实验 2

PB17111626 秦亚璇

1 实验题目:

利用 MPI 进行蒙特卡洛模拟

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：

（1）假设当前周期开始时，速度是 v。

（2）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。

（3）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。

（4）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。

2 实验环境(操作系统,编译器,硬件配置等):

操作系统 Ubuntu 16.04,编译器 g++，硬件配置 ThinkPad X1 Carbon(16G)

3 算法设计与分析(写出解题思路和实现步骤)

要完成的实验是对 n 辆车的前进情况做一个周期模拟，他们每一个都被前面的车所限制，遵照实验一的思路我们很明显能够想到将 n 辆车的进程平均分配到 m 个进程上面进行同时的计算，这样的实现部分其实就是简单的对要求进行了一次代码实现的编写，所以处理的关键在于要处理相连的部分，要将位于前面的车（后面的进程上）的 pos 作为信息传进来，进行新一轮的 distance 更新，至此其实整个设计就结束了，在这个过程中认识到了一个很好用的逐轮向前传递的 API 接口 Bsend，正常使用 Recv 接受信息即可。这里需要注意的一个 point 是由于我们进行了多周期的模拟，所以我们必须避免信息更新时间不一致的问题，那么我的解决方案是在进入下一轮的时间周期之前，对所有的 process 进行一次对齐。

4 核心代码(写出算法实现的关键部分,如核心的循环等)

```
start = clock();
for(j=0;j<2000;j++)
{
    if(myid != 0)
        MPI_Bsend(&(car_list[num_car/numofprocess*myid].pos),1,MPI_INT,myid-1,0,MPI_COMM_WORLD);
    for(i=(num_car/numofprocess*myid);i<num_car/numofprocess*(myid+1)-1;i++)
    {
        car_list[i].d=car_list[i+1].pos-car_list[i].pos;
        if(car_list[i].v < vmax)
            car_list[i].v++;
        if(car_list[i].d<=car_list[i].v)
            car_list[i].v = car_list[i].d;
        srand((unsigned)time(NULL));
        if(car_list[i].v > 1 && rand()%10 < p)
            car_list[i].v--;
        car_list[i].pos += car_list[i].v;
    }
    if(myid!=numofprocess-1)
    {
        int cac;
        MPI_Status status;
        MPI_Recv(&cac,1,MPI_INT,myid+1,0,MPI_COMM_WORLD,&status);
        car_list[i].d = cac - car_list[i].pos;
    }
    if(car_list[i].v < vmax)
        car_list[i].v++;
    if(car_list[i].d<=car_list[i].v)
        car_list[i].v=car_list[i].d;
    srand((unsigned) time(NULL));
    if( car_list[i].v>1 && rand()%10<p )
        car_list[i].v--;
    car_list[i].pos += car_list[i].v;
    MPI_Barrier(MPI_COMM_WORLD);
}
```

5 实验结果

运行时间

规模\进程数	1	2	4	8
100000/2000	726s	253s	141s	184s
500000/500	580s	430s	309s	230s
1000000/300	597s	526s	420s	301s

加速比

规模\进程数	1	2	4	8
100000/2000	1	2.87	5.15	3.95
500000/500	1	1.35	1.88	2.52
1000000/300	1	1.13	1.42	1.98

6 分析与总结

1. 当问题规模和周期数都比较大的时候，显然进程数越多加速效果会越好，而问题复杂度达不到一定程度的时候，较小的进程数也许能通过节约通信开销而非计算开销实现更好的效果，但在总的问题规模（简单使用规模*周期数来进行估计）更大的时候，加速比看起来有一定程度的降低，我对此的思考是，周期对问题计算效率的影响远不如规模来的影响大，故而多次进程对齐实际开销尚且不如问题规模的通信开销来的明显。
2. 对我个人而言，总结一下这次实验中可能出现的问题，确实耗时有些惊人的长，首先是虚拟机本身内存不大，进程池可能也不大，同时我为了节约时间同时开多个 terminal 进行多个实验可能对性能造成了影响，引以为戒