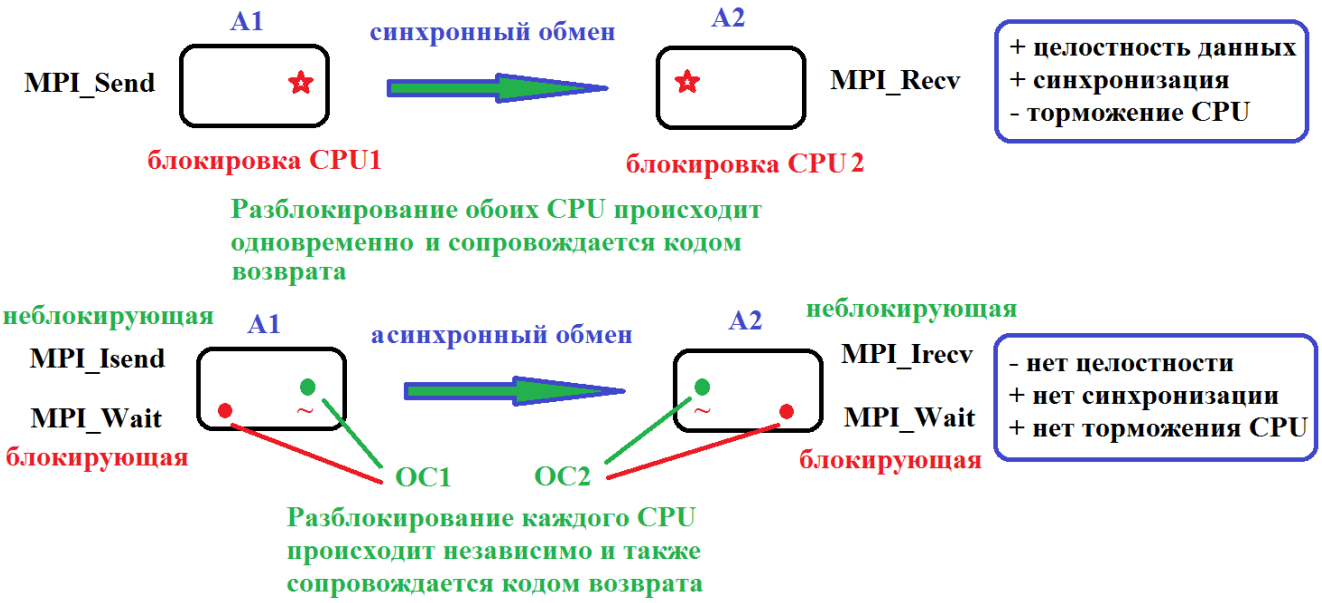
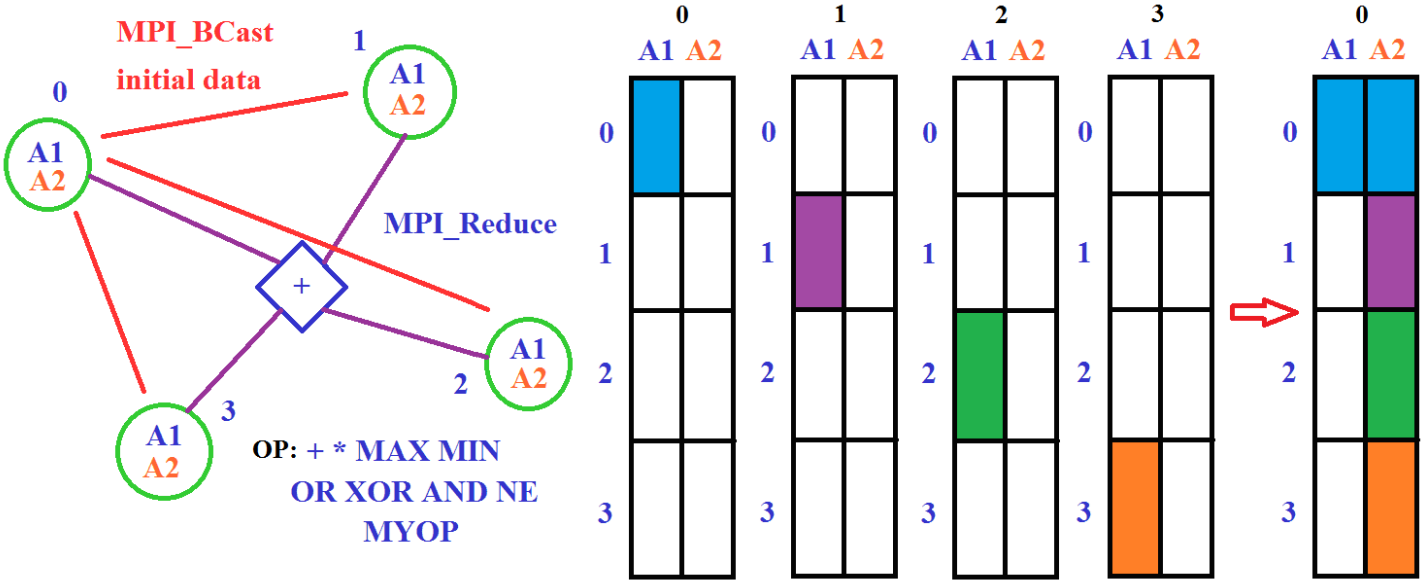
Семинар 5. Библиотека MPI. Асинхронные обмены. Коллективные операции.

1. Библиотека MPI. Дополнительные функции.

1) Асинхронные обмены – MPI_Isend, MPI_Irecv, MPI_Wait, MPI_Waitall.



2) Коллективные обмены и операции – MPI_Bcast, MPI_Reduce, MPI_Allreduce.



2. Задачи численного интегрирования и распараллеливание.

2.1. Квадратуры.

$$I_{1D}=\int_a^b f(x)\,dx \approx \sum_{i=1}^{N_x} f(x_i)h_x,\quad x_i=a+h_x(i-0.5),\quad h_x=\frac{b-a}{N_x};$$

$$I_{2D}=\int\limits_{D=[a,b]\times[c,d]} f(x,y)\,dxdy \approx \sum_{j=1}^{N_y}\sum_{i=1}^{N_x} f(x_i,y_j)h_xh_y,\quad y_j=c+h_y(j-0.5),\quad h_y=\frac{d-c}{N_y};$$

$$I_{3D}=\int\limits_{D=[a,b]\times[c,d]\times[e,g]} f(x,y,z)\,dxdydz \approx \sum_{k=1}^{N_z}\sum_{j=1}^{N_y}\sum_{i=1}^{N_x} f(x_i,y_j,z_k)h_xh_yh_z,\quad z_k=e+h_z(k-0.5),\quad h_z=\frac{g-e}{N_z};$$

$$I_{nD}=\int\limits_{D=\prod_{k=1}^{n}[a_k,b_k]} f(\mathbf{x})\,d\mathbf{x},\quad \mathbf{x}=(x_1,\ldots,x_n),\quad h_k=\frac{b_k-a_k}{N_k},\quad I_{nD}\approx \sum_{i_1=1}^{N_1}\ldots\sum_{i_n=1}^{N_n} f(x_{1,i_1},\ldots,x_{n,i_n})\prod_{k=1}^{n}h_k.$$

2.2. Два способа распараллеливания:

1)
$$I_{1D}=\int_a^b f(x)dx=\sum_{m=0}^{p-1} I_m,\quad I_m=\int_{a_m}^{b_m} f(x)dx\approx\sum_{i=1}^{N_p} f\left(a_m+(i-0.5)h_m\right)h_m,$$
$$a_m=a+h_p m,\quad b_m=a_m+h_p(m+1),\quad h_p=\frac{b-a}{p},\quad N_p=\frac{N}{p},\quad h_m=\frac{b_m-a_m}{N_p}.$$

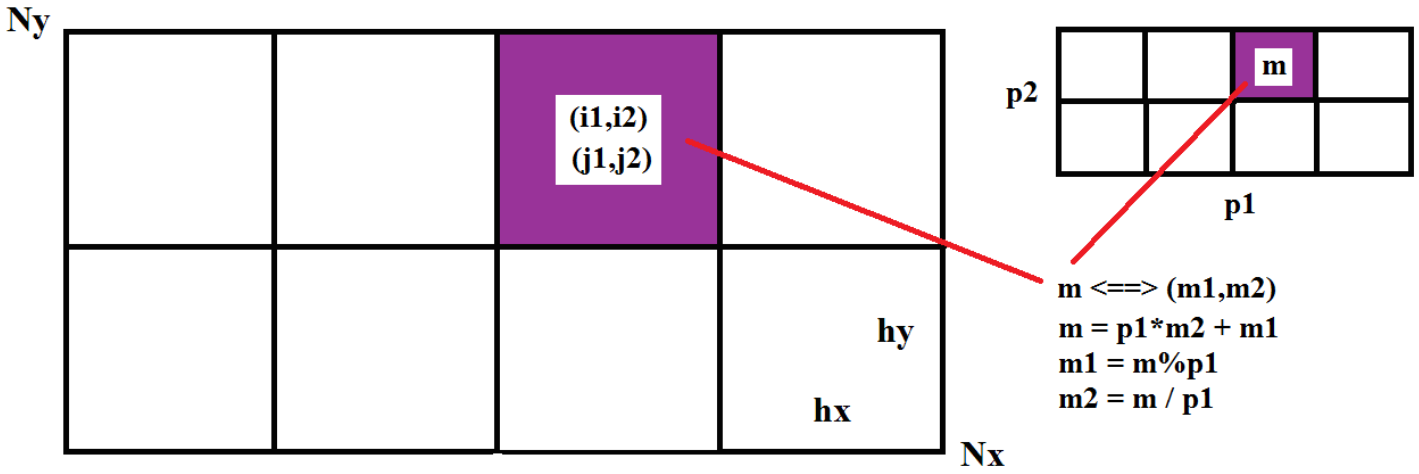
Смысл способа: сначала распараллеливаем, затем дискретизируем.

2)
$$I_{1D}=\int_a^b f(x)dx\approx\sum_{i=1}^{N_x} f(x_i)h_x=\sum_{m=0}^{p-1} S_m,\quad S_m=\sum_{i=i_1^{(m)}}^{i_2^{(m)}} f(x_i)h_x,$$
$$i_1^{(m)}=1+mN_p,\quad i_2^{(m)}=(m+1)N_p,\quad N_p=\frac{N}{p}.$$

Смысл способа: сначала дискретизируем, затем распараллеливаем.

Более правильно делать вторым способом, т.к. тогда ситуация проще и точность не зависит от количества вычислителей!

2.3. Топология расчетной сетки и топология в пространстве вычислителей.



N=Nx*Ny ==> p=p1*p2

Разбиение интеграла после дискретизации на частичные суммы:

$$I_{2D}=\int\limits_{D=[a,b]\times[c,d]} f(x,y)dxdy\approx\sum_{j=1}^{N_y}\sum_{i=1}^{N_x} f(x_i,y_j)h_x h_y=\sum_{m_2=0}^{p_2-1}\sum_{m_1=0}^{p_1-1} S_{m_1,m_2},\quad S_{m_1,m_2}=\sum_{j=j_1^{(m)}}^{j_2^{(m)}}\sum_{i=i_1^{(m)}}^{i_2^{(m)}} f(x_i,y_j)h_x h_y.$$

Вычислитель с номером m имеет координаты (m_1,m_2) в индексном пространстве $M=\{(m_1,m_2),\ m_1=0,...,p_1-1,\ m_2=0,...,p_2-1\}$ вычислителей и будет обрабатывать фрагмент $\Omega_m=\{(i,j),\ i=i_1^{(m)},...,i_2^{(m)},\ j=j_1^{(m)},...,j_2^{(m)}\}$ из множества индексов $\Omega=\{(i,j),\ i=1,...,N_x,\ j=1,...,N_y\}$ сетки.

Критерий выбора решетки вычислителей:

$$W_{2D}=\left(\frac{1.0\cdot N_x}{N_y}-\frac{1.0\cdot p_1}{p_2}\right)^2\to\min\quad \textit{при условии}\quad p_1\cdot p_2=p;$$

$$W_{3D}=\left(\frac{1.0\cdot N_x}{N_y}-\frac{1.0\cdot p_1}{p_2}\right)^2+\left(\frac{1.0\cdot N_x}{N_z}-\frac{1.0\cdot p_1}{p_3}\right)^2\to\min\quad \textit{при условии}\quad p_1\cdot p_2\cdot p_3=p;$$

$$W_{mD}=\sum_{k=2}^{m}\left(\frac{1.0\cdot N_1}{N_k}-\frac{1.0\cdot p_1}{p_k}\right)^2\to\min\quad \textit{при условии}\quad p_1\cdot p_2\cdot\ldots\cdot p_m=p.$$

Решение находится либо полным перебором в целых числах,
либо приближенно методами оптимизации.

3. Реализация примеров.

Пример 1 – вычисление двойного интеграла в прямоугольнике [a,b]x[c,d] (ex08a.c).

```
#include <stdio.h>
#include <math.h>
#include "mynet.h"

double f1(double x); double f1(double x) { double e=exp(x); return 0.5*(e+1.0/e); }
double f2(double x); double f2(double x) { double e=exp(x); return 0.5*(e-1.0/e); }
double f(double x, double y); double f(double x, double y) { return f1(x)*f2(y); }

static int np, mp, nl, np1, np2, mp1, mp2, iflag;
static char pname[MPI_MAX_PROCESSOR_NAME]; static MPI_Status status;
static double tick, t1, t2, t3;
static double a=0.0, b=1.0, c=0.0, d=1.0; static int nx=10000, ny=10000;


int main(int argc, char *argv[])
{
  int i, i1, i2, j, j1, j2; double s, p, hx, hy, hxy, x, y;
  MyNetInit(&argc,&argv,&np,&mp,&nl,pname,&tick);
  if (argc<3) mpierr("Too small arguments",1);
  sscanf(argv[1],"%d",&np1); sscanf(argv[2],"%d",&np2); sscanf(argv[3],"%d",&iflag);
  if (np1*np2!=np) mpierr("Bad processor grid",2);
  mp2 = mp / np1; mp1 = mp % np1;
  fprintf(stderr,"mp=%d grid=%dx%d coord=(%d,%d)\n",mp,np1,np2,mp1,mp2);
  t1 = MPI_Wtime();
  hx=(b-a)/nx; hy=(d-c)/ny; hxy=hx*hy;
  if (np1==1) { i1 = 0; i2 = nx-1; }
  else { i1 = mp1 * (nx/np1); if (mp1<np1-1) i2 = i1+(nx/np1)-1; else i2 = nx-1; }
  if (np2==1) { j1 = 0; j2 = ny-1; }
  else { j1 = mp2 * (ny/np2); if (mp2<np2-1) j2 = j1+(ny/np2)-1; else j2 = ny-1; }
  fprintf(stderr,"mp=%d i1=%d i2=%d j1=%d j2=%d\n",mp,i1,i2,j1,j2);
  s = 0;
  for (j=j1; j<=j2; j++) {
    y = c + (j*1.0+0.5)*hy;
    for (i=i1; i<=i2; i++) {
      x = a + (i*1.0+.5)*hx; s = s + hxy * f(x,y);
    }
  }
  t2 = MPI_Wtime(); t1 = t2 - t1;
  if (np==1)
    t2 = 0;
  else {
    p = s;
    if (iflag==0) MPI_Reduce(&p, &s, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
    else          MPI_Allreduce(&p, &s, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
    t2 = MPI_Wtime()-t2;
  }
  t3 = t1 + t2;
  fprintf(stderr,"mp=%d t1=%le t2=%le t3=%le int=%le\n",mp,t1,t2,t3,s);
  MPI_Finalize();
  return 0;
}
```

Трансляция:

```
>mpicc -o ex08a.px -O2 ex08a.c mynet.c -lm
```

Результаты запуска:

```
>mpirun -np 1 -nolocal -machinefile hosts ex08a.px 1 1 0 (запуск в классе)
>mpirun -np 1 ex08a.px 1 1 0                             (запуск на сервере)
mp=0 grid=1x1 coord=(0,0)
mp=0 i1=0 i2=9999 j1=0 j2=9999
mp=0 t1=1.811444e+01 t2=0.000000e+00 t3=1.811444e+01 int=6.382290e-01

>mpirun -np 1 -nolocal -machinefile hosts ex08a.px 2 1 0 (запуск в классе)
>mpirun -np 1 ex08a.px 2 1 0                             (запуск на сервере)
mp=0 grid=2x1 coord=(0,0)
mp=1 grid=2x1 coord=(1,0)
```

```
mp=0 i1=0 i2=4999 j1=0 j2=9999
mp=1 i1=5000 i2=9999 j1=0 j2=9999
mp=0 t1=9.298582e+00 t2=4.600000e-05 t3=9.298628e+00 int=6.382290e-01
mp=1 t1=8.900826e+00 t2=5.300000e-05 t3=8.900879e+00 int=3.552322e-01

>mpirun -np 1 -nolocal -machinefile hosts ex08a.px 2 2 0  (запуск в классе)
>mpirun -np 1 ex08a.px 2 2 0                              (запуск на сервере)
mp=0 grid=2x2 coord=(0,0)
mp=1 grid=2x2 coord=(1,0)
mp=2 grid=2x2 coord=(0,1)
mp=3 grid=2x2 coord=(1,1)
mp=0 i1=0 i2=4999 j1=0 j2=4999
mp=1 i1=5000 i2=9999 j1=0 j2=4999
mp=2 i1=0 i2=4999 j1=5000 j2=9999
mp=3 i1=5000 i2=9999 j1=5000 j2=9999
mp=3 t1=4.388040e+00 t2=5.284000e-03 t3=4.393324e+00 int=2.717513e-01
mp=1 t1=4.502950e+00 t2=4.100000e-05 t3=4.502991e+00 int=8.348090e-02
mp=2 t1=4.659110e+00 t2=5.200000e-05 t3=4.659162e+00 int=2.164915e-01
mp=0 t1=4.698314e+00 t2=5.500000e-05 t3=4.698369e+00 int=6.382290e-01

Пример 2 – вычисление двойного интеграла, ввод/вывод в файл, асинхр. пересылки (ex08b.c).
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "mycom.h" #include "mynet.h"
double f1(double x); double f1(double x) { double e=exp(x); return 0.5*(e+1.0/e); }
double f2(double x); double f2(double x) { double e=exp(x); return 0.5*(e-1.0/e); }
double f(double x, double y); double f(double x, double y) { return f1(x)*f2(y); }
static int np, mp, nl, np1, np2, mp1, mp2, ier;
static char pname[MPI_MAX_PROCESSOR_NAME]; static MPI_Status status;
static char sname[13] = "ex08b.p00";
static MPI_Request Req[100];
static union_t buf; static double t0, t1, t2, t3;
static FILE *Fi = NULL; static FILE *Fo = NULL;
static int nx, ny, fl; static double xa, xb, ya, yb;
int main(int argc, char *argv[])
{
  int i, i1, i2, j, j1, j2; double s, p, x, y, hx, hy, hxy;
  MyNetInit(&argc,&argv,&np,&mp,&nl,pname,&tick);
  sprintf(sname+11,"%02d",mp);
  ier = fopen_m(&Fo,sname,"wt");
  if (ier!=0) mpierr("Protocol file not opened",1);
  if (mp==0) {
    ier = fopen_m(&Fi,"ex08b.d","rt");
    if (ier!=0) mpierr("Data file not opened",2);
    fscanf(Fi,"%le\n %le\n %le\n %le\n %d\n %d\n",&xa,&xb,&ya,&yb,&nx,&ny,&fl);
    fclose_m(&Fi);
  }
  MPI_Bcast(&fl,1,MPI_INT,0,MPI_COMM_WORLD);
  if (np>1) {
    if (fl<1){ // variant 1:
      if (mp==0) {
        buf.ddata[0] = xa; buf.ddata[1] = xb;
        buf.ddata[2] = ya; buf.ddata[3] = yb;
        buf.idata[8] = nx; buf.idata[9] = ny;
        for (i=1; i<np; i++)
          MPI_Isend(buf.ddata,5,MPI_DOUBLE,i,MY_TAG,MPI_COMM_WORLD,Req+i);
        MPI_Waitall(np-1,Req+1,&status);
      }
      else {
        MPI_Recv(buf.ddata,5,MPI_DOUBLE,0,MY_TAG,MPI_COMM_WORLD,&status);
        xa = buf.ddata[0]; xb = buf.ddata[1];
        ya = buf.ddata[2]; yb = buf.ddata[3];
        nx = buf.idata[8]; ny = buf.idata[9];
      }
      MPI_Barrier(MPI_COMM_WORLD);
    }
```

```
    else { // variant 2:
      if (mp==0) {
        buf.ddata[0] = xa; buf.ddata[1] = xb;
        buf.ddata[2] = ya; buf.ddata[3] = yb;
        buf.idata[8] = nx; buf.idata[9] = ny;
      }
      MPI_Bcast(buf.ddata,5,MPI_DOUBLE,0,MPI_COMM_WORLD);
      if (mp>0) {
        xa = buf.ddata[0]; xb = buf.ddata[1];
        ya = buf.ddata[2]; yb = buf.ddata[3];
        nx = buf.idata[8]; ny = buf.idata[9];
      }
    }
  }
  if (np==1) { np1 = 1; np2 = 1; }
  else {
    s = sqrt(((double)np)) * ((double)nx) / ((double)ny);
    np1 = floor(s); if (s>0.5+((double)np1)) np1++;
    np2 = np / np1;
    if (np1*np2!=np) {
      if (nx>ny) {np1 = np; np2 = 1;} else {np1 = 1; np2 = np;}
    }
  }
  mp2 = mp / np1; mp1 = mp % np1;
  if (mp==0) fprintf(stderr,"Grid=%dx%d\n",np1,np2);
  fprintf(Fo,"Netsize: %d, process: %d, system: %s, tick=%12le\n",np,mp,pname,t0);
  fprintf(Fo,"Grid=%dx%d coord=(%d,%d)\n",np1,np2,mp1,mp2);
  fprintf(Fo,"xa=%le xb=%le ya=%le yb=%le nx=%d ny=%d fl=%d\n",xa,xb,ya,yb,nx,ny,fl);
  t1 = MPI_Wtime();
  hx = (xb-xa)/nx; hy = (yb-ya)/ny; hxy = hx*hy;
  if (np1==1) { i1 = 0; i2 = nx-1; }
  else        { i1 = mp1 * (nx/np1); if (mp1<np1-1) i2 = i1+(nx/np1)-1; else i2 = nx-1; }
  if (np2==1) { j1 = 0; j2 = ny-1; }
  else        { j1 = mp2 * (ny/np2); if (mp2<np2-1) j2 = j1+(ny/np2)-1; else j2 = ny-1; }
  s = 0;
  for (j=j1; j<=j2; j++) {
    y = ya + (j*1.0+.5)*hy;
    for (i=i1; i<=i2; i++) {
      x = xa + (i*1.0+.5)*hx; s = s + hxy * f(x,y);
    }
  }
  t2 = MPI_Wtime(); t1 = t2 - t1;
  if (np==1)
    t2 = 0;
  else {
    p = s;
    MPI_Reduce(&p, &s, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
    t2 = MPI_Wtime()-t2;
  }
  t3 = t1 + t2;
  if (mp==0) fprintf(stderr,"t1=%le t2=%le t3=%le int=%le\n",t1,t2,t3,s);
  fprintf(Fo,"i1=%d i2=%d j1=%d j2=%d\n",i1,i2,j1,j2);
  fprintf(Fo,"t1=%le t2=%le t3=%le int=%le\n",t1,t2,t3,s);
  ier = fclose_m(&Fo);
  MPI_Finalize();
  return 0;
}
```

Трансляция:

```
>mpicc -o ex08b.px -O2 ex08b.c mycom.c mynet.c -lm
```

Результаты запуска:

```
>mpirun -np 1 -nolocal -machinefile hosts ex08b.px (запуск в классе)
>mpirun -np 1 ex08b.px                             (запуск на сервере)
Grid=1x1
t1=8.531479e+00 t2=0.000000e+00 t3=8.531479e+00 int=2.635526e+02
>mpirun -np 12 -nolocal -machinefile hosts ex08b.px (запуск в классе)
>mpirun -np 12 ex08b.px                             (запуск на сервере)
Grid=3x4
t1=7.514590e-01 t2=3.230000e-04 t3=7.517820e-01 int=2.635526e+02
```