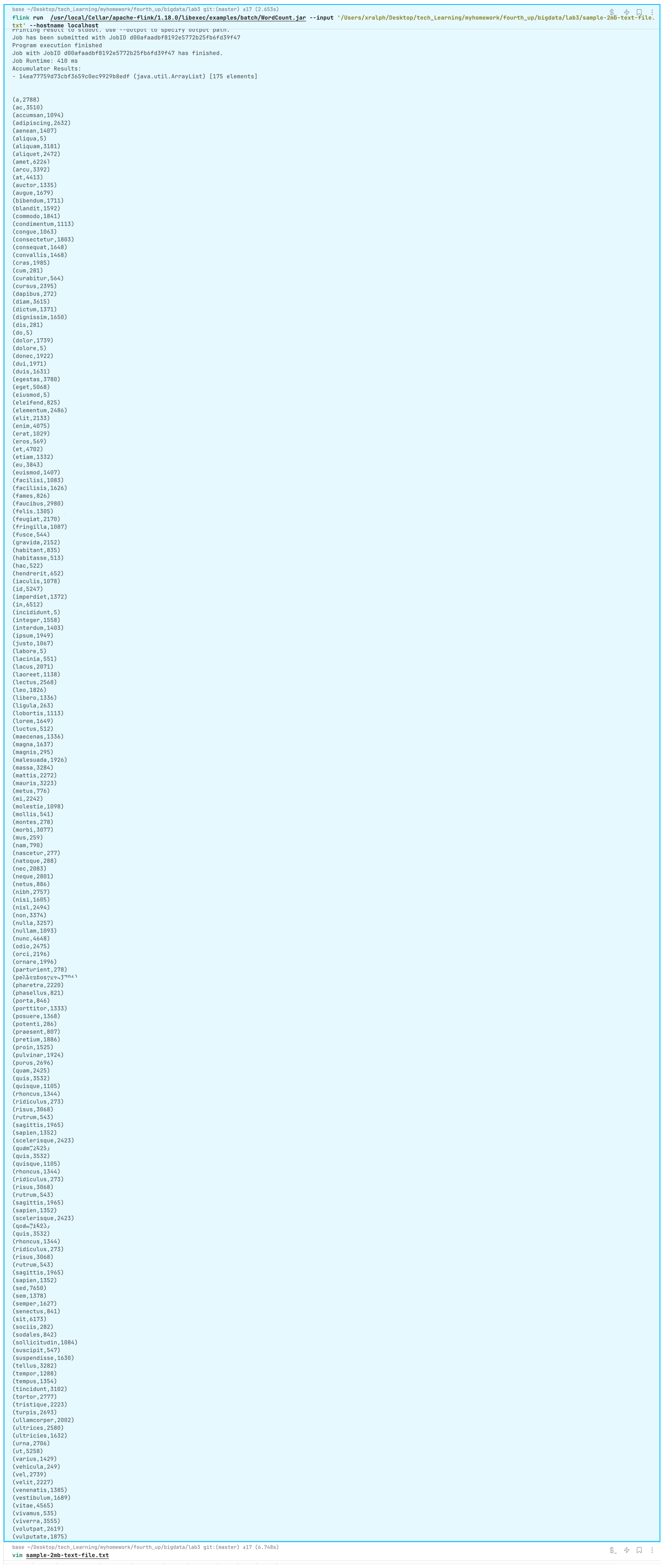
Completed Job List

Job Name	Start Time	Duration	End Time	Tasks	Status

二、运行 `WordCount`

- 直接使用实例中的WordCount，命令为`flink run  /usr/local/Cellar/apache-flink/1.18.0/libexec/examples/batch/WordCount.jar --input '~/Desktop/tech_Learning/myhomework/fourth_up/bigdata/lab3/sample-2mb-text-file.txt' --hostname localhost`

- 结果如下



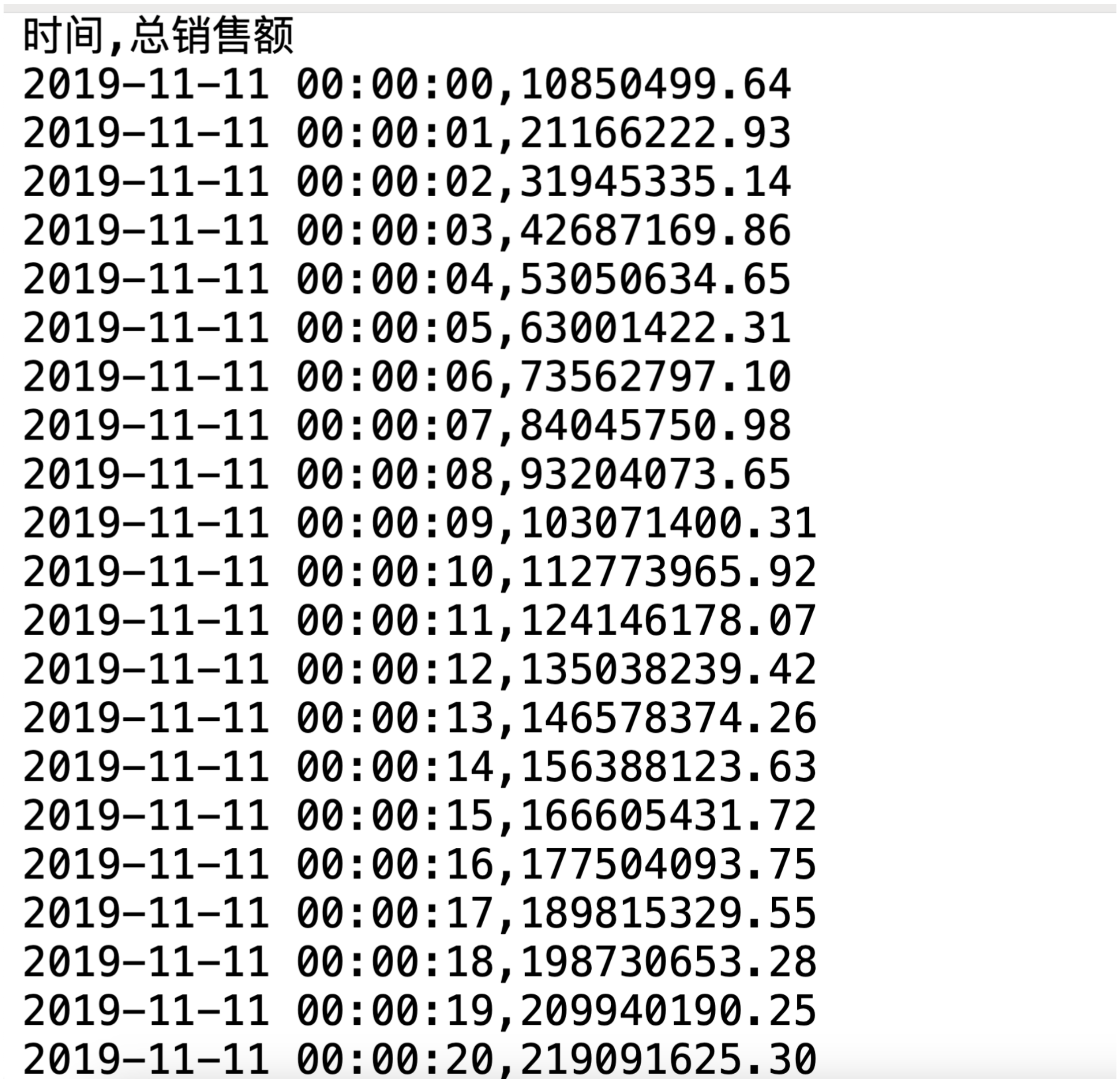
```
base ~/Desktop/tech_Learning/myhomework/fourth_up/bigdata/lab3 git:(master) ±17 (2.653s)
flink run  /usr/local/Cellar/apache-flink/1.18.0/libexec/examples/batch/WordCount.jar --input '/Users/xralph/Desktop/tech_Learning/myhomework/fourth_up/bigdata/lab3/sample-2mb-text-file.
txt' --hostname localhost
Printing result to stdout. Use --output to specify output path.
Job has been submitted with JobID d00afaadbf8192e5772b25fb6fd39f47
Program execution finished
Job with JobID d00afaadbf8192e5772b25fb6fd39f47 has finished.
Job Runtime: 410 ms
Accumulator Results:
- 14ea77759d73cbf3659c0ec9929b8edf (java.util.ArrayList) [175 elements]


(a,2788)
(ac,3510)
(accumsan,1094)
(adipiscing,2632)
(aenean,1407)
(aliqua,5)
(aliquam,3181)
(aliquet,2472)
(amet,6224)
(arcu,3392)
(at,4413)
(auctor,1335)
(augue,1679)
(bibendum,1711)
(blandit,1592)
(commodo,1841)
(condimentum,1113)
(congue,1063)
(consectetur,1803)
(consequat,1648)
(convallis,1468)
(cras,1985)
(cum,281)
(curabitur,564)
(cursus,2395)
(dapibus,272)
(diam,3615)
(dictum,1371)
(dignissim,1650)
(dis,281)
(do,5)
(dolor,1739)
(dolore,5)
(donec,1922)
(dui,1971)
(duis,1631)
(egestas,3780)
(eget,5068)
(eiusmod,5)
(eleifend,825)
(elementum,2486)
(elit,2133)
(enim,4075)
(erat,1029)
(eros,569)
(et,4702)
(etiam,1332)
(eu,3843)
(euismod,1407)
(facilisi,1083)
(facilisis,1626)
(fames,826)
(faucibus,2980)
(felis,1305)
(feugiat,2170)
(fringilla,1087)
(fusce,544)
(gravida,2152)
(habitant,835)
(habitasse,513)
(hac,522)
(hendrerit,652)
(iaculis,1078)
(id,5247)
(imperdiet,1372)
(in,6512)
(incididunt,5)
(integer,1558)
(interdum,1403)
(ipsum,1949)
(justo,1067)
(labore,5)
(lacinia,551)
(lacus,2071)
(laoreet,1138)
(lectus,2568)
(leo,1826)
(libero,1336)
(ligula,263)
(lobortis,1113)
(lorem,1649)
(luctus,512)
(maecenas,1336)
(magna,1637)
(magnis,295)
(malesuada,1926)
(massa,3284)
(mattis,2272)
(mauris,3223)
(metus,776)
(mi,2242)
(molestie,1098)
(mollis,541)
(montes,278)
(morbi,3077)
(mus,259)
(nam,790)
(nascetur,277)
(natoque,288)
(nec,2083)
(neque,2801)
(netus,886)
(nibh,2757)
(nisi,1605)
(nisl,2494)
(non,3374)
(nulla,3257)
(nullam,1093)
(nunc,4648)
(odio,2475)
(orci,2196)
(ornare,1996)
(parturient,278)
(pellentesque,3706)
(pharetra,2220)
(phasellus,821)
(porta,846)
(porttitor,1333)
(posuere,1368)
(potenti,286)
(praesent,807)
(pretium,1886)
(proin,1525)
(pulvinar,1924)
(purus,2696)
(quam,2425)
(quis,3532)
(quisque,1105)
(rhoncus,1344)
(ridiculus,273)
(risus,3068)
(rutrum,543)
(sagittis,1965)
(sapien,1352)
(scelerisque,2423)
(quam,2425)
(quis,3532)
(quisque,1105)
(rhoncus,1344)
(ridiculus,273)
(risus,3068)
(rutrum,543)
(sagittis,1965)
(sapien,1352)
(scelerisque,2423)
(quam,2425)
(quis,3532)
(rhoncus,1344)
(ridiculus,273)
(risus,3068)
(rutrum,543)
(sagittis,1965)
(sapien,1352)
(sed,7650)
(sem,1378)
(semper,1627)
(senectus,841)
(sit,6173)
(sociis,282)
(sodales,842)
(sollicitudin,1084)
(suscipit,547)
(suspendisse,1630)
(tellus,3282)
(tempor,1288)
(tempus,1354)
(tincidunt,3102)
(tortor,2777)
(tristique,2223)
(turpis,2693)
(ullamcorper,2002)
(ultrices,2580)
(ultricies,1632)
(urna,2706)
(ut,5258)
(varius,1429)
(vehicula,249)
(vel,2739)
(velit,2227)
(venenatis,1385)
(vestibulum,1689)
(vitae,4565)
(vivamus,535)
(viverra,3555)
(volutpat,2619)
(vulputate,1875)
base ~/Desktop/tech_Learning/myhomework/fourth_up/bigdata/lab3 git:(master) ±17 (6.748s)
vim sample-2mb-text-file.txt
```

三、统计销售额

- 由于对python不太熟悉，这里使用java实现，代码如下

```
package org.example;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.List;
import java.util.concurrent.atomic.AtomicBoolean;
import org.apache.commons.lang3.StringUtils;
import org.apache.flink.api.common.typeinfo.Types;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.api.java.utils.ParameterTool;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class CountCumulativeSales {
    private static final Logger logger = LoggerFactory.getLogger(CountCumulativeSales.clas
s);

    public static void main(String[] args) throws Exception {
        // 读入输入参数
        ParameterTool parameterTool = ParameterTool.fromArgs(args);
        String input = parameterTool.get("input");
        String output = parameterTool.get("output");
        if (StringUtils.isEmpty(input) && StringUtils.isEmpty(output)) {
            logger.error("Input and output are required");
            return;
        }
        if (!Files.exists(Paths.get(input))) {
            logger.error("File not exists: {}", input);
            return;
        }
        StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironmen
t();
        DataStream<String> inputDataStream = env.readTextFile(input);
        AtomicBoolean first = new AtomicBoolean(true);
        final BigDecimal[] arr = new BigDecimal[] {new BigDecimal("0.0")};
        // 聚合
        DataStream<Tuple2<String, BigDecimal>> salesStream = inputDataStream.filter(value
-> {
            if (first.get()) {
                first.set(false);
                return false;
            }
            return true;
        }).map((value) -> {
            String[] tokens = value.split(",");
            String timestamp = tokens[0];
            double sales = 0.0;
            for (int i = 1; i < tokens.length; i++) {
                sales += Double.parseDouble(tokens[i]);
            }
            return new Tuple2<>(timestamp, sales);
        }).returns(Types.TUPLE(Types.STRING, Types.DOUBLE)).keyBy(0).map(value -> {
            arr[0] = arr[0].add(BigDecimal.valueOf(value.f1));
            return new Tuple2<>(value.f0, arr[0].setScale(2, RoundingMode.HALF_UP));
        }).returns(Types.TUPLE(Types.STRING, Types.BIG_DEC));
        // 输出结果
        List<String> result = salesStream.map(tuple -> tuple.f0 + "," + tuple.f1).executeA
ndCollect(37000);
        Files.write(Paths.get(output), "时间,总销售额\n".getBytes(), StandardOpenOption.CREA
TE,
            StandardOpenOption.WRITE, StandardOpenOption.TRUNCATE_EXISTING);
        Files.write(Paths.get(output), result, StandardOpenOption.APPEND);
    }
}
```

- 由于结果太长，请参见附件中的task3_output.txt文件，这里展示部分文件内容



```
时间,总销售额
2019-11-11 00:00:00,10850499.64
2019-11-11 00:00:01,21166222.93
2019-11-11 00:00:02,31945335.14
2019-11-11 00:00:03,42687169.86
2019-11-11 00:00:04,53050634.65
2019-11-11 00:00:05,63001422.31
2019-11-11 00:00:06,73562797.10
2019-11-11 00:00:07,84045750.98
2019-11-11 00:00:08,93204073.65
2019-11-11 00:00:09,103071400.31
2019-11-11 00:00:10,112773965.92
2019-11-11 00:00:11,124146178.07
2019-11-11 00:00:12,135038239.42
2019-11-11 00:00:13,146578374.26
2019-11-11 00:00:14,156388123.63
2019-11-11 00:00:15,166605431.72
2019-11-11 00:00:16,177504093.75
2019-11-11 00:00:17,189815329.55
2019-11-11 00:00:18,198730653.28
2019-11-11 00:00:19,209940190.25
2019-11-11 00:00:20,219091625.30
```

- 如何运行？
  - 前置要求：JAVA Version：JDK 21，Maven Version : 3.9.5
  - 进入目录`code`中，运行`mvn clean package`，生成对应的jar包
  - 运行`flink run  ./target/code-1.0-SNAPSHOT.jar --input sales_data.txt --output task3_output.txt --hostname localhost`。注意，请将`--input`参数修改为你的电脑中`sales_data.txt`的路径，将`--output`修改为输出文件的路径。

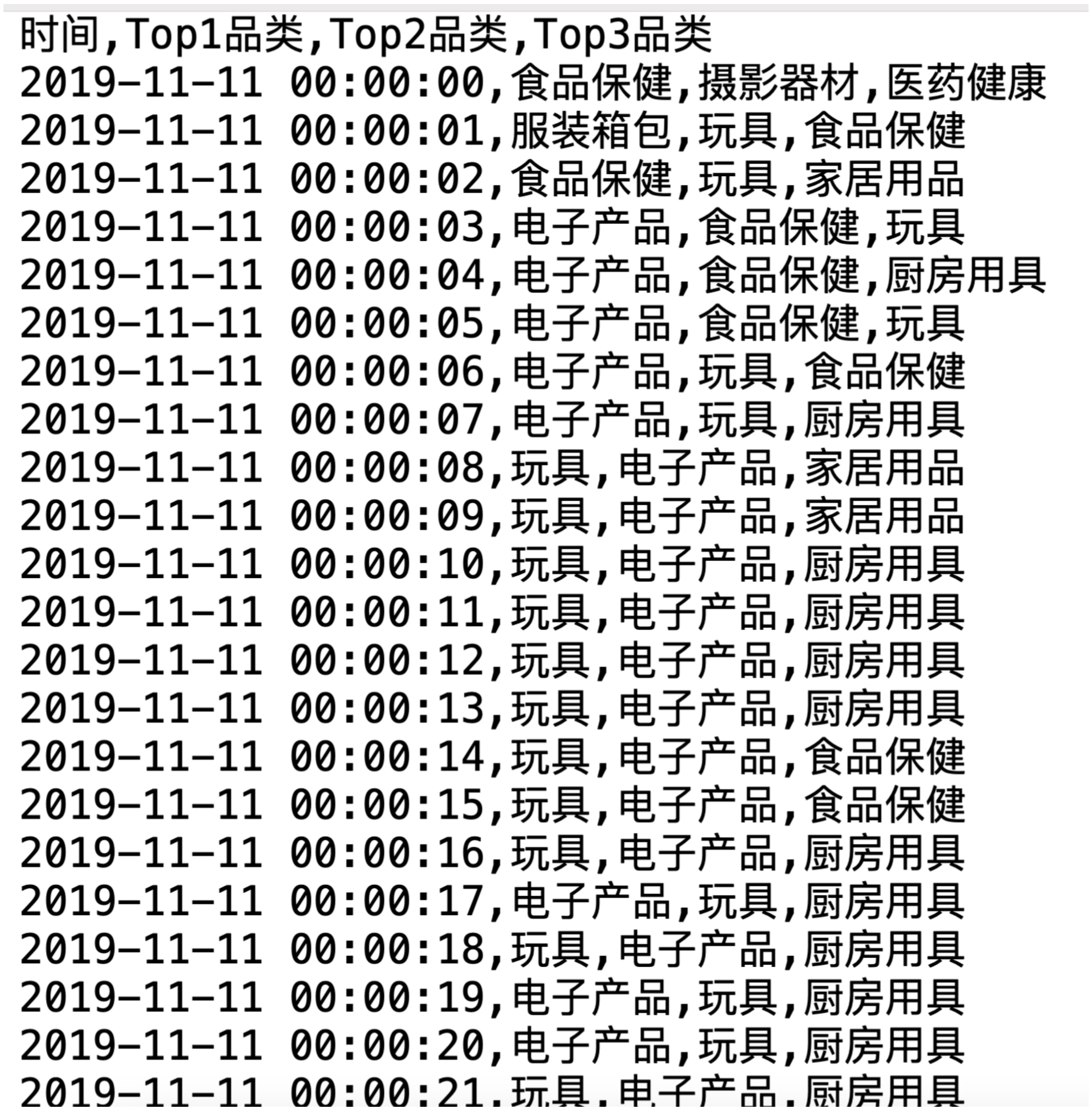
四、统计各个时刻销售额最高的三个品类

- 和三类似，主要添加了排序的内容，代码如下

```
package org.example;

import java.math.BigDecimal;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.stream.Collectors;
import org.apache.commons.lang3.StringUtils;
import org.apache.flink.api.common.typeinfo.Types;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.api.java.utils.ParameterTool;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class CountCumulativeSales {
    private static final Logger logger = LoggerFactory.getLogger(CountCumulativeSales.clas
s);

    public static void main(String[] args) throws Exception {
        ParameterTool parameterTool = ParameterTool.fromArgs(args);
        String input = parameterTool.get("input");
        String output = parameterTool.get("output");
        if (StringUtils.isEmpty(input) && StringUtils.isEmpty(output)) {
            logger.error("Input and output are required");
            return;
        }
        if (!Files.exists(Paths.get(input))) {
            logger.error("File not exists: {}", input);
            return;
        }

        StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironmen
t();
        DataStream<String> inputDataStream = env.readTextFile(input);
        List<String> header = Arrays.stream(
                Files.readAllLines(Paths.get(input), Charset.forName("GB18030")).stream().
findFirst().get().split(","))
```

```
            .map(String::trim).collect(Collectors.toList());
        final AtomicBoolean first = new AtomicBoolean(true);
        final List<BigDecimal> listArr = new ArrayList<>(Collections.nCopies(21, BigDecima
l.valueOf(0.0)));
        DataStream<Tuple2<String, List<String>>> salesStream = inputDataStream.filter(valu
e -> {
            if (first.get()) {
                first.set(false);
                return false;
            }
            return true;
        }).map((value) -> {
            String[] tokens = value.split(",");
            String timestamp = tokens[0];
            List<BigDecimal> list = new ArrayList<>();
            for (int i = 1; i < tokens.length; i++) {
                list.add(BigDecimal.valueOf(Double.parseDouble(tokens[i])));
            }
            return new Tuple2<>(timestamp, list);
        }).returns(Types.TUPLE(Types.STRING, Types.LIST(Types.BIG_DEC))).keyBy(0).map(valu
e -> {
            for (int i = 0; i < listArr.size(); i++) {
                listArr.set(i, listArr.get(i).add(value.f1.get(i)));
            }
            List<String> topThreePositions =
                listArr.stream().sorted(Comparator.reverseOrder()).limit(3).map(listArr::i
ndexOf)
                    .map(index -> header.get(index + 1)).collect(Collectors.toList());
            return new Tuple2<>(value.f0, topThreePositions);
        }).returns(Types.TUPLE(Types.STRING, Types.LIST(Types.STRING)));

        List<String> result =
            salesStream.map(tuple -> tuple.f0 + "," + StringUtils.join(tuple.f1, ",")).exe
cuteAndCollect(37000);
        Files.write(Paths.get(output), "时间,Top1品类,Top2品类,Top3品类\n".getBytes(), Stand
ardOpenOption.WRITE,
            StandardOpenOption.TRUNCATE_EXISTING);
        Files.write(Paths.get(output), result, StandardOpenOption.APPEND);
    }
}
```

- 由于结果太长，请参见附件中的task4_output.txt文件，这里展示部分文件内容



```
时间,Top1品类,Top2品类,Top3品类
2019-11-11 00:00:00,食品保健,摄影器材,医药健康
2019-11-11 00:00:01,服装箱包,玩具,食品保健
2019-11-11 00:00:02,食品保健,玩具,家居用品
2019-11-11 00:00:03,电子产品,食品保健,玩具
2019-11-11 00:00:04,电子产品,食品保健,厨房用具
2019-11-11 00:00:05,电子产品,食品保健,玩具
2019-11-11 00:00:06,电子产品,玩具,食品保健
2019-11-11 00:00:07,电子产品,玩具,厨房用具
2019-11-11 00:00:08,玩具,电子产品,家居用品
2019-11-11 00:00:09,玩具,电子产品,家居用品
2019-11-11 00:00:10,玩具,电子产品,厨房用具
2019-11-11 00:00:11,玩具,电子产品,厨房用具
2019-11-11 00:00:12,玩具,电子产品,厨房用具
2019-11-11 00:00:13,玩具,电子产品,厨房用具
2019-11-11 00:00:14,玩具,电子产品,食品保健
2019-11-11 00:00:15,玩具,电子产品,食品保健
2019-11-11 00:00:16,玩具,电子产品,厨房用具
2019-11-11 00:00:17,电子产品,玩具,厨房用具
2019-11-11 00:00:18,玩具,电子产品,厨房用具
2019-11-11 00:00:19,电子产品,玩具,厨房用具
2019-11-11 00:00:20,电子产品,玩具,厨房用具
2019-11-11 00:00:21,玩具,电子产品,厨房用具
```

- 如何运行？
  - 前置要求：JAVA Version：JDK 21，Maven Version : 3.9.5
  - 进入目录 `code_task4` 中，运行 `mvn clean package` , 生成对应的jar包
  - 运行 `flink run ./target/code-1.0-SNAPSHOT.jar --input sales_data.txt --output task4_output.txt --hostname localhost` 。注意，请将 `--input` 参数修改为你的电脑中 `sales_data.txt` 的路径，将 `--output` 修改为输出文件的路径。

五、其他部分

- task3_output.txt和task4_output.txt，分别为我在任务三和任务四中输出的文件
- 附件中的replication.mp4为我演示运行的视频（如果您的电脑无法运行task3/4的话，请查看视频，视频中有编译、运行代码的全过程）