第十章　Flink习题解答

毕倪飞、杨溢

華東師範大學

大纲

- 计算平均学分绩点
- 区间测速
- 泳池水量
- 邮箱别名分配
- 整数转换
- 最大公约数

题目描述

- 要求：**假设每个学生都选修了10学分的课程，现请**根据成绩单和公式计算所有学生的平均学分绩点
- 输入：每行为一个学生选修某课程的成绩信息，包括学生id、课程学分、课程成绩
- 输出：学生id及其总学分和平均学分绩点

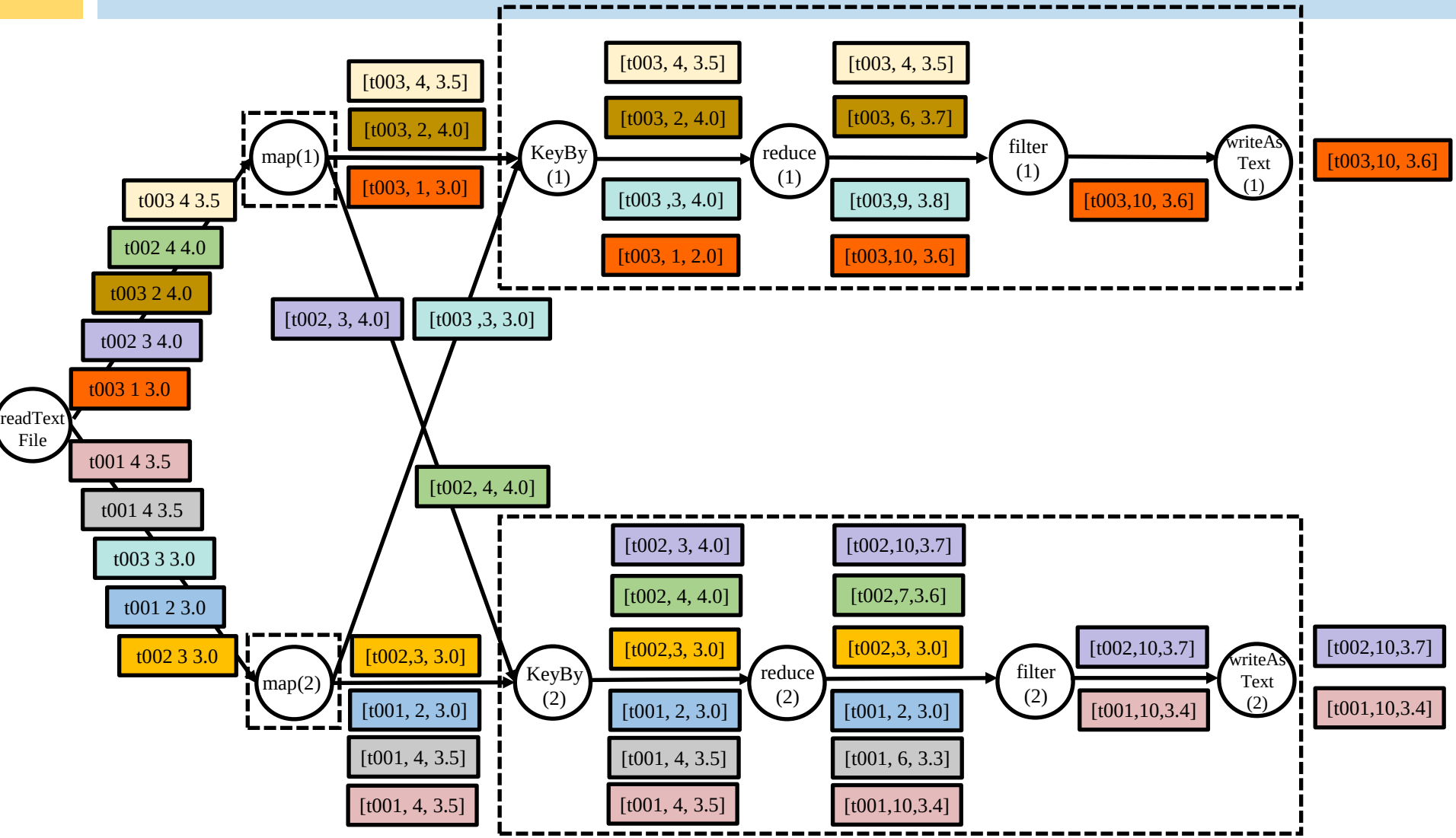
$$\text{平均学分绩点} = \frac{\sum(\text{课程学分} * \text{绩点})}{\sum \text{课程学分}}$$

输入	输出
t001 4 3.5 t003 2 4.0 t003 4 3.5 t001 2 3.0 t002 3 3.0 t002 3 4.0 t001 4 3.5 t002 4 4.0 t003 3 4.0 t003 1 2.0	(t001,10,3.4) (t002,10,3.7) (t003,10,3.6)

解决方案

- map
  - 将每行映射为(学生id, 课程学分, 课程成绩)元组
- keyBy
  - 以学生id为key对元组分组，使每个学生各自的成绩信息被划分到同一个分组内
- reduce
  - 根据平均学分绩点公式，计算得到(学生id，已修学分，平均学分绩点）元组
- filter
  - 筛选出已修学分为10分的元组

计算平均学分绩点的运行过程



编写calculate方法


```java
@Override
public DataStream<Tuple3<String, Integer, Float>> calculate(DataStream<String> text) {
  // 将成绩信息映射为映射为[学生id，课程学分，课程成绩]
  DataStream<Tuple3<String, Integer, Float>> points = text.map(
      new MapFunction<String, Tuple3<String, Integer, Float>>() {
        @Override
        public Tuple3<String, Integer, Float> map(String value) throws Exception {
          String infos[] = value.split(" ");
          float point = Float.parseFloat(infos[2]);
          if (point >= 0 && point <= 5) {
            return new Tuple3<>(
                infos[0], Integer.parseInt(infos[1]), point);
          }
          // 成绩若为-1,表示该成绩异常，此时该科成绩以0分计。
          return new Tuple3<>(
              infos[0], Integer.parseInt(infos[1]), point + 1);
        }
      });
  // 将学生成绩信息按学生id进行分组,分组后计算每个学生总学分及平均学分绩点
  DataStream<Tuple3<String, Integer, Float>> pointsGroupByKey = points.keyBy(0).reduce(
      new ReduceFunction<Tuple3<String, Integer, Float>>() {
        @Override
        public Tuple3<String, Integer, Float> reduce(Tuple3<String, Integer, Float> preRecord,
            Tuple3<String, Integer, Float> currRecord) throws Exception {
          return Tuple3.of(preRecord.f0, preRecord.f1 + currRecord.f1,
              (float) (Math.round(
                  ((preRecord.f1 * preRecord.f2 + currRecord.f1 * currRecord.f2) / (preRecord.f1
                      + currRecord.f1)) * 10)) / 10);
        }
      });
  //筛选总学分为10的成绩记录
  DataStream<Tuple3<String, Integer, Float>> result = pointsGroupByKey
      .filter((value) -> value.f1 == 10);
  // 返回计算结果
  return result;
}
```

大纲

- 计算平均学分绩点
- 区间测速
- 泳池水量
- 邮箱别名分配
- 整数转换
- 最大公约数

题目描述

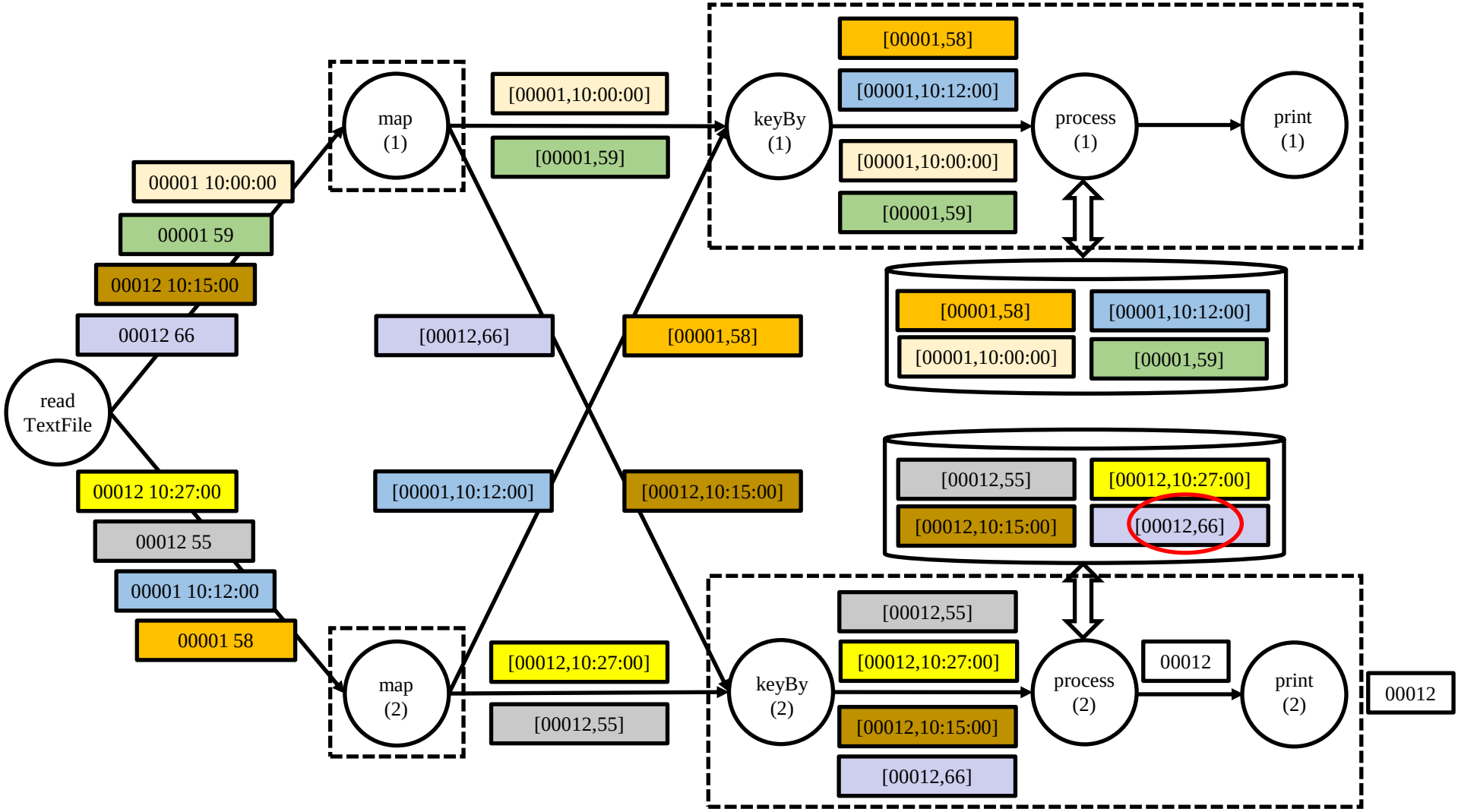
- 要求：10km路程的路段内，平均速度 ≤ 60km/h，途中两测速点瞬时速度 ≤ 60km/h
- 输入：每行文本包含两个字符串，第一个代表车牌号，第二个代表时刻或速度
- 输出：超速的车牌号

输入	输出
00001 10:00:00 00001 58 00001 59 00001 10:12:00 00012 10:15:00 00012 55 00012 66 00012 10:27:00	00012

解决方案

- map
  - 将每一行映射为[车牌号,时刻/速度]键值对
- keyBy
  - 将[车牌号,时刻/速度]键值对按车牌号进行分组
- process
  - 将收到的[车牌号,时刻/速度]键值对中的时刻/速度值存入状态，并判断状态中是否存满4个值
  - 若未存满4个值，则不做任何处理
  - 若已存满4个值，则计算平均速度，并根据平均速度和瞬时速度判断是否超速

区间测速的运行过程


read TextFile
00001 10:00:00
00001 59
00012 10:15:00
00012 66
00012 10:27:00
00012 55
00001 10:12:00
00001 58
map (1)
map (2)
[00001,10:00:00]
[00001,59]
[00012,66]
[00001,10:12:00]
[00012,10:27:00]
[00012,55]
[00001,58]
[00012,10:15:00]
keyBy (1)
keyBy (2)
process (1)
process (2)
print (1)
print (2)
00012
00012

编写avgSpeed方法


```java
@Override
public float averageSpeed(String[] times) {
  if (times[0].compareTo(times[1]) > 0) {
    float hours = ((timeToSeconds(times[0]) - timeToSeconds(times[1])) * 1.0f / 3600);
    return 10 / hours;
  } else {
    float hours = (timeToSeconds(times[1]) - timeToSeconds(times[0])) * 1.0f / 3600;
    return 10 / hours;
  }
}

/**
 * 将一个时刻转换为秒
 *
 * @param time 代表某一个时刻
 * @return 时刻中含有多少秒
 */
public Integer timeToSeconds(String time) {
  String[] splits = time.split(":");
  return Integer.parseInt(splits[0]) * 3600
      + Integer.parseInt(splits[1]) * 60
      + Integer.parseInt(splits[2]);
}
```

自定义HashMap来维护状态


```java
// 通过一个Map来维护每个Key所对应的状态
Map<String, List<String>> customState;

@Override
public void processElement(Tuple2<String, String> t, Context context, Collector<String> collector)
    throws Exception {
  if (customState == null) {
    customState = new HashMap<>();
  }

  // 取出当前Key对应的状态
  List<String> curState = customState.get(t.f0);
  if (curState == null) {
    curState = new ArrayList<>();
  }
  // 若状态中已包含3个值，则表示加上当前收到的数据后，就已收集到当前车牌号所对应的所有数据
  if (curState.size() == 3) {
    // 加上当前收到的数据
    curState.add(t.f1);
    String[] times = new String[2];
    int[] speeds = new int[2];
    int cntTimes = 0, cntSpeeds = 0;
    // 分离出入时刻数据与速度数据
    for (String s : curState) {
      if (s.contains(":")) {
        times[cntTimes++] = s;
      } else {
        speeds[cntSpeeds++] = Integer.parseInt(s);
      }
    }
    // 求区间平均速度
    float avgSpeed = averageSpeed(times);
    if (speeds[0] > 60 || speeds[1] > 60 || avgSpeed > 60) {
      collector.collect(t.f0);
    }
  } else {  // 否则继续将数据收集进状态
    curState.add(t.f1);
    customState.put(t.f0, curState);
  }
}

```

通过ValueState来维度状态

- 编写open方法

```java
private transient ValueState<List<String>> state;

@Override
public void open(Configuration parameters) throws Exception {
  ValueStateDescriptor<List<String>> stateDescriptor = new ValueStateDescriptor<>("s",
      Types.LIST(Types.STRING));
  state = getRuntimeContext().getState(stateDescriptor);
}
```

通过ValueState来维度状态

- 编写processElement方法

```java
@Override
public void processElement(Tuple2<String, String> t, Context context, Collector<String> collector)
    throws Exception {
  if (state.value() == null) {
    state.update(new ArrayList<>());
  }

  // 取出状态
  List<String> curState = state.value();
  // 若状态中已包含3个值，则表示加上当前收到的数据后，就已收集到当前车牌号所对应的所有数据
  if (curState.size() == 3) {
    // 加上当前收到的数据
    curState.add(t.f1);
    String[] times = new String[2];
    int[] speeds = new int[2];
    int cntTimes = 0, cntSpeeds = 0;
    // 分离出入时刻数据与速度数据
    for (String s : curState) {
      if (s.contains(":")) {
        times[cntTimes++] = s;
      } else {
        speeds[cntSpeeds++] = Integer.parseInt(s);
      }
    }
    // 求区间平均速度
    float avgSpeed = averageSpeed(times);
    if (speeds[0] > 60 || speeds[1] > 60 || avgSpeed > 60) {
      collector.collect(t.f0);
    }
  } else {  // 否则继续将数据收集进状态
    curState.add(t.f1);
    state.update(curState);
  }
}
```

大纲

- 计算平均学分绩点
- 区间测速
- 泳池水量
- 邮箱别名分配
- 整数转换
- 最大公约数

题目描述

- 输入：包含操作事件和查询事件的流
  - 格式: (是否查询事件, 泳池编号, 时间戳, 速度)
- 输出：每个查询事件对应的泳池的水量
  - 格式:(泳池编号, 时间戳, 水量)

输入	输出
false,1,0,10 false,2,0,20 true,1,5,0 true,2,5,0 false,1,10,100 false,2,10,0 true,1,20,0 true,2,20,0	1,5,50 2,5,100 1,20,1100 2,20,200

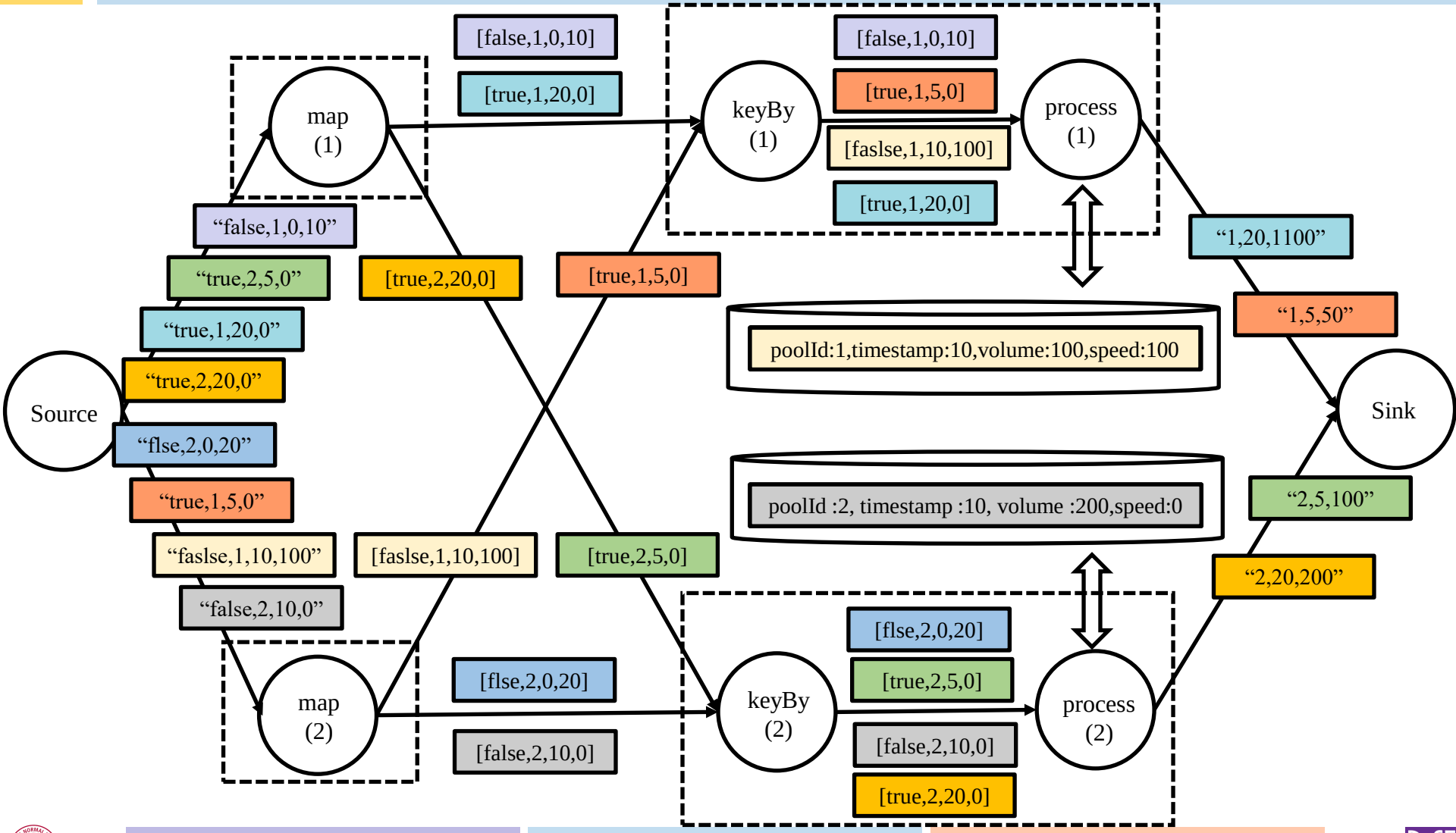
解决方案

- 解析字符串，得到(是否查询事件, 泳池编号, 时间戳, 速度)元组　　map
- 按泳池编号进行分组　　keyBy
- 处理事件　　process
  - 计算当前水量（注意溢出和负数）

$$水量 = 上次水量 + 上次速度 * (时间戳 - 上次时间戳)$$

  - 若为修改事件，则用时间戳、水量和速度来更新时间戳状态、水量状态和速度状态
  - 若为查询事件，则输出结果 (泳池编号,时间戳, 水量)

WaterProblem的运行过程



编写execute方法


```java
@Override
public DataStream<String> execute(DataStream<String> dataStream) {
  return dataStream
      // 把数据格式从String转换到结构体
      .map(line -> {
        String[] items = line.split(",");
        boolean isquery = "true".equals(items[0]);
        int poolId = Integer.parseInt(items[1]);
        int time = Integer.parseInt(items[2]);
        int speed = Integer.parseInt(items[3]);
        return new InputEvent(isquery, poolId, time, speed);
      })
      // 按泳池编号进行分区
      .keyBy(x -> x.poolId)
      // 处理输入事件、维护状态、输出结果
      .process(new ProcessEventFunction());
}

public static class InputEvent {
  boolean isQuery;   // 查询事件为true，修改事件为false
  int poolId;// 泳池编号
  int time;// 事件戳
  int speed; // 速度
  public InputEvent(boolean isQuery, int poolId, int time, int speed) {
    this.poolId = poolId;
    this.time = time;
    this.speed = speed;
    this.isQuery = isQuery;
  }
}
```

编写ProcessEventFunction类


```java
private static class ProcessEventFunction extends
    KeyedProcessFunction<Integer, InputEvent, String> implements Serializable {
  // 记录上次修改事件发生时的水量
  private transient ValueState<Long> volume;
  // 记录上次修改事件的速度
  private transient ValueState<Integer> speedState;
  // 记录上次修改事件的时间戳
  private transient ValueState<Integer> timeState;

  public ProcessEventFunction() {}

  @Override
  public void open(Configuration parameters) throws Exception {
    // 在open函数中注册状态
    ValueStateDescriptor<Long> volumeDescriptor = new ValueStateDescriptor<>("v", Types.LONG);
    volume = getRuntimeContext().getState(volumeDescriptor);
    ValueStateDescriptor<Integer> inSpeedDescriptor = new ValueStateDescriptor<>("is", Types.INT);
    speedState = getRuntimeContext().getState(inSpeedDescriptor);
    ValueStateDescriptor<Integer> timeDescriptor = new ValueStateDescriptor<>("t", Types.INT);
    timeState = getRuntimeContext().getState(timeDescriptor);
  }

  void initStates() throws IOException {
    if (volume.value() == null) volume.update(0L);
    if (speedState.value() == null)  speedState.update(0);
    if (timeState.value() == null)  timeState.update(0);
  }

  @Override
  public void processElement(InputEvent inputEvent, Context context,
      Collector<String> collector) throws Exception {
      ......
  }
}
```

编写processElement方法


```java
@Override
public void processElement(InputEvent inputEvent, Context context,
    Collector<String> collector) throws Exception {
  // 对状态进行初始化
  initStates();

  // 计算当前水量，因为水量可能会超出int的取值范围，所以要使用long
  long currentVolume = volume.value() +
      (long) speedState.value() * (inputEvent.time - timeState.value());

  // 水量不能是负的
  currentVolume = Math.max(0, currentVolume);

  if (inputEvent.isQuery) {
    // 如果是查询事件，就输出查询结果
    collector.collect(inputEvent.poolId + "," + inputEvent.time + "," + currentVolume);
  } else {
    // 如果是修改事件，就更新各个状态
    volume.update(currentVolume);
    speedState.update(inputEvent.speed);
    timeState.update(inputEvent.time);
  }
}
```

注意溢出和负数

大纲

- 计算平均学分绩点
- 区间测速
- 泳池水量
- 邮箱别名分配
- 整数转换
- 最大公约数

题目描述

- 要求：根据别名分配规则以及顺序的别名申请或注销请求序列，计算各请求的状态（成功或失败）
  - 别名长度的取值范围是[5,11]，且别名只能由英文字母、数字或下划线（_）组成
  - 同一学院内的别名不能发生重复
  - 不同学院间的别名可以发生重复

题目描述

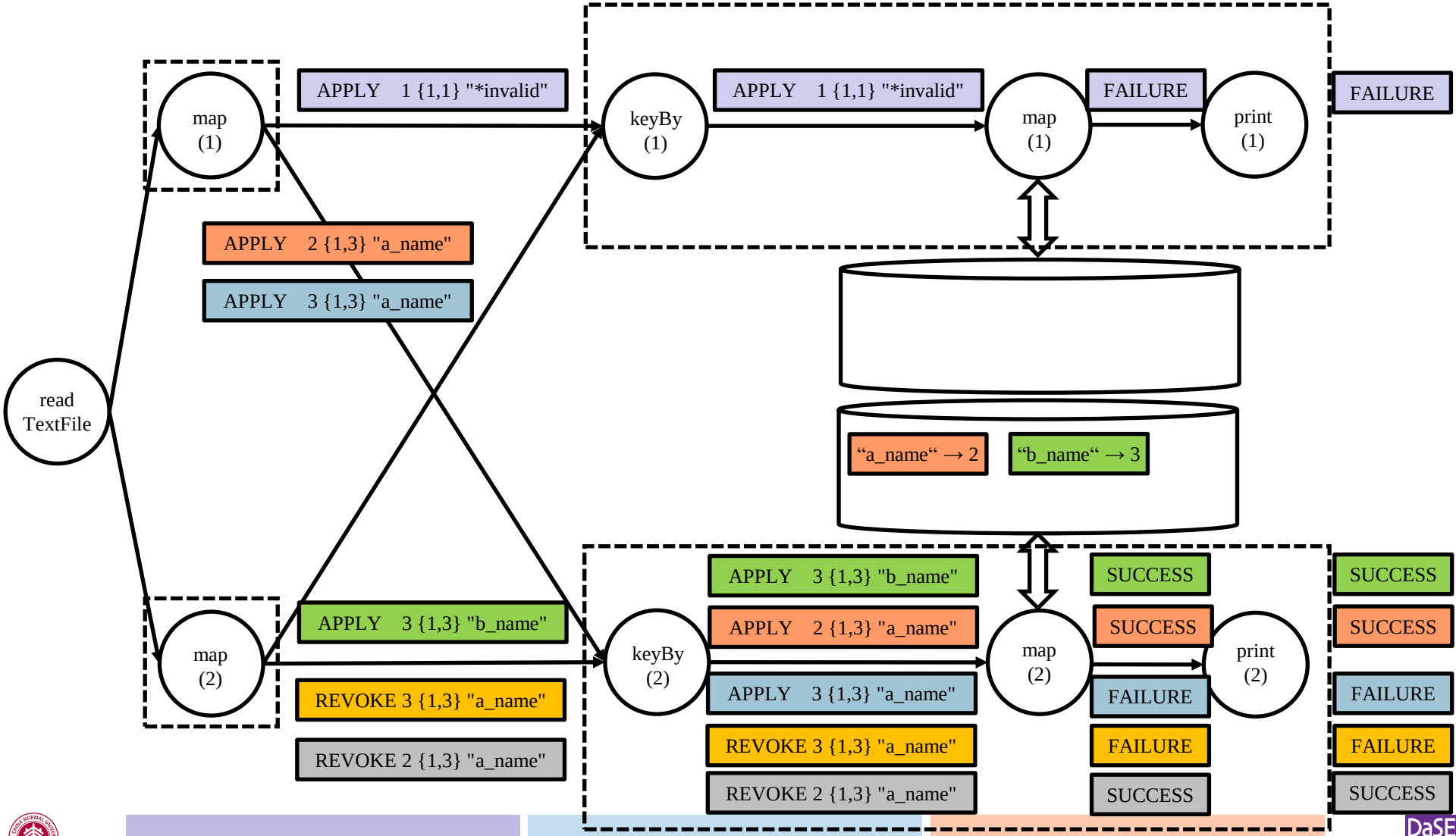
- 输入：由请求类型type、用户工号id、用户所属学院depart以及请求的别名组成
- 输出：每个请求的完成状态，“SUCCESS”表示请求成功执行，“FAILURE”表示请求执行失败

输入	Request{type=APPLY, id=1, depart=Department{1,1}, alias="*invalid"} Request{type=APPLY, id=2, depart=Department{1,3}, alias="a_name"} Request{type=APPLY, id=3, depart=Department{1,3}, alias="b_name"} Request{type=APPLY, id=3, depart=Department{1,3}, alias="a_name"} Request{type=REVOKE, id=3, depart=Department{1,3}, alias="a_name"} Request{type=REVOKE, id=2, depart=Department{1,3}, alias="a_name"}
输出	FAILURE SUCCESS SUCCESS FAILURE FAILURE SUCCESS

解决方案

- keyBy
  - 按学院对请求进行分组，需要自定义哈希函数
- map：使用MapState计算请求完成状态
  - 若别名格式不符合要求，直接判定失败
  - 若是申请别名请求：
    - 检测状态中是否存在别名，若存在则判定失败，否则判定成功，并向MapState中存入[别名,工号]
  - 若是注销别名请求：
    - 检测状态中是否存在与当前请求对应的[别名,工号]，若是则判定成功并从状态中删除当前别名，否则判定失败

运行过程


read TextFile
map (1)
map (2)
APPLY 1 {1,1} "*invalid"
APPLY 2 {1,3} "a_name"
APPLY 3 {1,3} "a_name"
APPLY 3 {1,3} "b_name"
REVOKE 3 {1,3} "a_name"
REVOKE 2 {1,3} "a_name"
keyBy (1)
keyBy (2)
APPLY 1 {1,1} "*invalid"
map (1)
FAILURE
print (1)
FAILURE
"a_name" → 2
"b_name" → 3
APPLY 3 {1,3} "b_name"
APPLY 2 {1,3} "a_name"
APPLY 3 {1,3} "a_name"
REVOKE 3 {1,3} "a_name"
REVOKE 2 {1,3} "a_name"
map (2)
SUCCESS
SUCCESS
FAILURE
FAILURE
SUCCESS
print (2)
SUCCESS
SUCCESS
FAILURE
FAILURE
SUCCESS

按部门对请求进行分组


```java
public class EmailAssignmentImpl extends EmailAssignment {

  @Override
  public DataStream<String> processRequest(DataStream<Request> requests) {
    return requests
        .keyBy(r -> hash(r.getDepart()))
        // ...
  }

  /**
   * 生成部门的哈希值
   * @param depart 部门
   * @return 对应哈希值
   */
  private int hash(Department depart) {
    return 31 * depart.getFirstLevelCode() + depart.getSecondLevelCode();
  }
}
```

获取MapState


```
public class EmailAssignmentImpl extends EmailAssignment {

  @Override
  public DataStream<String> processRequest(DataStream<Request> requests) {
    return requests
        .keyBy(r -> hash(r.getDepart()))
        .map(new RichMapFunction<Request, String>() {
          // 别名 -> 工号
          private MapState<String, Integer> aliasMap;
          private final Pattern aliasPattern = Pattern.compile("^[_a-zA-Z0-9]{5,11}$");

          @Override
          public void open(Configuration parameters) throws Exception {
            super.open(parameters);
            aliasMap = getRuntimeContext().getMapState(
                new MapStateDescriptor<>("aliasMap", String.class, Integer.class));
          }

          @Override
          public String map(Request request) throws Exception {
            // ...
          }
        });
  }
  // ...
}
```

计算请求完成状态


```java
@Override
public String map(Request request) throws Exception {
  String alias = request.getAlias();
  if (!aliasPattern.matcher(alias).matches()) {
    return "FAILURE";
  }

  if (request.getType() == RequestType.APPLY) {
    if (aliasMap.contains(alias)) {
      return "FAILURE";
    }
    aliasMap.put(alias, request.getId());
    return "SUCCESS";
  } else { // request.getType() == RequestType.REVOKE
    if (aliasMap.contains(alias) && aliasMap.get(alias) == request.getId()) {
      aliasMap.remove(alias);
      return "SUCCESS";
    }
    return "FAILURE";
  }
}
```

大纲

- 计算平均学分绩点
- 区间测速
- 泳池水量
- 邮箱别名分配
- 整数转换
- 最大公约数

题目描述

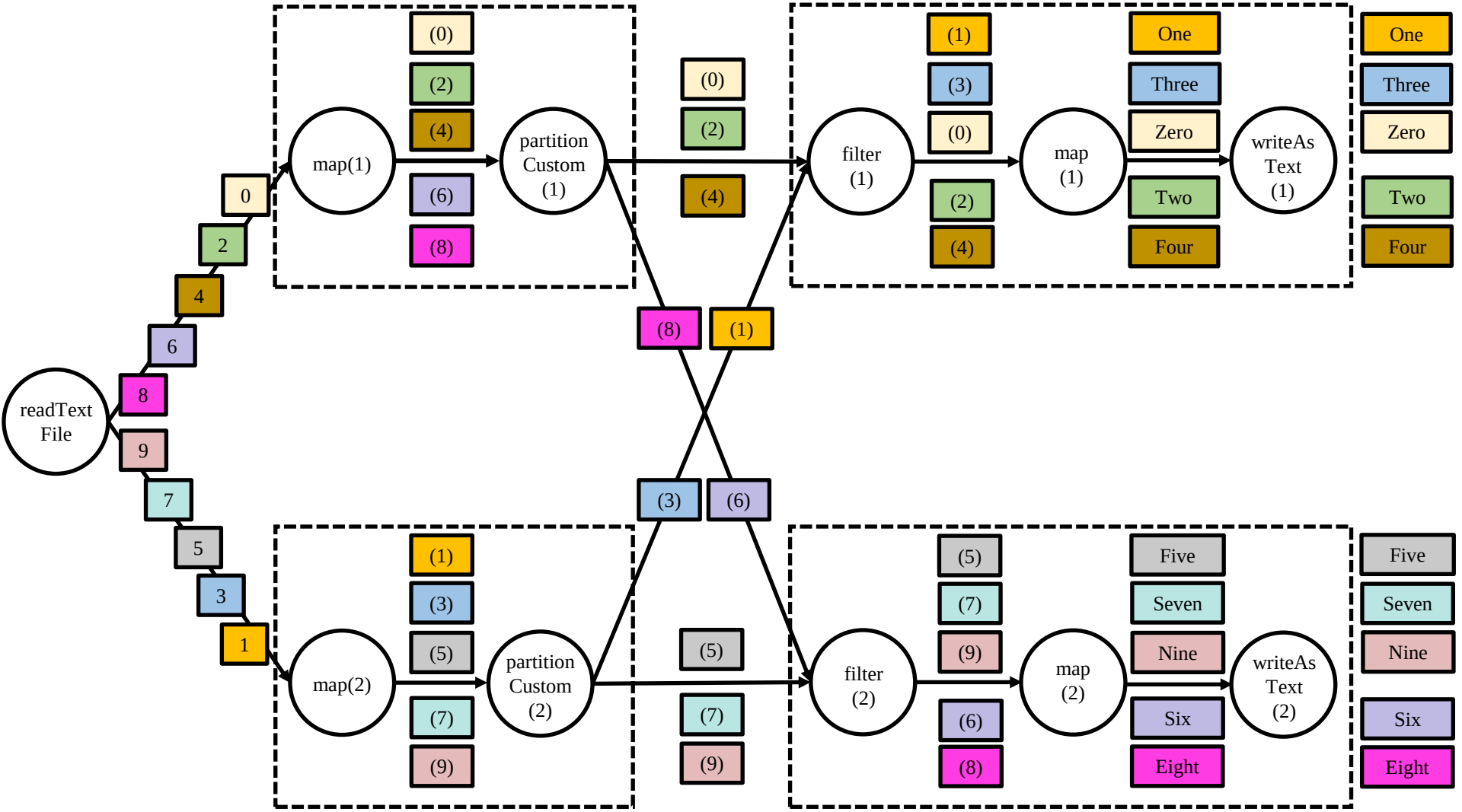
- 要求：某程序会持续地产生一些位于[0, 9]区间的数。这些数均匀地分布在[0, 4]和[5, 9]两个区间内。要求将这些数均匀地划分到两个任务中，并将其中的整数转换为英文单词。
- 输入：每行为一个数
- 输出：整数转换后的结果

输入	输出	
0	任务一输出	任务二输出
1	One	Five
2	Three	Seven
3	Zero	Nine
4	Two	Six
5	Four	Eight
6		
7		
8		
9		

解决方案

- 均匀划分数
  - 通过自定义Partitioner划分数
- 将整数转换为英文单词
  - 通过filter过滤非整数
  - 通过map将整数转换为英文单词

整数转换的运行过程


readText File
0
2
4
6
8
9
7
5
3
1
map(1)
(0)
(2)
(4)
(6)
(8)
partition Custom (1)
map(2)
(1)
(3)
(5)
(7)
(9)
partition Custom (2)
(0)
(2)
(4)
(8)
(1)
(3)
(6)
(5)
(7)
(9)
filter (1)
(1)
(3)
(0)
(2)
(4)
map (1)
One
Three
Zero
Two
Four
writeAs Text (1)
One
Three
Zero
Two
Four
filter (2)
(5)
(7)
(9)
(6)
(8)
map (2)
Five
Seven
Nine
Six
Eight
writeAs Text (2)
Five
Seven
Nine
Six
Eight

编写partition方法


```java
public class DigitalPartitionerImpl extends DigitalPartitioner {

  @Override
  public int partition(String key, int numPartitions) {
    double parsedKey = Double.parseDouble(key);
    // 条件表达式，[0, 4]的数字划分给任务一，而[5, 9]的数字划分给任务二
    return parsedKey >= 0 && parsedKey <= 4 ? 0 : 1;
  }
}
```

编写digitalConversion方法


```java
public class DigitalConversionImpl extends DigitalConversion {

  @Override
  public DataStream<String> digitalConversion(DataStream<Tuple1<String>> digitals) {
    // 过滤掉小数
    DataStream<Tuple1<String>> integer = digitals.filter((FilterFunction<Tuple1<String>>) value
        -> value.getField(0).toString().matches("[0-9]+")).returns(Types.TUPLE(Types.STRING));

    // 将整数转换为对应的单词
    DataStream<String> word = integer.map((MapFunction<Tuple1<String>, String>) value -> {
      // 解析出整数
      int index = Integer.parseInt(value.getField(0));
      // 通过枚举类获取整数对应的英文单词
      return DigitalWord.values()[index].getWord();
    }).returns(Types.STRING);

    return word;
  }
}
```

大纲

- 计算平均学分绩点
- 区间测速
- 泳池水量
- 邮箱别名分配
- 整数转换
- 最大公约数

题目描述

- 要求：给定两个正整数，利用辗转相除法以迭代方式求两者的最大公约数（GCD）

$$gcd(a,\ b) = \begin{cases} \mathbf{gcd}(b, a\ \%\ b)\ ,\ (a\%b)\ !=\ \mathbf{0} \\ b,\ (a\%b)\ =\ \mathbf{0} \end{cases}$$

- 输入：字母标识符以及两个正整数a, b组成的计算请求
- 输出：以格式(字母标识符,除数,余数)输出每个计算请求每轮迭代的除数和余数

输入	输出
A 5 2 B 18 6	(A,2,1) (B,6,0) (A,1,0)

辗转相除法

算法执行过程

计算请求$(A, 5, 2)$的迭代过程

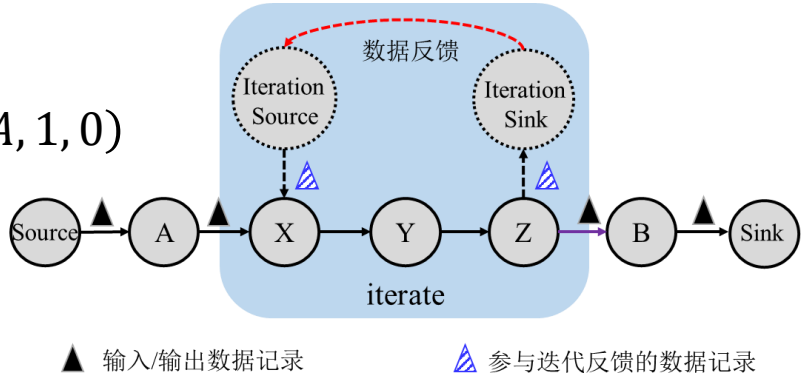
迭代1

$$\gcd(a,b) \rightarrow \gcd(b, a\ \%\ b)$$
$$\gcd(5,2) \rightarrow \gcd(2,1) \longrightarrow (A,2,1)$$

迭代2

$$\gcd(a,b) \rightarrow \gcd(b, a\ \%\ b)$$
$$\gcd(2,1) \rightarrow \gcd(1,0) \longrightarrow (A,1,0)$$

$a\ \%\ b = 0$，不需要再向迭代前端反馈



将$(A, b, a\%b)$作为输出向后传递

若$a\ \%\ b\ !=\ 0$，则将$(A, b, a\%b)$作为下一轮迭代的输入

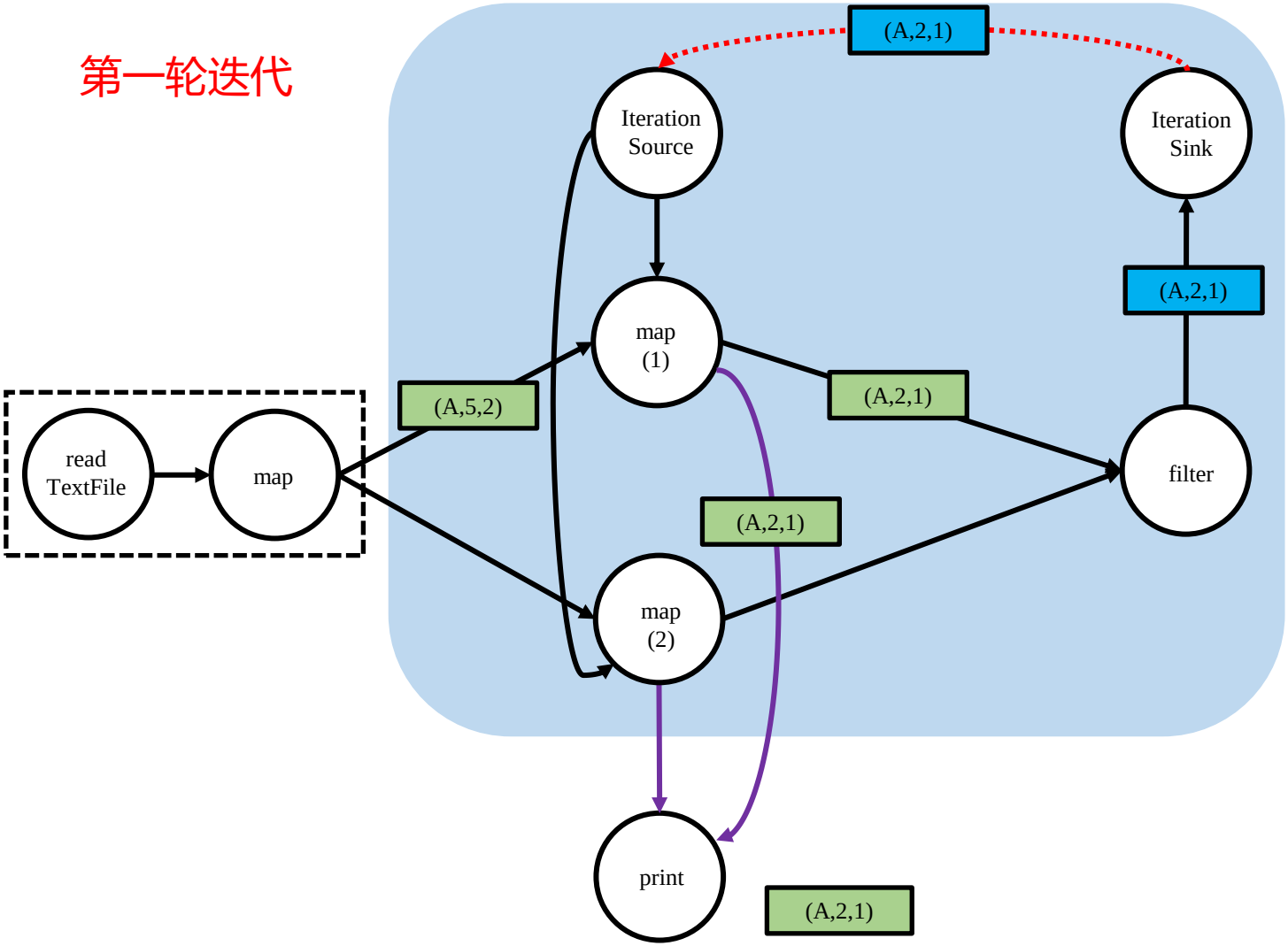
解决方案

- 一轮迭代的处理逻辑
  - 根据(标识符, a, b)计算出(标识符, b, $a\%b$)
    - map
  - 将(标识符, b, $a\%b$)作为输入反馈给迭代前端
    - filter
  - 将(标识符, b, $a\%b$)作为输出向后传递
    - 无需操作

是否需要保证$a > b$？

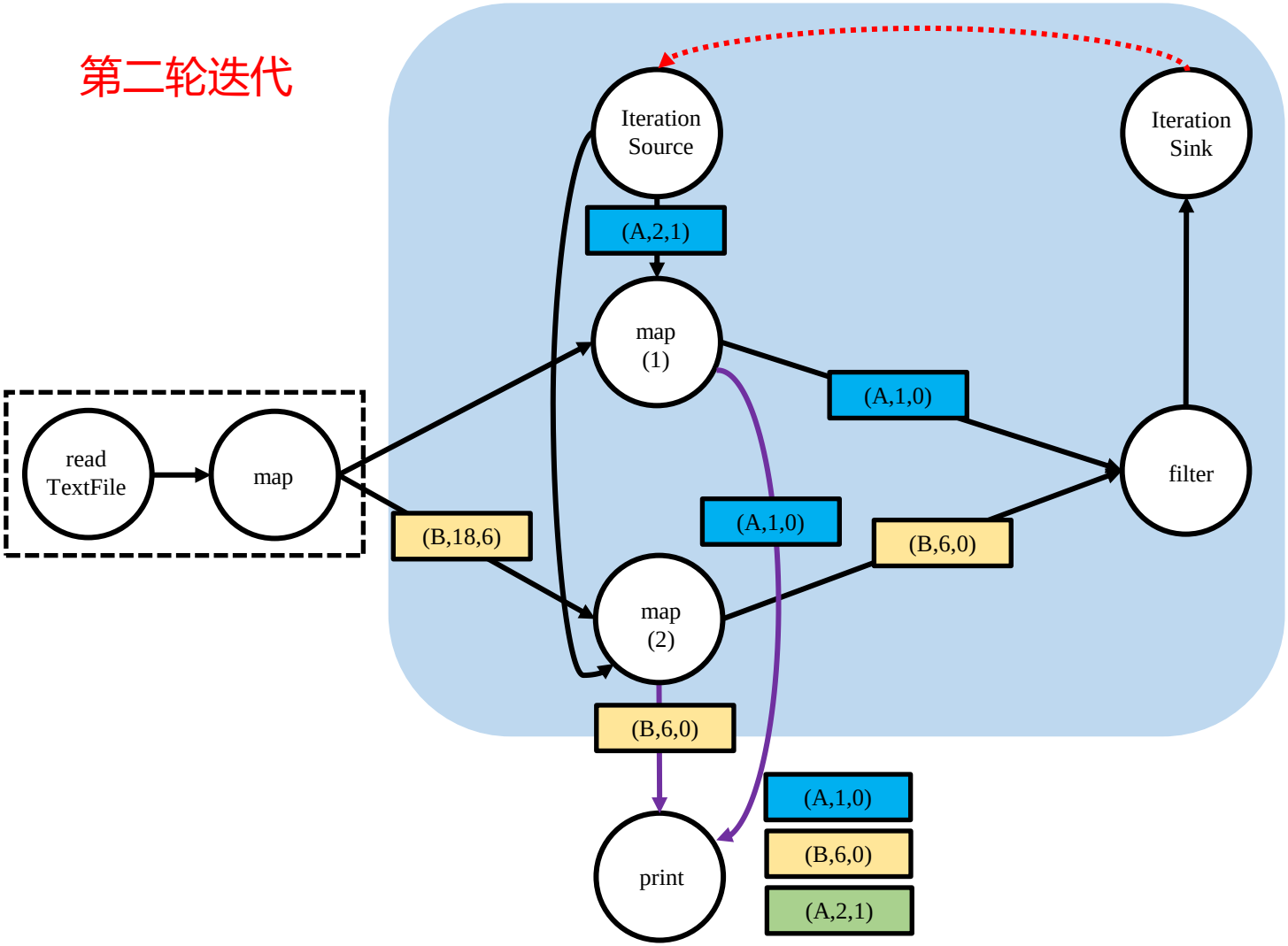
最大公约数的运行过程

第一轮迭代



最大公约数的运行过程

第二轮迭代



编写calGCD方法


```java
public class GCDImpl extends GCD {
  @Override
  public DataStream<Tuple3<String, Integer, Integer>> calGCD(IterativeStream<Tuple3<String, Integer, Integer>>
iteration) {
    // 实现每一轮迭代计算的逻辑
    DataStream<Tuple3<String, Integer, Integer>> stepStream =
        iteration.map(
            new MapFunction<Tuple3<String, Integer, Integer>, Tuple3<String, Integer, Integer>>() {
              @Override
              public Tuple3<String, Integer, Integer> map(Tuple3<String, Integer, Integer> tuple3)
                  throws Exception {
                int a = tuple3.f1;
                int b = tuple3.f2;
                if (a < b) {
                  int temp = a;
                  a = b;
                  b = temp;
                }
                return new Tuple3<>(tuple3.f0, b, a % b);
              }
            });
    // 创建反馈流
    DataStream<Tuple3<String, Integer, Integer>> feedbackStream =
        stepStream.filter(tuple3 -> tuple3.f2 != 0);
    iteration.closeWith(feedbackStream);
    // 创建输出流
    DataStream<Tuple3<String, Integer, Integer>> outputStream = stepStream;
    return outputStream;
  }
}
```

谢谢！Q&A