第四次实验

实验目的

1. 学习基于MPI分布式并行计算程序设计方法。
2. 学习基于MapReduce框架的分布式并行计算程序设计方法。
3. 学习基于Spark框架的分布式并行计算程序设计方法。

实验题目

题目 1

编写基于 MPI 的并行计算程序，实现对连续函数 $f(x)=\sin(x^2+x)(x\in R)$ 在区 间 $[a,b]$ 上的定积分求解。要求使用“梯形法（Trapezoidal Rule）”进行近似计算。

题目 2

输入文件为学生成绩信息，包含了必修课与选修课成绩，格式如下：
班级1, 姓名1, 科目1, 必修, 成绩1 `<br>`（注：`<br>` 为换行符）
班级2, 姓名2, 科目1, 必修, 成绩2 `<br>`
班级1, 姓名1, 科目2, 选修, 成绩3 `<br>`
……, ……, ……, …… `<br>`
编写 Hadoop 平台上的 MapReduce 程序，分别实现如下功能：

1. 计算每个学生必修课的平均成绩。
2. 统计每个班级中所有课程（必修+选修）平均成绩排名前五的学生姓名和成绩

题目 3

输入文件的每一行为具有父子/父女/母子/母女/关系的一对人名，例如：
Tim, Andy `<br>`
Harry, Alice `<br>`
Mark, Louis `<br>`
Andy, Joseph `<br>`
……., …… `<br>`
假定不会出现重名现象。
编写 Hadoop 平台上的 MapReduce 程序，找出所有具有 grandchild-grandparent 关系的人名对。

题目 4

输入文件为学生成绩信息，包含了必修课与选修课成绩，格式如下：
班级1, 姓名1, 科目1, 必修, 成绩1 `<br>`（注： `<br>` 为换行符）
班级2, 姓名2, 科目1, 必修, 成绩2 `<br>`
班级1, 姓名1, 科目2, 选修, 成绩3 `<br>`
……, ……, ……, …… `<br>`
编写Spark程序，实现如下功能： 按班级统计必修课平均成绩在：90~100、80~89、70~79、60~69 和 60 分以下这 5 个分数段的学生人数。

准备工作

安装 Microsoft MPI

首先需要安装 Microsoft MPI。

参考了 windows安装MPI及python安装mpi4py_下载mpi4py代码在vmware-CSDN博客。

直接照着下载安装配置环境变量即可。

安装 Docker

安装 WSL 2

Docker Desktop for Windows 需要 WSL 2 支持，所以需要先安装 WSL 2。

参考了 # Windows10安装WSL的Ubuntu20.04系统及踩坑记录

安装 Docker

参考了 在国内 Windows 平台上安装 Docker 的详细教程_docker windows intel-CSDN博客

部署 Hadoop 和 Spark 实验环境

`git clone` 一下 hadoop-sandbox/hadoop-sandbox: A fully-functional Hadoop Yarn cluster as docker-compose deployment.

然后切换到解压后的目录下，使用 `powershell` 执行 `docker-compose up -d`，会根据该目录下 `.yml` 文件来下载镜像并启动容器。

接着执行 `docker-compose down` 来关闭启动的容器。

下载老师提供的 `spark-install.zip` 压缩包，并解压。然后用 `powershell` 切换到解压后的目录，并执行 `docker build -t packet23/hadoop-client:latest .` ，此命令将根据 `Dockerfile` 的指示在名为 `hadoop-client` 的 Docker 镜像中下载并安装 Python3.8，Spark-3.2.1 和 apache-maven-3.8.5。

上述操作结束后就成功生成了实验所需的全部 Docker 镜像。

为 clientnode 安装 mrjob 库

由于我使用的语言为 `Python`，所以需要为容器安装 `mrjob` 库。

由于容器内没有 `pip`，所以先要对其安装 `pip`。

先 `docker ps` 查看 `clientnode` 容器 ID，然后 `docker exec -it --user root clientnode bash` （其中 `clientnode` 换成容器 ID），进入后再执行

```
apt update
apt install -y python3-pip
pip3 install mrjob
```

如此，便安装好了 `mrjob` 库。

为 nodemanager 安装 python3

再测试老师给的代码时，发现 `nodemanager` 没有安装 `python3` ，无法运行。

所以需要对其安装 `python3`。

先 `docker ps` 查看 `nodemaanager` 容器 ID，然后 `docker exec -it --user root nodemanager bash` （其中 `nodemanager` 换成容器 ID），进入后再执行

```
apt-get update
apt-get install -y python3 python3-pip
```

如此，便成功为 `nodemanager` 安装了 `python3` 。

实验思路

题目 1

梯形法计算定积分近似值。

考虑将定积分区间分成 n 个子区间，然后再均分给进程（开的进程数是 n 的因数，假设为 size ），每个进程计算 n // size 个 子区间用梯形法计算出的值的和，然后再用 Reduce 函数来把各个进程的结果用 MPI.SUM 累加起来即可。

题目 2

对于第一个要求，计算每个学生必修课的平均成绩，在 map 阶段只需将每一行信息转换成键值对 `{(班级，姓名)，(必修课分数，1)}`，键加入班级是为了避免重名问题。然后在 reduce 阶段对同一个人的成绩进行聚合，计算每个人的必修课平均成绩即可。

在做第一个要求的时候遇到了一个问题，花了很长时间才解决，那就是输出的文件的中文总是会变成 `unicode` 编码。问了很久大模型，提供的方法都无法解决。最后无果去 google 了一下，在 `mrjob` 的 `github` 仓库找到了一个 issue，也是跟我一样的问题，参考了一下成功解决了。看来还是不能太依赖大模型（当然也有可能是我提问的姿势不太对）。

对于第二个要求，在 map 阶段把每一行信息转换为键值对 `{班级, (姓名, 分数, 1)}`。在 reduce 阶段对同一个班级的学生成绩，开一个字典统计其总分与科目数，用于计算平均成绩，接着根据平均成绩进行从高到低排序，取前五即可。

题目 3

根据给出的亲子关系可以画出一个图，要找 grandchild-grandparent 关系的话，只需找一个中介点，其所有子代与父代相互之间构成 grandchild-grandparent 关系。

所以，在 map 阶段，将每行的信息 `A,B`，输出两个键值对 `{A, ("child", B)}, {B, ("parent", A)}`，分别代表 A 的孩子是 B，B 的父母是 A。

在 reduce 阶段，对同一个人的信息进行聚合，开两个列表分别存储这个人的孩子与父母，接着把这两个列表相互之间输出 grandchild-grandparent 关系即可。

题目 4

显然，这个题目需要用到题目 2 的要求 1 生成的结果。

这个结果的格式为 `班级 姓名 必修课平均成绩`。

所以读入之后，先对其进行两次 map，第一次将信息按照空格分割成一个列表，第二次将列表的第一个字段和第三个字段（班级和分数，因为不需要关注姓名）形成键值对，形成了一个键值对 rdd。

得到这个键值对 rdd 后，可以写一个函数来把分数映射到对应的区间，然后生成形如 `{班级, (分数区间, 1)}` 的键值对，这个只需写好这个分数映射函数之后使用 `mapValues` 算子即可。

然后，是对同一个班级的同一个分数区间进行计数统计，所以可以把原键值对 `{班级, (分数区间, 1)}` 变成一个新的键值对 `{(班级, 分数区间), 1)}`，这样方便统计结果。这个也是用 `map` 算子即可。

然后统计结果就很简单了，直接用 `reduceByKey` 即可。

为了使结果更清晰，我使用 `sortByKey` 函数把结果按照区间从小到大的顺序来输出了。

代码

题目 1

```python
from mpi4py import MPI
from math import sin

def f(x):
    return sin(x * x + x)

comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

if rank == 0:
    a = float(input("请输入定积分下限："))
    b = float(input("请输入定积分上限："))
else :
    a = None
    b = None

a = comm.bcast(a, root=0)
b = comm.bcast(b, root=0)

n = 10000000
chunk_size = n // size
length = (b - a) / n

partial_res = 0
for i in range(0, chunk_size):
    l = a + (rank * chunk_size + i) * length
    partial_res += length * (f(l) + f(l + length)) * 0.5

total_res = comm.reduce(partial_res, MPI.SUM, root=0)

if rank == 0:
    print(f"近似值为：{total_res}")
```

题目 2

1.

```python
# aveScore.py
from mrjob.job import MRJob
from mrjob.protocol import RawProtocol

class MRAveScore(MRJob):
    OUTPUT_PROTOCOL = RawProtocol
    # Mapper函数：把每一行信息分解出来，然后为每一行信息生成一个键值对{(班级，姓名)，(必修课分数，1)}
    def mapper(self, _, line):
        cls, name, subject, subject_type, score = line.strip().split(',')
```

```
        score = float(score)
        if subject_type == "必修":
            yield (cls, name), (score, 1)

    # Reducer函数：对同一个人的成绩进行聚合，计算每个人的必修课平均成绩。
    def reducer(self, key, values):
        total_score = 0
        total_num = 0
        for score, num in values:
            total_score += score
            total_num += num
        average_score = total_score / total_num if total_num else 0
        yield f"{key[0]} {key[1]}", f"{average_score:.2f}"

if __name__ == '__main__':
    MRAveScore().run()
```

2.

```
# top5.py
from mrjob.job import MRJob
from mrjob.protocol import RawProtocol

class MRTop5(MRJob):
    OUTPUT_PROTOCOL = RawProtocol

    # Mapper函数：把每一行的信息分解出来，生成键值对 {班级, (姓名, 分数, 1)}
    def mapper(self, _, line):
        cls, name, subject, subject_type, score = line.strip().split(',')
        score = float(score)
        yield cls, (name, score, 1)

    # Reducer函数：对同一个班级的学生信息进行聚合，开一个 scores 字典用来统计该班级内
的学生总分与科目数，用于计算平均成绩
    def reducer(self, key, values):
        scores = {}

        for name, score, num in values:
            if name not in scores:
                scores[name] = [0.0, 0]
            scores[name][0] += score
            scores[name][1] += num

        avg_scores = [(name, score / num) for name, (score, num) in
scores.items()]

        top5 = sorted(avg_scores, key=lambda x: x[1], reverse=True)[:5]
```

```
        for name, score in top5:
            yield key, f"{name}\t{score:.2f}"

if __name__ == '__main__':
    MRTop5.run()
```

题目 3

```
from mrjob.job import MRJob

class MRFindGrandparents(MRJob):
    # Mapper函数：输出两个键值对 `{A, ("child", B)}, {B, ("parent", A)}`，分别代表 A 的孩子是 B，B 的父母是 A。
    def mapper(self, _, line):
        parent, child = line.strip().split(',')
        yield parent, ("child", child)
        yield child, ("parent", parent)

    # Reducer函数：对同一个人的信息进行聚合，生成 grandchild-grandparent 关系
    def reducer(self, key, values):
        parents = []
        children = []
        for relation, name in values:
            if relation == "child":
                children.append(name)
            else:
                parents.append(name)
        for child in children:
            for parent in parents:
                yield child, parent

if __name__ == '__main__':
    MRFindGrandparents.run()
```

题目 4

```
from pyspark import SparkContext

# 分数映射到区间函数
def score_to_range(score):
    if score < 60:
        return "0~59"
    elif score < 70:
        return "60~69"
    elif score < 80:
        return "70~79"
    elif score < 90:
```

```
            return "80~89"
        else:
            return "90~100"

    def score_statistics(input_file):
        sc = SparkContext("local", "Score Statistics")

        data_rdd = sc.textFile(input_file)

        class_score = data_rdd.map(lambda line: line.split()).map(lambda fields:
    (fields[0], float(fields[2])))

        # 分数映射到区间
        class_score_ranges = class_score.mapValues(lambda score:
    (score_to_range(score), 1))

        # 将键变为 {班级, 分数区间}，方便统计数量
        class_score_counts = class_score_ranges.map(lambda x: ((x[0], x[1][0]),
    x[1][1]))

        class_score_result = class_score_counts.reduceByKey(lambda x, y: x + y)

        result = class_score_result.sortByKey(ascending=True,
    numPartitions=1).map(lambda x: (x[0][0], x[0][1], x[1]))

        result.saveAsTextFile("result")

        sc.stop()

    if __name__ == "__main__":
        input_file = "input.txt"
        score_statistics(input_file)
```

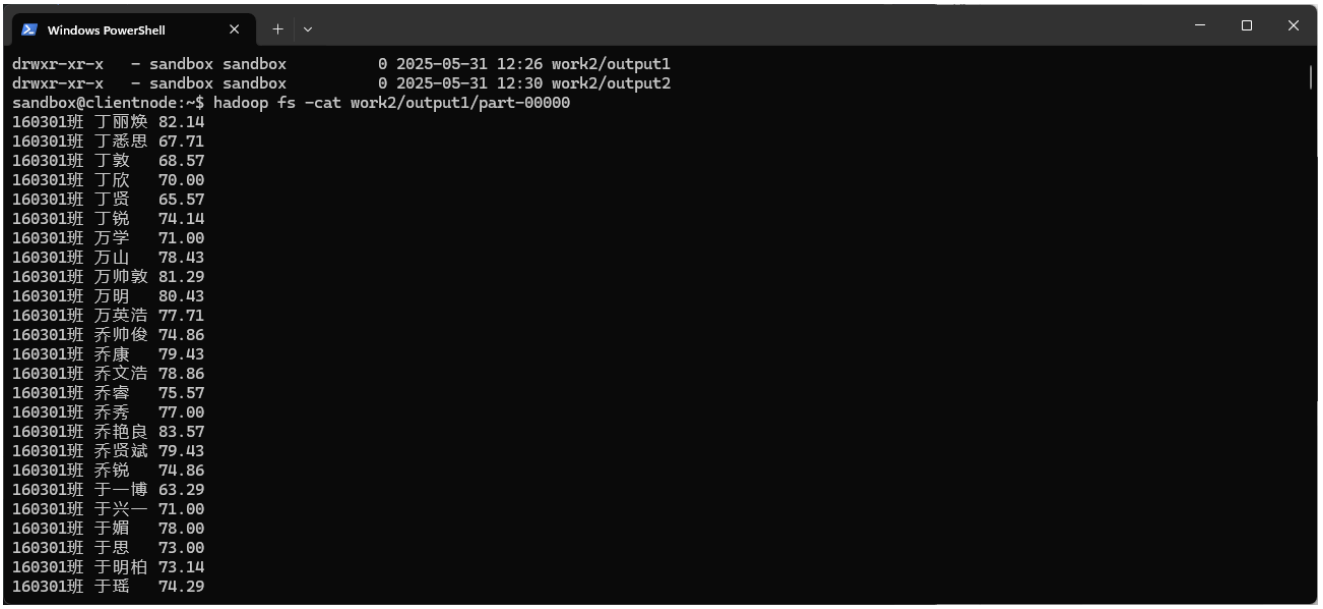
运行示例

题目 1

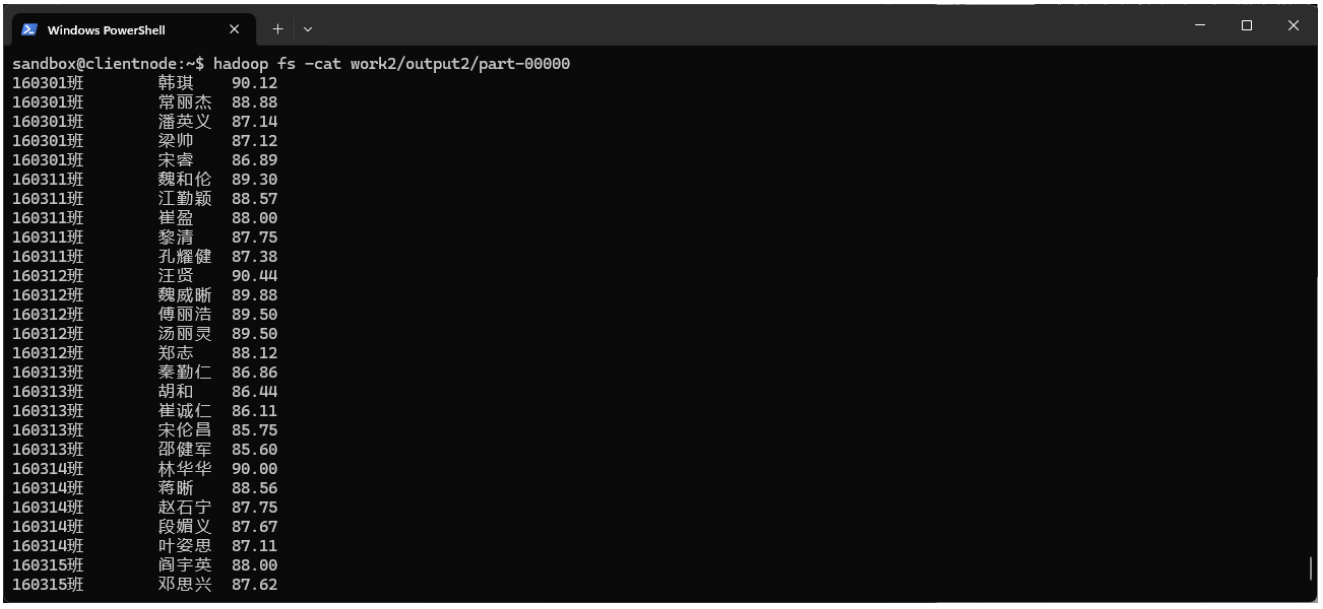
```
请输入定积分下限：1
请输入定积分上限：4
近似值为：-0.12718775757891396
PS D:\Desktop\classwork\sixth\Distributed-Computing\Distributed-Computing-lab\lab4\work1> mpiexec -n 8 python work1.py
请输入定积分下限：1
请输入定积分上限：10
近似值为：-0.03186496671580331
PS D:\Desktop\classwork\sixth\Distributed-Computing\Distributed-Computing-lab\lab4\work1> mpiexec -n 8 python work1.py
请输入定积分下限：1
请输入定积分上限：35
近似值为：-0.06570189940993855
PS D:\Desktop\classwork\sixth\Distributed-Computing\Distributed-Computing-lab\lab4\work1> mpiexec -n 8 python work1.py
请输入定积分下限：10
请输入定积分上限：20
近似值为：-0.06127398186111078
```

题目 2

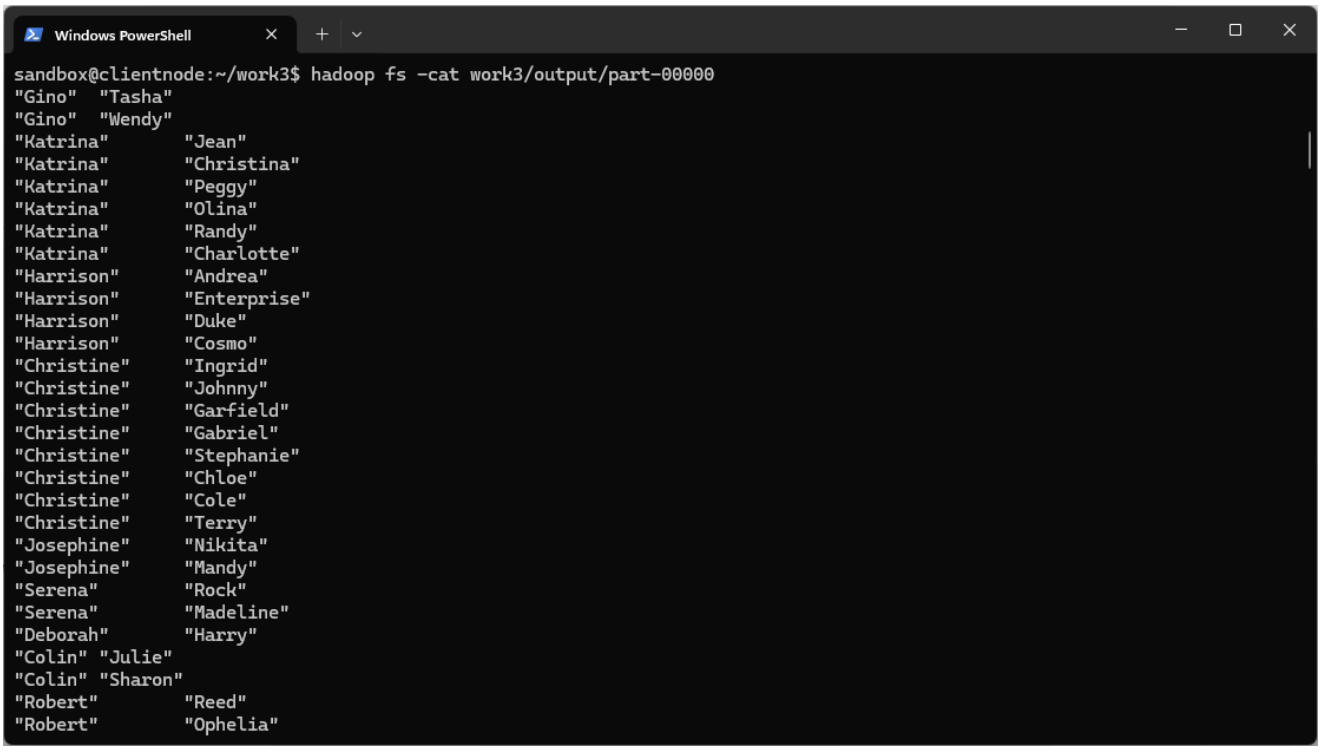
1.


Windows PowerShell
drwxr-xr-x - sandbox sandbox 0 2025-05-31 12:26 work2/output1
drwxr-xr-x - sandbox sandbox 0 2025-05-31 12:30 work2/output2
sandbox@clientnode:~$ hadoop fs -cat work2/output1/part-00000
160301班 丁丽焕 82.14
160301班 丁悉思 67.71
160301班 丁敦 68.57
160301班 丁欣 70.00
160301班 丁贤 65.57
160301班 丁锐 74.14
160301班 万学 71.00
160301班 万山 78.43
160301班 万帅敦 81.29
160301班 万明 80.43
160301班 万英浩 77.71
160301班 乔帅俊 74.86
160301班 乔康 79.43
160301班 乔文浩 78.86
160301班 乔睿 75.57
160301班 乔秀 77.00
160301班 乔艳良 83.57
160301班 乔贤斌 79.43
160301班 乔锐 74.86
160301班 于一博 63.29
160301班 于兴一 71.00
160301班 于媚 78.00
160301班 于思 73.00
160301班 于明柏 73.14
160301班 于瑶 74.29

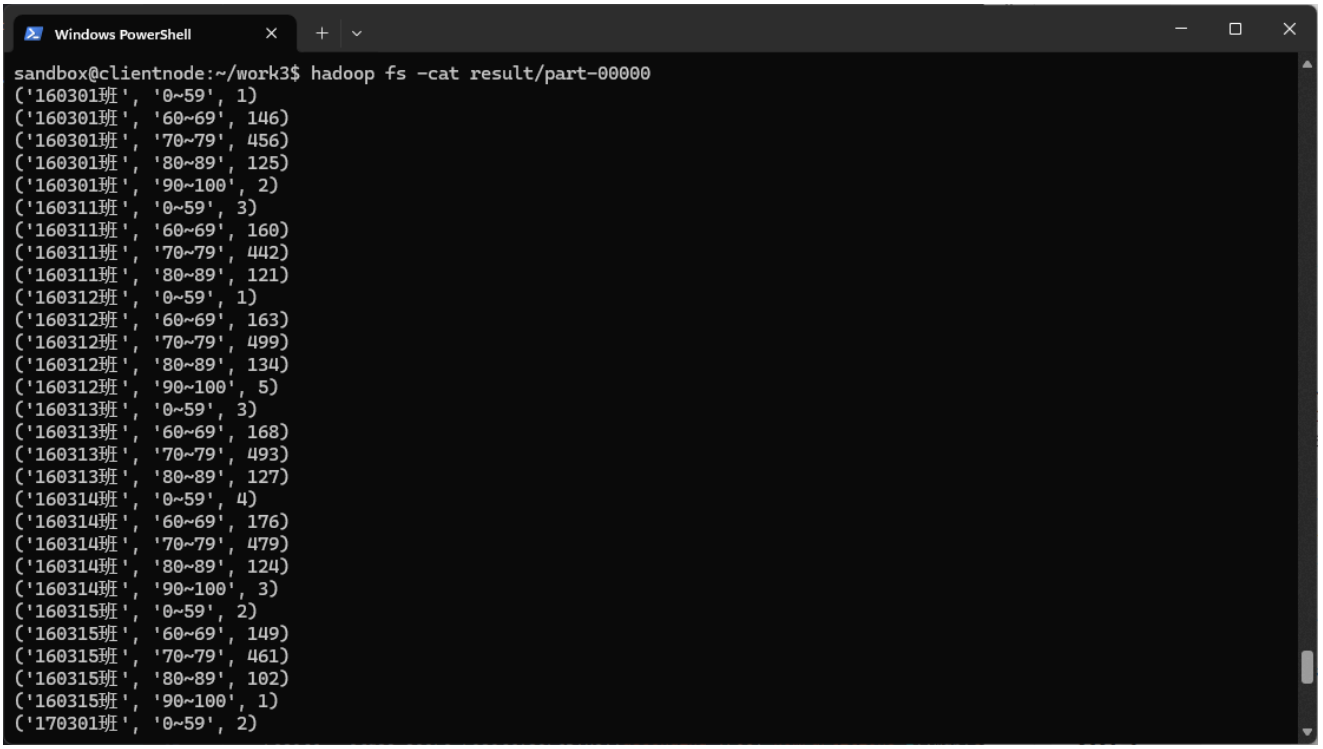
2.


Windows PowerShell
sandbox@clientnode:~$ hadoop fs -cat work2/output2/part-00000
160301班 韩琪 90.12
160301班 常丽杰 88.88
160301班 潘英义 87.14
160301班 梁帅 87.12
160301班 宋睿 86.89
160311班 魏和伦 89.30
160311班 江勤颖 88.57
160311班 崔盈 88.00
160311班 黎清 87.75
160311班 孔耀健 87.38
160312班 汪贤 90.44
160312班 魏威晰 89.88
160312班 傅丽浩 89.50
160312班 汤丽灵 89.50
160312班 郑志 88.12
160313班 秦勤仁 86.86
160313班 胡和 86.44
160313班 崔诚仁 86.11
160313班 宋伦昌 85.75
160313班 邵健军 85.60
160314班 林华华 90.00
160314班 蒋晰 88.56
160314班 赵石宁 87.75
160314班 段媚义 87.67
160314班 叶姿思 87.11
160315班 阎宇英 88.00
160315班 邓思兴 87.62

题目 3


Windows PowerShell
sandbox@clientnode:~/work3$ hadoop fs -cat work3/output/part-00000
"Gino" "Tasha"
"Gino" "Wendy"
"Katrina"   "Jean"
"Katrina"   "Christina"
"Katrina"   "Peggy"
"Katrina"   "Olina"
"Katrina"   "Randy"
"Katrina"   "Charlotte"
"Harrison"   "Andrea"
"Harrison"   "Enterprise"
"Harrison"   "Duke"
"Harrison"   "Cosmo"
"Christine"   "Ingrid"
"Christine"   "Johnny"
"Christine"   "Garfield"
"Christine"   "Gabriel"
"Christine"   "Stephanie"
"Christine"   "Chloe"
"Christine"   "Cole"
"Christine"   "Terry"
"Josephine"   "Nikita"
"Josephine"   "Mandy"
"Serena"   "Rock"
"Serena"   "Madeline"
"Deborah"   "Harry"
"Colin" "Julie"
"Colin" "Sharon"
"Robert"   "Reed"
"Robert"   "Ophelia"

题目 4


Windows PowerShell
sandbox@clientnode:~/work3$ hadoop fs -cat result/part-00000
('160301班', '0~59', 1)
('160301班', '60~69', 146)
('160301班', '70~79', 456)
('160301班', '80~89', 125)
('160301班', '90~100', 2)
('160311班', '0~59', 3)
('160311班', '60~69', 160)
('160311班', '70~79', 442)
('160311班', '80~89', 121)
('160312班', '0~59', 1)
('160312班', '60~69', 163)
('160312班', '70~79', 499)
('160312班', '80~89', 134)
('160312班', '90~100', 5)
('160313班', '0~59', 3)
('160313班', '60~69', 168)
('160313班', '70~79', 493)
('160313班', '80~89', 127)
('160314班', '0~59', 4)
('160314班', '60~69', 176)
('160314班', '70~79', 479)
('160314班', '80~89', 124)
('160314班', '90~100', 3)
('160315班', '0~59', 2)
('160315班', '60~69', 149)
('160315班', '70~79', 461)
('160315班', '80~89', 102)
('160315班', '90~100', 1)
('170301班', '0~59', 2)