lab2

韩佳乐 PB16051152

实验题目

1. 题目：
   利用 MPI 进行蒙特卡洛模拟
2. 内容：
   在道路交通规划上，需要对单条道路的拥堵情况进行估计。因为仅考虑单条车道，所以不存在超车。假设共有 n 辆车，分别编号 0, 1, …, n-1，每辆车占据一个单位的空间。初始状态如下，n 辆车首尾相连，速度都是 0。每个时间周期里每个车辆的运动满足以下规则：
   - 假设当前周期开始时，速度是 v。
   - 和前一辆车的距离为 d（前一辆车车尾到这辆车车头的距离，对于第 0 号车，d=无穷大），若 d>v，它的速度会提高到 v + 1 。最高限速 v_max。若d <= v，那么它的速度会降低到 d。
   - 前两条完成后，司机还会以概率 p 随机减速 1 个单位。速度不会为负值。
   - 基于以上几点，车辆向前移动 v（这里的 v 已经被更新）个单位
3. 实验要求：
   - v_max，p 的值请自行选取，要求 v_max 不低于 10，p 不为 0 即可
   - 实验规模：
     - 车辆数量为 100 000，模拟 2000 个周期后的道路情况。
     - 车辆数量为 500 000 模拟 500 个周期后的道路情况。
     - 车辆数量为 1 000 000 ，模拟 300 个周期后的道路情况。

实验环境

操作系统	编译器	硬件配置
Ubuntu 16.04	mpicc	双核 4G内存

算法设计与分析

初始化条件

n辆车首尾相连，速度都是0。第0号车，d=无穷大。第 n-1 号车的位置为0，每辆车占据一个单位的空间。最大速度10，最小速度0。随机减速概率为0.5。

车辆结构体

v是速度，pos是位置，d是与前车的距离。

```
typedef struct
{
    int v;
    int pos;
    int d;
}car;
```

核心循环

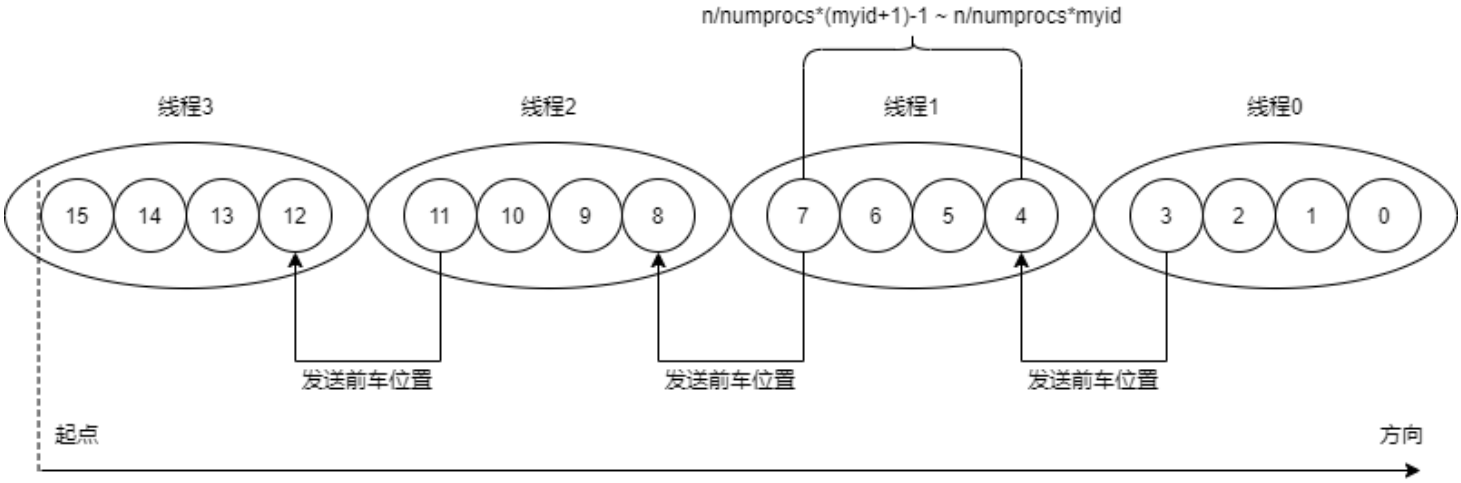
根据题意可以推出以下逻辑。需要对这段程序进行周期次数的循环。

1. 当 d>v 且 v 不超过最大速度时，v 增加到 v+1；
2. 当 d<=v 时，v 变成 d；
3. 使用 srand() 和 rand() 产生随机数进行减速；
4. 车辆向前移动 v 个单位；
5. 每辆车与前车的距离等于牵扯与自己的位置之差。

```
if(list[k].d > list[k].v && list[k].v < V_MAX) list[k].v++;
if(list[k].d <= list[k].v) list[k].v = list[k].d;
srand(k * j + clock());
if(list[k].v >= 1){
    if(rand() % 10 < P) list[k].v--;
}
list[k].pos += list[k].v;
list[k].d = list[k - 1].pos - list[k].pos;
```

并行设计

下图是16辆车，4线程的概念模型。n是车辆总数，numprocs是线程数，myid是线程号。



1. 初始化MPI环境，获取并行环境参数(总线程数、本地进程编号等)。

```
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
MPI_Comm_rank(MPI_COMM_WORLD, &myid);
```

2. 从0号进程获取输入车辆数和周期，并将其广播出去。

```
if(myid == 0){
    printf("车辆数量:");
    scanf("%d", &n);
    printf("周期:");
    scanf("%d", &t);
    start = MPI_Wtime();
}
MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);    //将n值广播出去
MPI_Bcast(&t, 1, MPI_INT, 0, MPI_COMM_WORLD);    //将t值广播出去
```

3. 初始化车辆列表。

```
    list = (car*)malloc(n * sizeof(car));
    for(int i = 0; i < n; i++){
        list[i].v = 0;
        list[i].d = 0;
        list[i].pos = n - 1 - i;
    }
    list[0].d = D_MAX;
```

4. 根据线程数numprocs的值，将车辆分为numprocs个区间，每个区间负责 n/numprocs 辆车的计算。
5. 除 numprocs-1 号线程外每个线程需要把最后一辆车 n/numprocs*(myid+1)-1 的位置信息发送给后一个线程的第一辆车 n/numprocs*myid，因为后车需要前车的位置来更新自己与前车的距离。因此，每个线程的第一辆车的 d 值的更新方式有些不同。其次，第0号车的 d 值不需要更新，一直是无穷大。

```
int k = n / numprocs * (myid + 1) - 1;
if(myid != numprocs - 1){
    // 向后一个线程发送最后车辆的位置信息
    MPI_Send(&list[k].pos, 1, MPI_INT, myid + 1, myid, MPI_COMM_WORLD);
}
```

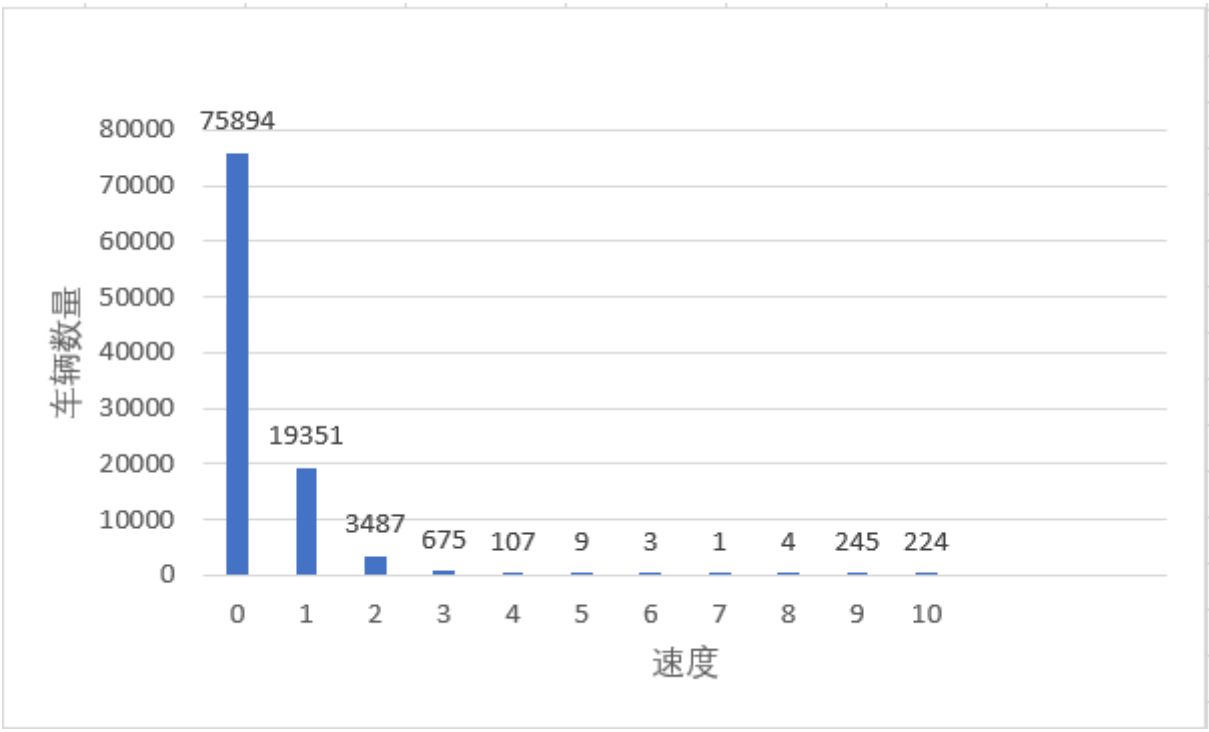
```
for(; k > n / numprocs * myid; k--){
    ...//核心循环
}
if(myid != 0){
    int front_car_pos;
    MPI_Status stat;
    // 接收前一个线程发送的位置信息
    MPI_Recv(&front_car_pos, 1, MPI_INT, myid - 1, myid - 1, MPI_COMM_WORLD, &stat);
    // 更新与前车的距离
    list[k].d = front_car_pos - list[k].pos;
}
```

6. 在统计结果之前等待所有进程完成。

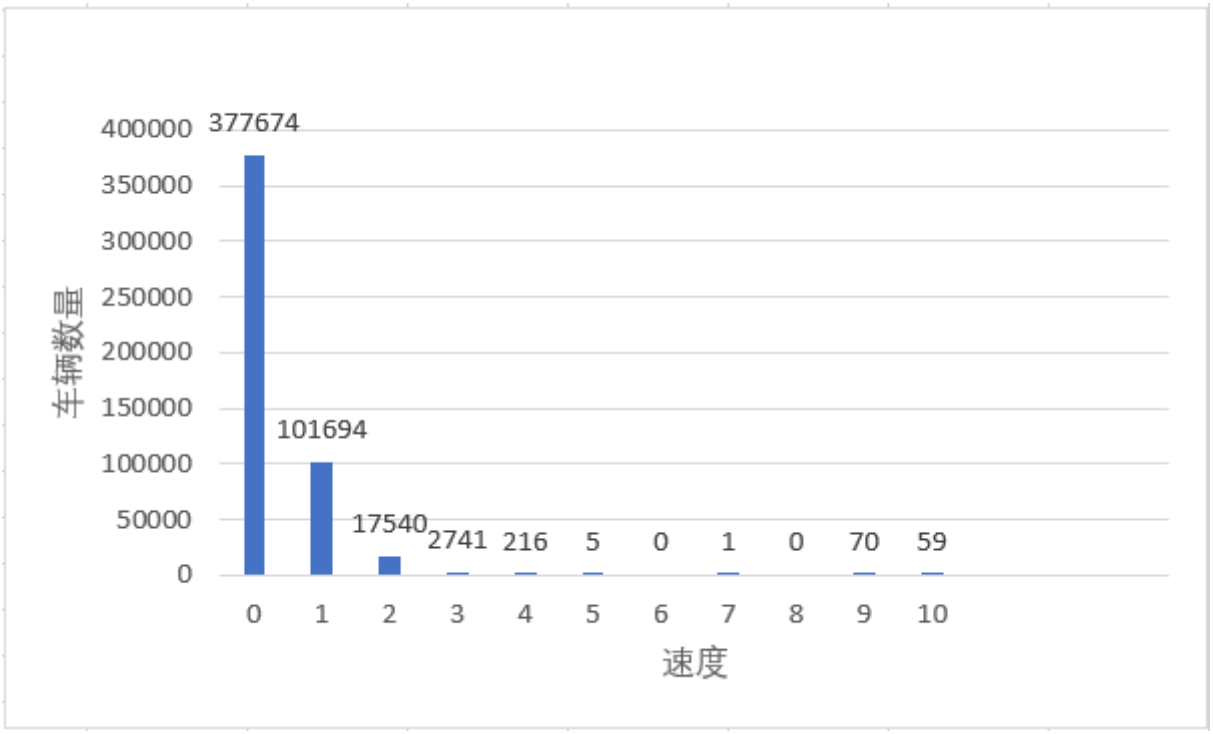
```
MPI_Barrier(MPI_COMM_WORLD);
```

结果统计

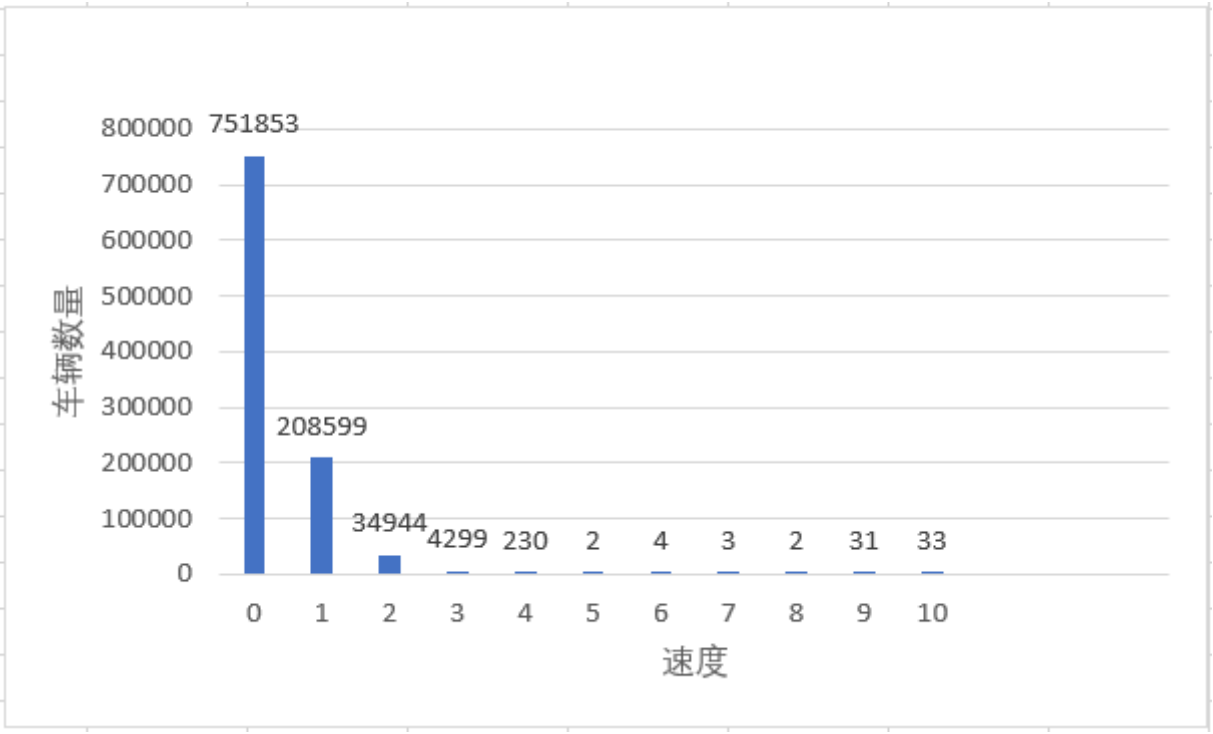
规模：100000辆车 2000周期



规模：500000辆车 500周期



规模：1000000辆车 300周期



运行时间(s)

规模/线程数	1	2	4	8
100000辆车 2000周期	668.715471	345.102948	343.784128	356.262993
500000辆车 500周期	804.459397	431.121841	442.075001	433.319804

1000000辆车 300周期	938.806691	552.925860	537.474469	573.570365

加速比

规模/线程数	1	2	4	8
100000辆车 2000周期	1	1.938	1.945	1.877
500000辆车 500周期	1	1.866	1.820	1.857
1000000辆车 300周期	1	1.698	1.747	1.637