并行分布式计算实验报告

项目二：最短路径

数据科学与计算机学院　17大数据与人工智能

17341015　陈鸿峥

一、 题目描述

现实世界的很多场景中，需要计算最短路径，如导航中的路径规划等，但是如何在大规模路径中快速找出最短路径，是一个具有挑战性的问题。本项目要求利用MPI + OpenMP，从一个至少包含1万个节点，10万条边的图中，寻找最短路径，边上的权重可随机产生。统一测试程序的执行时间，进行排名，根据排名计算成绩。

输入数据格式：

- 20001个点，7000w+条边，求点0到点20000的最短路径（有解）
- 每行3个数，分别为源节点、汇节点、权重（非负）

输出：整个路径及路径长度

测试环境：3个节点（同主机的虚机），每个节点两个核心（双核四线程），4G内存，机械硬盘

参照：

- https://github.com/Lehmannhen/MPI-Dijkstra
- https://github.com/laplaceyc/Parallel-Programing

二、 解决方案

Dijkstra算法是串行求解单源最短路径的常见算法，伪代码如下

Algorithm 1 Sequential Dijkstra SSSP

```
1:  procedure DIJKSTRA(Graph,Source)
2:      Create vertex set Q                                  ▷ Initialization
3:      for each vertex in Graph do
4:          dist[v] ← INFINITY
5:          prev[v] ← UNDEFINED
6:          add v to Q
7:      dist[source] ← 0
8:      while Q is not empty do                              ▷ Dijkstra
9:          u ← vertex in Q with min dist[u]
10:         remove u from Q
11:         for each neighbor v of u do
12:             alt ← dist[u] + length(u,v)
13:             if alt < dist[v] then                        ▷ Relaxation
14:                 dist[v] ← alt
15:                 prev[v] ← u
16:     return dist[],prev[]
```

实际上，对于两个`for each`部分，就可以使用并行方法进行加速，因为在循环间都没有依赖关系。还有一个可并行的地方则是在求解距离最小值的部分（伪代码第9行）。

由于我现在的研究方向就是图计算，故本次作业直接复用了我今年以第一作者投稿于SC'19的部分代码，该代码是开源的[1]。

简单来说，优化技术有以下几点。

- 将输入文件以字符串形式完全读入内存后，才并行进行处理，包括并行的初始化、并行的**基数排序**、并行的赋值等。
- 用**CSR(Compressed Sparse Row)格式**存储，见图1

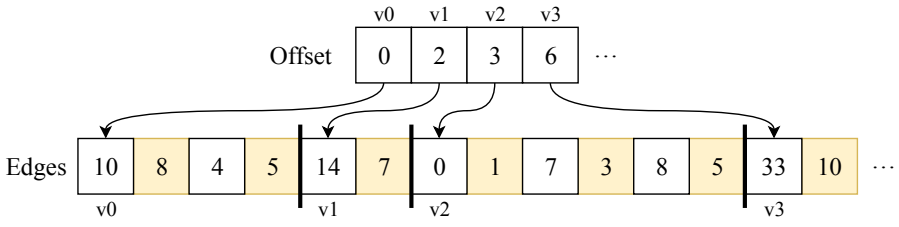

图 1: CSR格式

其中，第一个数组存储的是边表的偏移量`offset`，第二个数组存储的是每个源结点的邻居（第一项）及边权（第二项）。以CSR格式存储，可以使程序具有良好的空间局部性。

[1]Krill: An Efficient Concurrent Graph Processing System, https://github.com/chhzh123/Krill

当确定了源结点（当前迭代最小距离点）后，内层循环都是遍历源结点的邻居。而在CSR格式中，邻居都是紧密存储的（包括边权），故这可以确保所需的数据都在Cache中，进而大大缩短访存时间。

头文件都在`include`文件夹中，并行操作都已封装在`parallel.h`头文件中（如OpenMP的`parallel_for`等），主函数位于`sssp.cpp`文件。

三、 实验结果

运行可直接键入`make run`，结果如下（调用了Linux的`time`函数进行计时）

```
$time make run
./sssp 10001x10001GraphExamples.txt 20001
Dist: 2
20000<-826<-0

real    0m4.243s
user    1m26.519s
sys     0m19.259s
```