Assignment02

From Siqi Liang (liangsiq@usc.edu)

Part I

1. Submit the source code of global_pi.c

global_pi.c :

```
#include "mpi.h"
#include <stdio.h>
#define NBIN 10000000
int nprocs;  /* Number of processors */
int myid;    /* My rank */

double global_sum(double partial) {
    /* Implement your own global summation here */
    double mydone, hisdone;
    int bitvalue, partner;
    MPI_Status status;
    mydone = partial;

    for(bitvalue=1;bitvalue<nprocs;bitvalue*=2){
        partner = myid ^ bitvalue;
        MPI_Send(&mydone, 1, MPI_DOUBLE,partner, bitvalue, MPI_COMM_WORLD);
 // bitvalue is treated as communication label here
        MPI_Recv(&hisdone, 1, MPI_DOUBLE,partner, bitvalue, MPI_COMM_WORLD,
&status);
        mydone += hisdone;
    }
    return mydone;
}

int main(int argc, char *argv[]) {
    double partial, sum = 0.0, avg, step, x, pi, cpu1, cpu2;
    int i;

    step = 1.0/NBIN;

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

```
    cpu1 = MPI_Wtime();
    for(i=myid;i<NBIN;i+=nprocs){
        x = (i+0.5)*step;
        sum += 4.0/(1.0 + x*x);
    }

    partial = sum*step;
    pi = global_sum(partial);
    cpu2 = MPI_Wtime();

    if (myid == 0) {
        printf("Number of processors: %d\n", nprocs);
        printf("Value of Pi = %le\n", pi);
        printf("Execution time (s) = %le\n", cpu2 - cpu1);
    }

    MPI_Finalize();

    return 0;
}
```

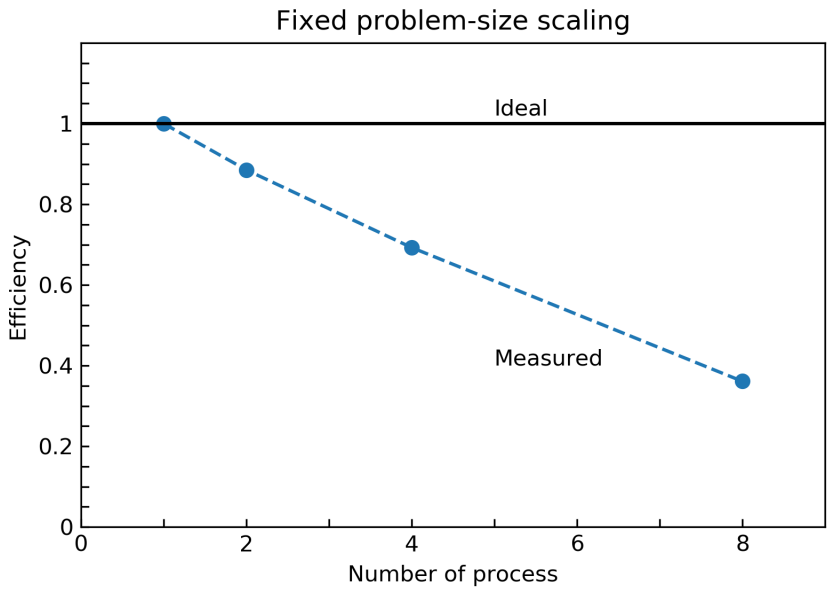
Part II

2. (Fixed problem-size scaling)

In fixed problem-size scaling,

$$E_P = \frac{T(W,1)}{PT(W,P)}$$

The figure below shows that $E_P \leq 1$, where $P = 1, \ldots, 8$.



global_pi.out:

```
Number of processors: 8
Value of Pi = 3.141593e+00
Execution time (s) = 4.895091e-02
Number of processors: 4
Value of Pi = 3.141593e+00
Execution time (s) = 5.103588e-02
Number of processors: 2
Value of Pi = 3.141593e+00
Execution time (s) = 7.986498e-02
Number of processors: 1
Value of Pi = 3.141593e+00
Execution time (s) = 1.413810e-01
```

Data processing:

```
import numpy as np
from matplotlib import pyplot as plt
import matplotlib
time = np.array([14.13810, 7.986498, 5.103588, 4.895091])/100
P = np.array([1,2,4,8])
E_tmp = np.ones(shape=time.shape)*time[0]/P
E_p = E_tmp/time
```

Plot the figure:

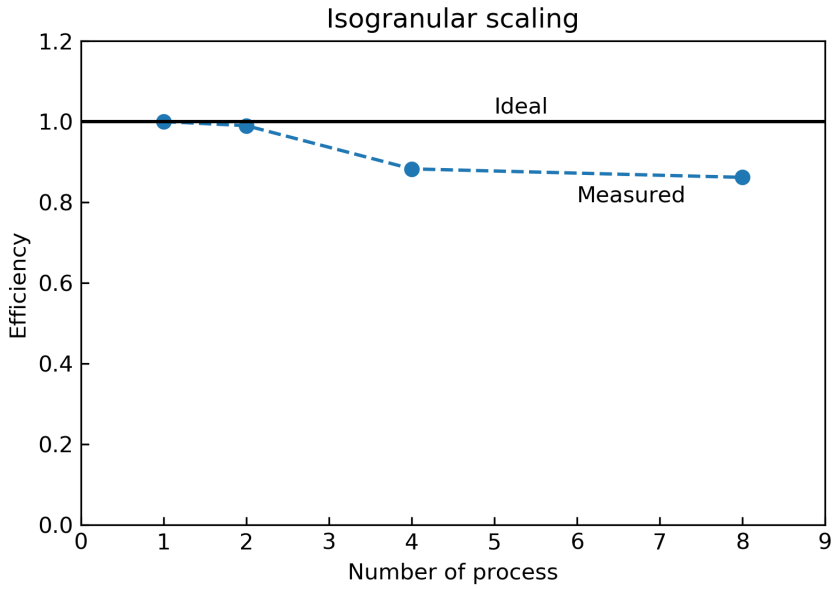
```
matplotlib.rcParams['xtick.direction'] = 'in'
matplotlib.rcParams['ytick.direction'] = 'in'
plt.plot(P,E_p,'o--')
plt.plot(range(10),np.ones(len(range(10))),'k')
plt.xlabel('Number of process')
plt.ylabel('Efficiency')
plt.title('Fixed problem-size scaling')
plt.xlim((0,9))
plt.ylim((0,1.2))
plt.yticks(np.arange(0,1.2,0.05),
('0','','','','0.2','','','','0.4','','','','0.6','','','','0.8','','','','1',
'','','','1.2'))
plt.xticks(range(9),('0','','2','','4','','6','','8'))
plt.text(s='Ideal',x=5,y=1.02)
plt.text(s='Measured',x=5,y=0.4)
plt.savefig('./fixed_problem_size_scaling.png',dpi=300)
```

3. (Isogranular scaling)

In isogranular scaling,

$$E_P = \frac{T(W_1, 1)}{T(W_P, P)}$$

The figure below shows that $E_P \leq 1$, where $P = 1, \ldots, 8$.



global_pi_iso.out:

```
Number of processes: 8
Value of Pi = 3.141593e+00
Execution time (s) = 3.661902e-01
Number of processes: 4
Value of Pi = 3.141593e+00
Execution time (s) = 3.574691e-01
Number of processes: 2
Value of Pi = 3.141593e+00
Execution time (s) = 3.187609e-01
Number of processes: 1
Value of Pi = 3.141593e+00
Execution time (s) = 3.155441e-01
```

Data processing:

```
time2 = np.array([3.155441, 3.187609, 3.574691, 3.661902])
E_p2 = np.ones(shape=(4,1))*time2[0]/time2
```

Plot the figure:

```
plt(P,E_p2,'o--')
plt.plot(range(10),np.ones(len(range(10))),'k')
plt.xlabel('Number of process')
plt.ylabel('Efficiency')
plt.title('Isogranular scaling')
plt.xlim((0,9))
plt.ylim((0,1.2))
plt.text(s='Ideal',x=5,y=1.02)
plt.text(s='Measured',x=6,y=0.8)
plt.savefig('./isogranular_scaling.png',dpi=300)
```