实验1

实验环境

macOS 10.14，mpirun (Open MPI) 4.0.3，gcc version 9.3.0 (Homebrew GCC 9.3.0_1)

由于电脑线程不够（仅有2核4线程），在使用mpi时使用了`oversubscribe`，所以有部分无法运行。

算法设计与分析

素数

利用多个进程对循环进行计数，然后规约。在openmp中可以简单使用reduction，在mpi中则要给各个进程分配local变量计数，然后规约。

π

利用多个进程对迭代进行计算，然后规约到pi的值。在openmp中可以简单使用reduction，在mpi中则要给各个进程分配local变量计数，然后规约。

核心代码

素数

```
#pragma omp parallel for schedule(dynamic)reduction(+:sum)
    for (int i = 2; i < N; i++)
    {
        if(isPrime(i))
        {
        sum++;
        }
    }
```

```
MPI_Bcast(&a,1,MPI_INT,0,MPI_COMM_WORLD);
    for(int i=mypid*2+1; i<=a; i+=size*2)
    {
    local+=isPrime(i);
    }
MPI_Reduce(&local,&sum,1,MPI_INT,MPI_SUM,0,MPI_COMM_WORLD);
```

π

```
#pragma omp parallel for default(shared) private(i, local, tmp) reduction(+ : pi)
    for (i = 0; i < N; i++)
    {
        local = (i + 0.5) * w;
        tmp = 4.0 / (1.0 + local * local);
        pi += tmp;
    }
```

```
MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
    sum = 0.0;
    width = 1.0 / n;
    for (i = Comm_rank; i < n; i += Comm_size) {
        local = width * ((double)i + 0.5);
        sum += 4.0 / (1.0 + local * local);
    }
localpi = width * sum;
MPI_Reduce(&localpi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
```

实验结果

素数（mpi）

规模/进程数	1	2	4	8
1,000	0.000045	0.000211/0.213270142	0.000241/0.190871369	0.000658/0.083586626
10,000	0.000898	0.000927/0.968716289	0.001271/0.678992919	0.000979/0.911133810
100,000	0.020409	0.010258/1.989569117	0.008287/2.547966695	0.006913/2.970635035
1000,000	0.500320	0.274400/1.823323615	0.152580/3.259529427	0.159259/3.208735456

运行时间(sec)/加速比

素数（openmp）

规模/进程数	1	2	4	8
1,000	44	378/0.116402	527/0.090090	707/0.070721
10,000	554	1454/0.381018	2197/0.251139	1866/0.324223
100,000	11847	14324/0.827073	20456/0.478686	21173/0.461484
1000,000	227577	263300/0.864326	379463/0.592016	381219/0.606830

运行时间(ms)/加速比

π（mpi）

规模/进程数	1	2	4	8
1,000	0.000161	0.000385/0.418182	0.000797/0.227723	0.000913/0.176342
10,000	0.000197	0.000565/0.348673	0.000614/0.320847	*N/A*
50,000	0.000693	0.000499/1.38878	0.001025/0.676098	0.001231/0.562957
100,000	0.000942	0.000703/1.33997	0.001146/0.82199	*N/A*

运行时间(sec)/加速比

π（openmp）

规模/进程数	1	2	4	8
1,000	0.000045	0.000119/0.378151	0.000177/0.254237	0.000269/0.167286
10,000	0.000115	0.000157/0.732484	0.000219/0.525114	0.000330/0.348485
50,000	0.000382	0.000536/0.712687	0.000372/1.02688	0.000419/0.911695
100,000	0.000734	0.001059/0.693107	0.000600/1.22333	0.000519/1.41426

运行时间(sec)/加速比

分析与总结

使用mpi求解素数数量得到了较为理想的结果，但是在使用OpenMP时无论如何都无法将加速比提升到1以上。为此我咨询了助教孙经纬，孙助教运行我的代码后一切正常，加速比也达到了较为合理的数值，所以应该是我运行环境的个例问题。在迭代求解π时则较为正常，无论使用何种方法都可以得到对应规模下理想的线程数。而且两个例子都说明，规模越大，对并行越有利。而线程数则会由于进程间通信开销随进程数增加，有一个相对的最佳值。