任务3 - 使用MapReduce编程模型实现WordCount算法

任务描述

1. 使用Hadoop或者其他框架实现WordCount算法
2. 测试程序在 1KB、1MB、10MB、100MB 等不同数据量情况下的加速比
3. 不可以直接调用Hadoop的Demo库完成WordCount算法, 需自行实现 mapper 和 reducer

Hadoop环境配置

- Hadoop 版本：3.3.6
- Python：使用miniconda的 Python3
- 依赖：matplotlib, HDFS 命令行工具可用, Hadoop 集群处于可运行状态
- 在集群或本机安装 Hadoop 并启动 NameNode、DataNode、YARN
- 确保 hadoop-streaming jar 路径正确

WordCount实现

数据准备 - data.py

- 作用：生成不同大小的测试文件(test_1k.txt、test_1m.txt、test_10m.txt、test_100m.txt)
- 实现要点：
  - 随机生成单词, 长度 3–10 字符, 字母 a–z
  - 按目标字节数写入文件, 随机控制每行单词数(示例 5–15), 写入直到达到目标大小
  - 在脚本末尾逐个生成 1KB/1MB/10MB/100MB 文件并打印实际生成大小

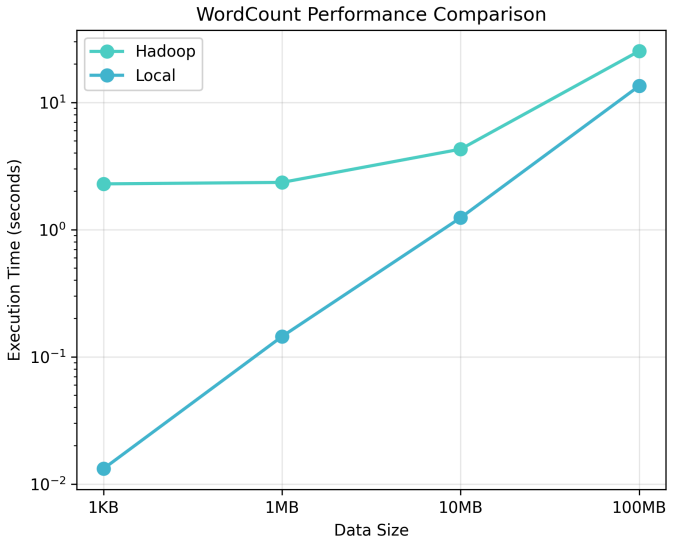
算法实现 - WordCountMapper.py / WordCountReducer.py

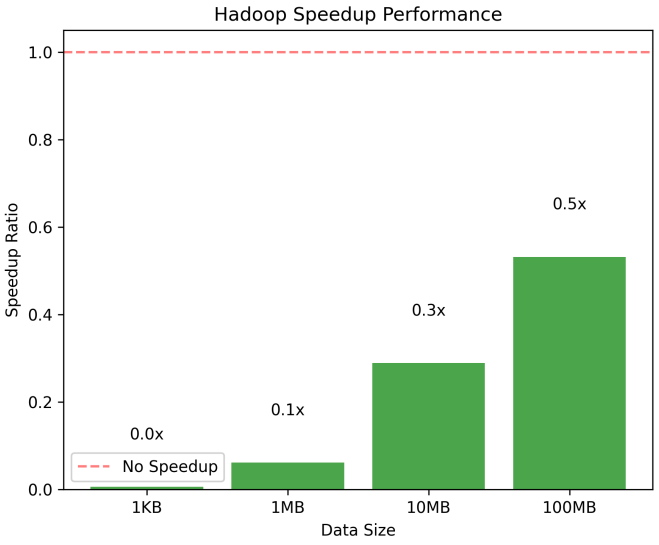
- Mapper(WordCountMapper.py)：
  - 从标准输入逐行读取, 使用正则 r"\b\w+\b" 提取单词并转为小写
  - 对每个单词输出 "word\t1" 作为中间键值对
- Reducer(WordCountReducer.py)：
  - 从标准输入读取已排序的 key\tvalue 行
  - 对连续相同单词累加计数, 遇到新单词则输出上一个单词的总计

性能测试 - main.py

- 测试代码流程：
  1. 运行 data.py 生成测试文件
  2. 将测试文件上传到 HDFS (示例路径 /user/hadoop/input/)
  3. 对每个测试文件运行 Hadoop Streaming 任务并记录耗时
  4. 在本地使用 pipeline(mapper | sort | reducer) 运行同样的流程并记录耗时, 用于对比
  5. 汇总每个数据规模的 Hadoop 时间、本地时间与计算加速比(local_time / hadoop_time), 并绘制图表

性能测试结果与分析





数据量	Hadoop时间(s)	本地时间(s)	加速比
1KB	2.29	0.01	0.01x
1MB	2.35	0.14	0.06x
10MB	4.29	1.24	0.29x
100MB	25.30	13.46	0.53x

- 结果解读：
  - 小数据量(1KB、1MB)时, 本地单进程处理明显更快, 原因是 Hadoop 的启动与调度开销高
  - 随着数据量增长, Hadoop 的分布式处理优势逐渐显现(10MB、100MB), 加速比提升