Lab 6c - CPI con MPI e con Python

https://elly2023.smfi.unipr.it/mod/page/view.php?id=5360

OBIETTIVO
Parallelizzazione di mpi_cpi.c tramite scomposizione del dominio 1D in sottodomini.

CPI

Attività svolte

Dopo aver creato la directory di lavoro "*mkdir ~/HPC2324/mpi/cpi/*", ho copiato al suo interno il file cpi.c che ho rinominato in mpi_cpi.c.
Inoltre ho copiato gli esempi di script slurm e python per l'analisi dello scaling da "*cp /hpc/home/roberto.alfieri/SHARE/mpi/cpi/* .*"

Ho modificato il programma mpi_cpi.c per renderlo parallelo.

mpi_cpi.c

```
// GNU Profiling    https://users.cs.duke.edu/~ola/courses/programming/gprof.html
// gcc cpi.c -pg  -o cpi-pg -lm
// ./cpi-pg
// gprof cpi-pg

#include <stdio.h>
#include <math.h>
#include <unistd.h>    //optarg
#include <time.h>      //gettime
#include <stdlib.h>
#include <mpi.h>

void options(int argc, char * argv[]) ;
void usage(char * argv[]);
double f1 (int Lfirst, int Llast);
double f2 (int Lfirst, int Llast);

long int n=1000000000;    // intervalli
int s=0;                 // tempo sleep
double h;
char hostname[100];

#define BILLION  1000000000L;

double ta,tb,tc,td;
//double tnp, tp1, tp2;
double sum1, sum2;
double pi1, pi2;

/*****************************************/

int main( int argc, char *argv[])
{
    int mpiTasks, mpiRank, taskSize, Lfirst, Llast;
    double Lsum1, Lsum2;
    char hostname[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &mpiTasks);
    MPI_Comm_rank(MPI_COMM_WORLD, &mpiRank);

    taskSize= n/mpiTasks;  //dimensione di ogni sottodominio
    Lfirst=(n/mpiTasks)*mpiRank+1;  // prima posizione
    Llast=(n/mpiTasks)*mpiRank+(n/mpiTasks);  // ultima posizione

    ta = MPI_Wtime();

    double  PI = 3.14159265358979323846264338327950288 ;

    gethostname(hostname, 100);
```

```
    options(argc, argv);  /* optarg management */

    h = 1.0 / (double) n;

    sleep (s);  // Simulazione codice non paralellizzabile

    tb = MPI_Wtime();

    Lsum1= f1(Lfirst, Llast);
    MPI_Reduce(&Lsum1, &sum1, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

    tc = MPI_Wtime();

    Lsum2= f2(Lfirst, Llast);
    MPI_Reduce(&Lsum2, &sum2, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

    td = MPI_Wtime();

    pi1 = 4 * h * sum1;
    pi2 = 4 * h * sum2;

    /*tnp = (double) ( tb.tv_sec  - ta.tv_sec )
        + (double) ( tb.tv_nsec - ta.tv_nsec )/ BILLION;

    tp1 = (double) ( tc.tv_sec  - tb.tv_sec )
        + (double) ( tc.tv_nsec - tb.tv_nsec )/ BILLION;

    tp2 = (double) ( td.tv_sec  - tc.tv_sec )
        + (double) ( td.tv_nsec - tc.tv_nsec )/ BILLION;*/


    if (mpiRank == 0){
        fprintf(stderr,"#Tasks       Inter        time1       time2       time3        error1       error2
hostname \n");
        fprintf(stdout,"MPI, %d, %ld,  %.5f,  %.5f,  %.5f, %.8e, %.8e, %s \n",
              mpiTasks,  n,  td-ta, tc-tb, td-tc, fabs(pi1-PI), fabs(pi2-PI), hostname);
    }

    MPI_Finalize();
    return 0;
}

double f1 (int Lfirst, int Llast)
{
    long int i;
    double x, sum=0.0;
    for (i = Lfirst; i <= Llast; i++)
    {
       x = h * ((double)i - 0.5);
       sum += sqrt(1-x*x) ;
    }
    return sum;
}

double f2 (int Lfirst, int Llast)
{
    long int i;
    double x, sum=0.0;
    for (i = Lfirst; i <= Llast; i++)
    {
       x = h * ((double)i - 0.5);
       sum += (1.0 / (1.0 + x*x));
    }
    return sum;
}

/********************************************/

void options(int argc, char * argv[])
{
  int i;
   while ( (i = getopt(argc, argv, "n:s:h")) != -1) {
        switch (i)
        {
```

```
        case 'n':  n  = strtol(optarg, NULL, 10);  break;
        case 's':  s  = strtol(optarg, NULL, 10);  break;
        case 'h':  usage(argv); exit(1);
        case '?':  usage(argv); exit(1);
        default:   usage(argv); exit(1);
        }
    }
}

/**************************************/

void usage(char * argv[])
{
  printf ("\n%s [-n intervals] [-s sleep] [-h]",argv[0]);
  printf ("\n");
}
```

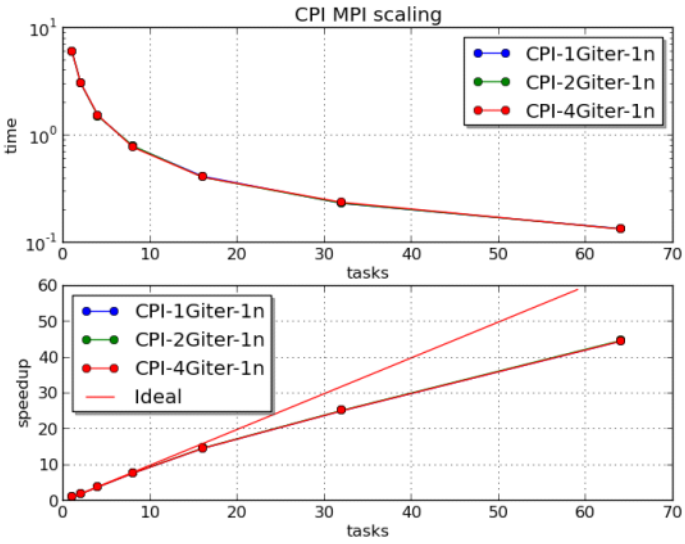
Output (esecuzione mpi_cpi_scaling.slurm)

```
[martina.genovese@ui01 cpi]$ more mpi_cpi_scaling.dat
MPI, 1, 1000000000,  6.01052,  4.89457,  1.11595, 1.04805054e-13, 1.77635684e-13, wn24
MPI, 2, 1000000000,  3.07063,  2.50073,  0.56989, 1.14130927e-13, 1.07913678e-13, wn24
MPI, 4, 1000000000,  1.51765,  1.23788,  0.27976, 2.39808173e-14, 2.79776202e-14, wn24
MPI, 8, 1000000000,  0.79149,  0.64771,  0.14377, 9.01501096e-14, 2.44249065e-14, wn24
MPI, 16, 1000000000,  0.41701,  0.34122,  0.07579, 1.28785871e-14, 3.95239397e-14, wn24
MPI, 32, 1000000000,  0.23383,  0.19016,  0.04366, 4.44089210e-14, 3.59712260e-14, wn24
MPI, 64, 1000000000,  0.13579,  0.11131,  0.02447, 3.90798505e-14, 2.22044605e-14, wn24
MPI, 1, 2000000000,  6.03579,  4.92050,  1.11529, 1.22836970e+00, 1.28700222e+00, wn24
MPI, 2, 2000000000,  3.06338,  2.49449,  0.56888, 1.22836970e+00, 1.28700222e+00, wn24
MPI, 4, 2000000000,  1.50790,  1.22835,  0.27954, 1.22836970e+00, 1.28700222e+00, wn24
MPI, 8, 2000000000,  0.79975,  0.65189,  0.14786, 1.22836970e+00, 1.28700222e+00, wn24
MPI, 16, 2000000000,  0.40922,  0.33353,  0.07568, 1.22836970e+00, 1.28700222e+00, wn24
MPI, 32, 2000000000,  0.23418,  0.19278,  0.04139, 1.22836970e+00, 1.28700222e+00, wn24
MPI, 64, 2000000000,  0.13525,  0.10955,  0.02570, 1.22836970e+00, 1.28700222e+00, wn24
MPI, 1, 4000000000,  6.00959,  4.89339,  1.11619, 2.15210923e+00, 2.16167800e+00, wn24
MPI, 2, 4000000000,  3.05797,  2.48934,  0.56862, 2.15210923e+00, 2.16167800e+00, wn24
MPI, 4, 4000000000,  1.53527,  1.25396,  0.28131, 2.15210923e+00, 2.16167800e+00, wn24
MPI, 8, 4000000000,  0.77590,  0.63146,  0.14444, 2.15210923e+00, 2.16167800e+00, wn24
MPI, 16, 4000000000,  0.40963,  0.33381,  0.07582, 2.15210923e+00, 2.16167800e+00, wn24
MPI, 32, 4000000000,  0.23869,  0.19459,  0.04410, 2.15210923e+00, 2.16167800e+00, wn24
MPI, 64, 4000000000,  0.13478,  0.10941,  0.02536, 2.15210923e+00, 2.16167800e+00, wn24
```

Output (esecuzione mpi_cpi_scaling.py)

```
[martina.genovese@ui01 cpi]$ python mpi_cpi_scaling.py
    par  nt           n        t       t1       t2        e1        e2    host
14  MPI   1  4000000000  6.00959  4.89339  1.11619  2.152109  2.161678    wn24
15  MPI   2  4000000000  3.05797  2.48934  0.56862  2.152109  2.161678    wn24
16  MPI   4  4000000000  1.53527  1.25396  0.28131  2.152109  2.161678    wn24
17  MPI   8  4000000000  0.77590  0.63146  0.14444  2.152109  2.161678    wn24
18  MPI  16  4000000000  0.40963  0.33381  0.07582  2.152109  2.161678    wn24
19  MPI  32  4000000000  0.23869  0.19459  0.04410  2.152109  2.161678    wn24
20  MPI  64  4000000000  0.13478  0.10941  0.02536  2.152109  2.161678    wn24
```

Plot (generato da mpi_cpi_scaling.py)



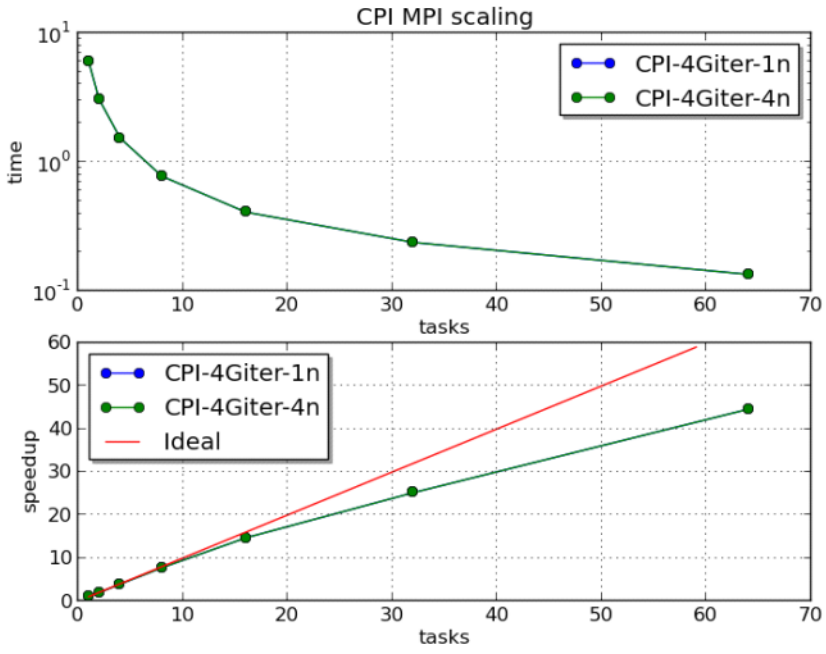
Output (esecuzione mpi_cpi_scaling-4.slurm)

```
[martina.genovese@ui01 cpi]$ more mpi_cpi_scaling-4.dat
MPI, 1, 1000000000,  6.40658,  5.21609,  1.19049, 1.04805054e-13, 1.77635684e-13, wn21
MPI, 2, 1000000000,  3.30898,  2.69274,  0.61624, 1.14130927e-13, 1.07913678e-13, wn21
MPI, 4, 1000000000,  1.72123,  1.40123,  0.31999, 2.39808173e-14, 2.79776202e-14, wn21
MPI, 8, 1000000000,  0.86551,  0.70521,  0.16030, 9.01501096e-14, 2.44249065e-14, wn21
MPI, 16, 1000000000,  0.52635,  0.42880,  0.09754, 1.28785871e-14, 3.95239397e-14, wn21
MPI, 32, 1000000000,  1.51784,  1.29884,  0.21899, 4.44089210e-14, 3.59712260e-14, wn21
MPI, 1, 2000000000,  6.42325,  5.23206,  1.19119, 1.22836970e+00, 1.28700222e+00, wn21
MPI, 2, 2000000000,  3.31419,  2.68814,  0.62605, 1.22836970e+00, 1.28700222e+00, wn21
MPI, 4, 2000000000,  1.72078,  1.40132,  0.31945, 1.22836970e+00, 1.28700222e+00, wn21
MPI, 8, 2000000000,  0.86419,  0.70392,  0.16026, 1.22836970e+00, 1.28700222e+00, wn21
MPI, 16, 2000000000,  0.52636,  0.42863,  0.09772, 1.22836970e+00, 1.28700222e+00, wn21
MPI, 32, 2000000000,  1.51150,  1.26490,  0.24660, 1.22836970e+00, 1.28700222e+00, wn21
MPI, 1, 4000000000,  6.40486,  5.21334,  1.19151, 2.15210923e+00, 2.16167800e+00, wn21
MPI, 2, 4000000000,  3.31087,  2.69302,  0.61784, 2.15210923e+00, 2.16167800e+00, wn21
MPI, 4, 4000000000,  1.72139,  1.40248,  0.31890, 2.15210923e+00, 2.16167800e+00, wn21
MPI, 8, 4000000000,  0.86563,  0.70545,  0.16017, 2.15210923e+00, 2.16167800e+00, wn21
MPI, 16, 4000000000,  0.52641,  0.42883,  0.09757, 2.15210923e+00, 2.16167800e+00, wn21
MPI, 32, 4000000000,  1.55661,  1.30031,  0.25629, 2.15210923e+00, 2.16167800e+00, wn21
[martina.genovese@ui01 cpi]$ sbatch mpi_cpi_scaling.slurm
```

Output (esecuzione mpi_cpi_scaling-4.py)

```
[martina.genovese@ui01 cpi]$ python mpi_cpi_scaling-4.py
    par  nt           n        t       t1       t2        e1        e2    host
14  MPI   1  4000000000  6.00959  4.89339  1.11619  2.152109  2.161678   wn24
15  MPI   2  4000000000  3.05797  2.48934  0.56862  2.152109  2.161678   wn24
16  MPI   4  4000000000  1.53527  1.25396  0.28131  2.152109  2.161678   wn24
17  MPI   8  4000000000  0.77590  0.63146  0.14444  2.152109  2.161678   wn24
18  MPI  16  4000000000  0.40963  0.33381  0.07582  2.152109  2.161678   wn24
19  MPI  32  4000000000  0.23869  0.19459  0.04410  2.152109  2.161678   wn24
20  MPI  64  4000000000  0.13478  0.10941  0.02536  2.152109  2.161678   wn24
```

Plot (generato da mpi_cpi_scaling-4.py)



CPI con Python

Osservazioni (note prese da Elly):

- **cpi.py** è la traduzione in python del programma cpi.c
- lo script **cpi.slurm** esegue e mette a confronto i tempi di esecuzione delle due versioni.
- La parallelizzazione a memoria distribuita con Python può essere fatta utilizzando diverse librerie. Le principali sono:
  - **Mpi4py**: è il porting in Python della libreria MPI. Vedi ad esempio la guida di mpi4py del cluster

HPC
    - **python multiprocessing**: è il modo nativo di Python per la gestione dei processi paralleli.
- In **pym_cpi.py** il calcolo di pigreco viene parallelizzato utilizzando Python Multiprocessing
- Lo script **mpi_pym_cpi_scaling.slurm** determina lo strong scaling mettendo a confronto le prestazioni della versione Python Multiprocessing con la versione MPI in C. Eseguire il calcolo e generare i plot di confronto.

Attività svolte

Ho eseguito lo script cpi.slurm, il quale mi ha restituito come output il file cpi.dat con il seguente contenuto:

```
[martina.genovese@ui01 python]$ more cpi.dat
C,  100000000,  1.04760645e-12, 6.33271213e-13,  0.46124, 0.30372, 0.15752, wn99
PY, 100000000, 1.0476064460362977e-12, 6.332712132461893e-13, 46.94,  26.24,  20.7, wn99
```

Ho eseguito lo script mpi_pym_cpi_scaling.slurm, il quale mi ha restituito come output il file mpi_pym_cpi_scaling.dat con il seguente contenuto:

```
[martina.genovese@ui01 python]$ cat mpi_pym_cpi_scaling.dat
MPI, 1, 100000000,  8.04730,  6.70346,  1.34383, nan, 2.74291804e+00, wn22
PYM, 1, 100000000,   35.89, 20.09, 15.81,  1.0476064460362977e-12, 6.332712132461893e-13, wn22
#PYM-f2 0/1 100000000 1-100000001
MPI, 2, 100000000,  4.22698,  3.52228,  0.70469, nan, 2.74291804e+00, wn22
PYM, 2, 100000000,   17.74, 9.94, 7.8,  5.022648963404208e-13, 1.1679546219056647e-13, wn22
#PYM-f2 0/2 100000000 1-50000001
#PYM-f2 1/2 100000000 50000001-100000001
MPI, 4, 100000000,  1.97912,  1.63851,  0.34060, nan, 2.74291804e+00, wn22
PYM, 4, 100000000,   9.41, 5.06, 4.36,  4.476419235288631e-13, 1.1057821325266559e-13, wn22
#PYM-f2 2/4 100000000 50000001-75000001
#PYM-f2 1/4 100000000 25000001-50000001
#PYM-f1 2/4 100000000 50000001-75000001
MPI, 8, 100000000,  1.06096,  0.88467,  0.17628, nan, 2.74291804e+00, wn22
PYM, 8, 100000000,   5.26, 2.97, 2.29,  3.228528555609955e-13, 2.1760371282653068e-14, wn22
#PYM-f2 2/8 100000000 25000001-37500001
#PYM-f2 7/8 100000000 87500001-100000001
#PYM-f2 1/8 100000000 12500001-25000001
#PYM-f2 4/8 100000000 50000001-62500001
#PYM-f2 5/8 100000000 62500001-75000001
#PYM-f1 2/8 100000000 25000001-37500001
#PYM-f1 1/8 100000000 12500001-25000001
#PYM-f1 6/8 100000000 75000001-87500001
#PYM-f1 7/8 100000000 87500001-100000001
#PYM-f1 4/8 100000000 50000001-62500001
#PYM-f1 5/8 100000000 62500001-75000001
MPI, 16, 100000000,  0.53681,  0.44808,  0.08873, nan, 2.74291804e+00, wn22
PYM, 16, 100000000,   2.72, 1.49, 1.22,  3.8058445284150366e-13, 8.926193117986259e-14, wn22
#PYM-f2 7/16 100000000 43750001-50000001
#PYM-f2 3/16 100000000 18750001-25000001
#PYM-f2 5/16 100000000 31250001-37500001
#PYM-f2 9/16 100000000 56250001-62500001
#PYM-f2 14/16 100000000 87500001-93750001
#PYM-f2 1/16 100000000 6250001-12500001
#PYM-f2 6/16 100000000 37500001-43750001
#PYM-f2 11/16 100000000 68750001-75000001
#PYM-f1 15/16 100000000 93750001-100000001
#PYM-f1 11/16 100000000 68750001-75000001
#PYM-f1 7/16 100000000 43750001-50000001
#PYM-f1 2/16 100000000 12500001-18750001
#PYM-f1 6/16 100000000 37500001-43750001
#PYM-f1 5/16 100000000 31250001-37500001
MPI, 32, 100000000,  0.29629,  0.24741,  0.04887, nan, 2.74291804e+00, wn22
PYM, 32, 100000000,   1.77, 0.98, 0.79,  3.077538224260934e-13, 1.6431300764452317e-14, wn22
#PYM-f2 13/32 100000000 40625001-43750001
#PYM-f2 15/32 100000000 46875001-50000001
#PYM-f2 7/32 100000000 21875001-25000001
#PYM-f2 10/32 100000000 31250001-34375001
#PYM-f2 0/32 100000000 1-3125001
#PYM-f2 2/32 100000000 6250001-9375001
#PYM-f2 6/32 100000000 18750001-21875001
#PYM-f2 8/32 100000000 25000001-28125001
#PYM-f2 9/32 100000000 28125001-31250001
#PYM-f2 12/32 100000000 37500001-40625001
#PYM-f2 11/32 100000000 34375001-37500001
#PYM-f2 17/32 100000000 53125001-56250001
#PYM-f2 16/32 100000000 50000001-53125001
#PYM-f2 14/32 100000000 43750001-46875001
#PYM-f2 5/32 100000000 15625001-18750001
#PYM-f2 3/32 100000000 9375001-12500001
#PYM-f2 20/32 100000000 62500001-65625001
#PYM-f2 19/32 100000000 59375001-62500001
```

```
#PYM-f2 18/32 100000000 56250001-59375001
#PYM-f2 4/32 100000000 12500001-15625001
#PYM-f2 23/32 100000000 71875001-75000001
#PYM-f2 24/32 100000000 75000001-78125001
#PYM-f2 21/32 100000000 65625001-68750001
#PYM-f2 25/32 100000000 78125001-81250001
#PYM-f2 29/32 100000000 90625001-93750001
#PYM-f2 31/32 100000000 96875001-100000001
#PYM-f2 30/32 100000000 93750001-96875001
#PYM-f2 28/32 100000000 87500001-90625001
#PYM-f2 27/32 100000000 84375001-87500001
#PYM-f2 1/32 100000000 3125001-6250001
#PYM-f2 22/32 100000000 68750001-71875001
#PYM-f2 26/32 100000000 81250001-84375001
#PYM-f1 16/32 100000000 50000001-53125001
#PYM-f1 2/32 100000000 6250001-9375001
#PYM-f1 3/32 100000000 9375001-12500001
#PYM-f1 1/32 100000000 3125001-6250001
#PYM-f1 6/32 100000000 18750001-21875001
#PYM-f1 4/32 100000000 12500001-15625001
#PYM-f1 10/32 100000000 31250001-34375001
#PYM-f1 5/32 100000000 15625001-18750001
#PYM-f1 9/32 100000000 28125001-31250001
#PYM-f1 13/32 100000000 40625001-43750001
#PYM-f1 15/32 100000000 46875001-50000001
#PYM-f1 11/32 100000000 34375001-37500001
#PYM-f1 8/32 100000000 25000001-28125001
#PYM-f1 19/32 100000000 59375001-62500001
#PYM-f1 12/32 100000000 37500001-40625001
#PYM-f1 14/32 100000000 43750001-46875001
#PYM-f1 18/32 100000000 56250001-59375001
#PYM-f1 23/32 100000000 71875001-75000001
#PYM-f1 24/32 100000000 75000001-78125001
#PYM-f1 31/32 100000000 96875001-100000001
#PYM-f1 21/32 100000000 65625001-68750001
#PYM-f1 28/32 100000000 87500001-90625001
#PYM-f1 25/32 100000000 78125001-81250001
#PYM-f1 27/32 100000000 84375001-87500001
#PYM-f1 30/32 100000000 93750001-96875001
#PYM-f1 29/32 100000000 90625001-93750001
#PYM-f1 17/32 100000000 53125001-56250001
#PYM-f1 0/32 100000000 1-3125001
#PYM-f1 26/32 100000000 81250001-84375001
#PYM-f1 20/32 100000000 62500001-65625001
#PYM-f1 22/32 100000000 68750001-71875001
#PYM-f1 7/32 100000000 21875001-25000001
```

Ho eseguito il programma in python mpi_pym_cpi_scaling.py per generare un plot (mpi_pym_cpi_scaling.png) che metta a confronto le prestazioni della versione Python Multiprocessing con la versione MPI in C.

mpi_pym_cpi_scaling.py

mpi_pym_cpi_scaling.py

```
#!/usr/bin/env python2

import matplotlib
matplotlib.use('Agg')   # Backend per PNG
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("mpi_pym_cpi_scaling.dat", names=["par","nt","n","t","t1","t2","e1","e2","host"])

#print(df)

dfM=df[ df["par"]=="MPI" ]
dfP=df[ df["par"]=="PYM" ]

#print (dfM)
#print (dfP)


plt.subplot(2,1,1)

plt.title('CPI MPI scaling ')
plt.grid()
plt.xlabel('tasks')
plt.ylabel('time')
plt.yscale('log')
plt.plot(dfM.nt,  dfM.t, '-o', label="MPI CPI-100Miter")
plt.plot(dfP.nt,  dfP.t, '-o', label="PYM CPI-100Miter")
plt.legend(shadow=True,loc="best")

plt.subplot(2,1,2)

plt.grid()
plt.xlabel('tasks')
plt.ylabel('speedup')
plt.plot(dfM.nt,  dfM.t.iloc[0]/dfM.t,  '-o', label="MPI CPI-100Miter")
plt.plot(dfP.nt,  dfP.t.iloc[0]/dfP.t,  '-o', label="PYM CPI-100Miter")
plt.plot(range(1,30),range(1,30),'-r', label='Ideal')
plt.legend(shadow=True,loc="best")

plt.savefig('mpi_pym_cpi_scaling.png')
```

mpi_pym_cpi_scaling.png