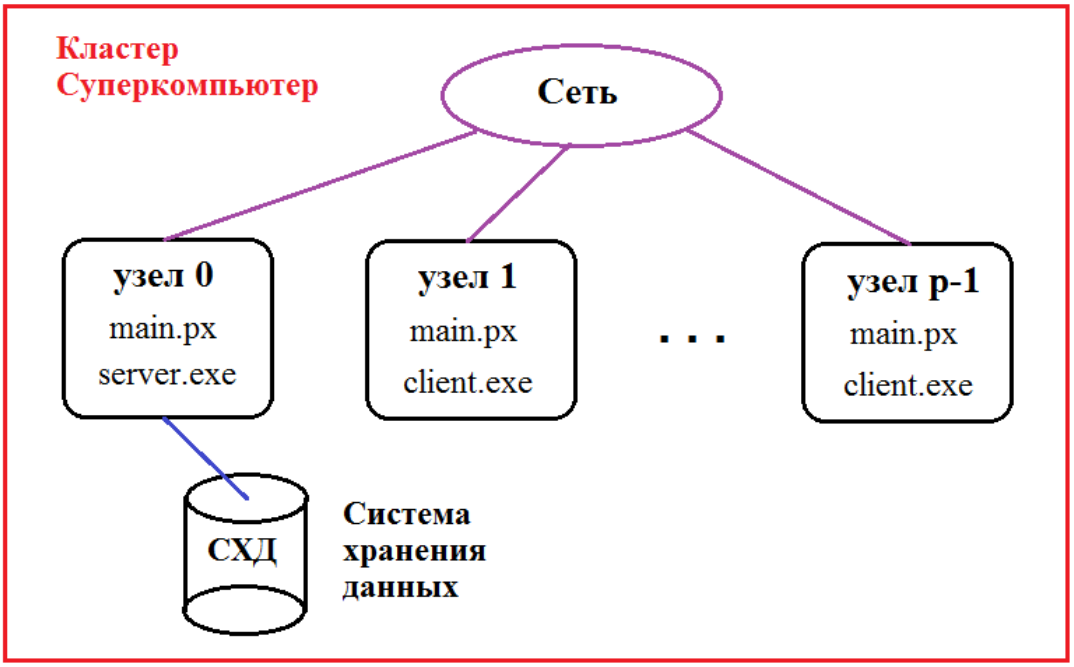
Семинар 4. Удаленные процессы и обмен данными между ними.
Стандарт MPI. Базовые функции.

1. Создание удаленных параллельных процессов и обмен данными между ними.

1.1. Понятие сетевого приложения:



1.2. Создание удаленных процессов

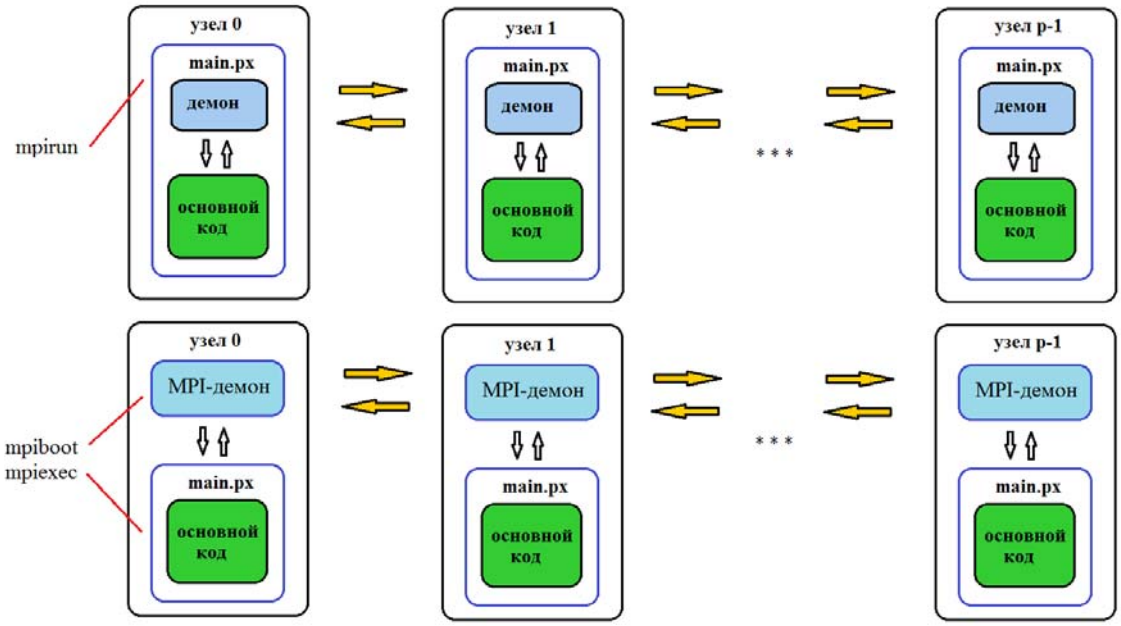
1) запуск программ на различных машинах (telnet, rlogin, rsh, ssh);
2) использование интерфейсов удаленного запуска: PVM, MPI.

1.3. Обмен данными

1) сокеты TCP/IP для межмашинной передачи данных;
2) библиотеки межмашинных коммуникаций: PVM, MPI.

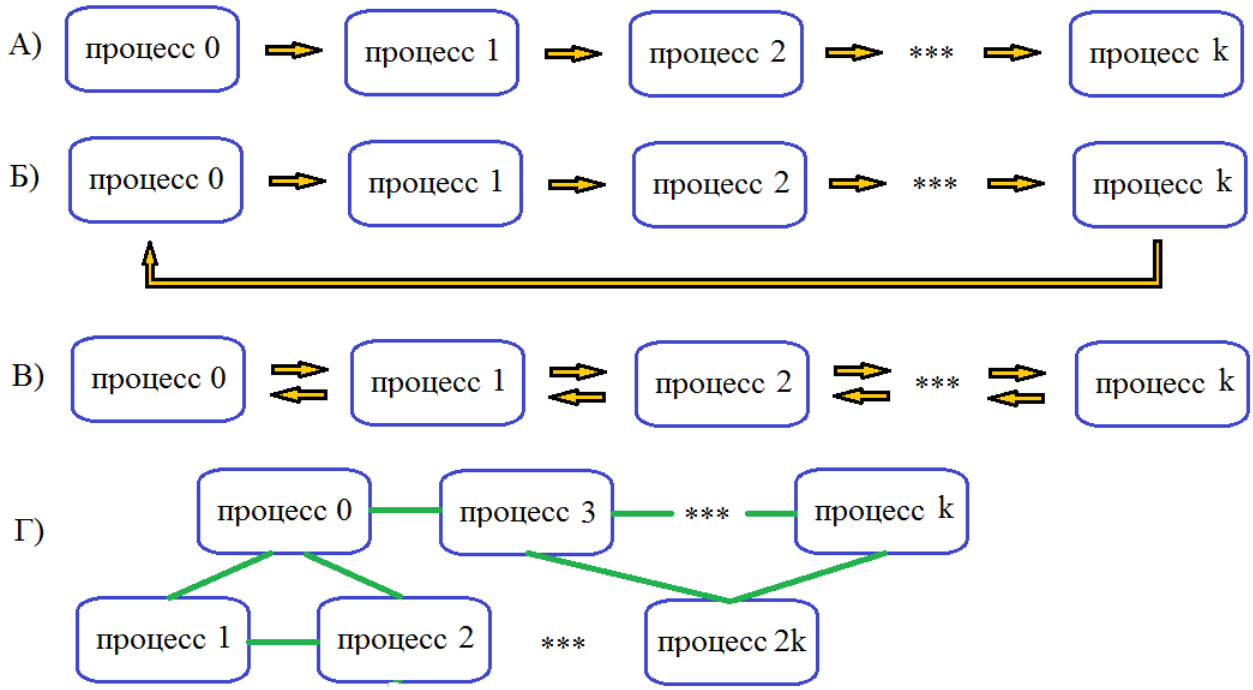
2. Стандарт MPI.

MPI (Message Passing Interface) – стандарт передачи сообщений по сети.
MPI – коммуникационная среда, библиотека функций для передачи данных, стандарт параллельного программирования.
1) Конфигурирование сервера (сервер и клиенты, доверенные хосты);
2) Инсталляция MPI на сервере (www.mpi-forum.org);
3) Разработка MPI-приложений;
4) Запуск MPI-приложений (mpirun; mpiboot, mpiexec).



3. Базовые функции MPI.

1) Инициализация и параметры среды – MPI_Init, MPI_Initialized, MPI_Comm_size, MPI_Comm_rank, MPI_Get_processor_name, MPI_Finalize, MPI_Abort, MPI_Wtick, MPI_Wtime;
2) Синхронные обмены типа точка-точка (синхронные и асинхронные) – MPI_Send, MPI_Recv, MPI_Sendrecv;
3) Процедуры синхронизации – MPI_Barrier.
4) Топологии обменов – структуры графа обменов данными.
Линейная (А), кольцевая (Б), двухсторонняя линейная (В), древовидная (Г), и др.



4. Реализация примеров.

Пример 1 – простейшая программа на MPI (ex06a.c).

```
#include <stdio.h>
#include "mpi.h"
int main(int argc, char *argv[])
{
  int i, ii, np, mp, nl; char pname[MPI_MAX_PROCESSOR_NAME];
  int nc=10000; double t0, t1, a;
  MPI_Initialized(&ii); fprintf(stderr,"Before MPI_Init ii=%d\n",ii);
  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&np);
  MPI_Comm_rank(MPI_COMM_WORLD,&mp);
  MPI_Get_processor_name(pname,&nl);
  t0 = MPI_Wtick();
  MPI_Initialized(&ii); fprintf(stderr,"Between MPI_Init & MPI_Finalize ii=%d\n",ii);
  fprintf(stderr,"Netsize: %d, process: %d, system: %s, tick=%12le\n",np,mp,pname,t0);
  t1 = MPI_Wtime();
  a = 0; for (i=0; i<nc; i++) {a = a + 1.23*i; if (a>1000) a = a / 1000;}
  t1 = MPI_Wtime()-t1;
  fprintf(stderr,"mp=%d, time=%12le res=%12le\n",mp,t1,a);
  MPI_Finalize(); MPI_Initialized(&ii); fprintf(stderr,"After MPI_Finalize ii=%d\n",ii);
  return 0;
}
```

Трансляция:

```
>mpicc -o ex06a.px -O2 ex06a.c -lm
```

Запуск и результаты исполнения:

```
>mpirun -np 1 -nolocal -machinefile hosts ex06a.px (запуск в классе)
>mpirun -np 1 ex06a.px                             (запуск на сервере)
Before MPI_Init ii=0
Between MPI_Init & MPI_Finalize ii=1
Netsize: 1, process: 0, system: cl73.limm, tick=1.000000e-06
```

```
mp=0, time=2.460000e-04 res=1.231108e+01
After MPI_Finalize ii=1

>mpirun -np 2 -nolocal -machinefile hosts ex06a.px (запуск в классе)
>mpirun -np 2 ex06a.px                             (запуск на сервере)
Before MPI_Init ii=0
Before MPI_Init ii=0
Between MPI_Init & MPI_Finalize ii=1
Netsize: 2, process: 0, system: cl73.limm, tick=1.000000e-06
Between MPI_Init & MPI_Finalize ii=1
Netsize: 2, process: 1, system: cl74.limm, tick=1.000000e-06
mp=0, time=2.450000e-04 res=1.231108e+01
mp=1, time=2.610000e-04 res=1.231108e+01
After MPI_Finalize ii=1
After MPI_Finalize ii=1
```

Пример 2 – Двунаправленный синхронный обмен с соседями в линейной топологии (ex06b.c).

```
#include <stdio.h>
#include <unistd.h>
#include "mpi.h"
int main(int argc, char *argv[])
{
  int i, np, mp, nl; char pname[MPI_MAX_PROCESSOR_NAME]; MPI_Status status;
  double t0, a1, a2, a3;
  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&np);
  MPI_Comm_rank(MPI_COMM_WORLD,&mp);
  MPI_Get_processor_name(pname,&nl);
  t0 = MPI_Wtick();
  fprintf(stderr,"Netsize: %d, process: %d, system: %s, tick=%12le\n",np,mp,pname,t0);
  a1 = -1; a2 = mp * 3.14; a3 = -1;
  if (np>1){
    sleep(1);
    if (mp%2==0){
      if (mp+1<np){
        MPI_Sendrecv(&a2,1,MPI_DOUBLE,mp+1,0,
                     &a3,1,MPI_DOUBLE,mp+1,0,
                     MPI_COMM_WORLD,&status);
        fprintf(stderr,"%02d <-> %02d\n",mp,mp+1);
      }
      if (mp>0){
        MPI_Sendrecv(&a2,1,MPI_DOUBLE,mp-1,0,
                     &a1,1,MPI_DOUBLE,mp-1,0,
                     MPI_COMM_WORLD,&status);
        fprintf(stderr,"%02d <-> %02d\n",mp,mp-1);
      }
    }
    if (mp%2==1){
      if (mp>0){
        MPI_Sendrecv(&a2,1,MPI_DOUBLE,mp-1,0,
                     &a1,1,MPI_DOUBLE,mp-1,0,
                     MPI_COMM_WORLD,&status);
        fprintf(stderr,"%02d <-> %02d\n",mp,mp-1);
      }
      if (mp+1<np){
        MPI_Sendrecv(&a2,1,MPI_DOUBLE,mp+1,0,
                     &a3,1,MPI_DOUBLE,mp+1,0,
                     MPI_COMM_WORLD,&status);
        fprintf(stderr,"%02d <-> %02d\n",mp,mp+1);
      }
    }
    sleep(1);
  }
  fprintf(stderr,"mp=%d a1=%le a2=%le a3=%le\n",mp,a1,a2,a3);
  MPI_Finalize();
  return 0;
}
```

Трансляция:
```
>mpicc -o ex06b.px -O2 ex06b.c -lm
```

Запуск и результаты исполнения:
```
>mpirun -np 1 -nolocal -machinefile hosts ex06b.px (запуск в классе)
>mpirun -np 1 ex06b.px                             (запуск на сервере)
Netsize: 1, process: 0, system: cl73.limm, tick=1.000000e-06
mp=0 a1=-1.000000e+00 a2=0.000000e+00 a3=-1.000000e+00

>mpirun -np 2 -nolocal -machinefile hosts ex06b.px (запуск в классе)
>mpirun -np 2 ex06b.px                             (запуск на сервере)
Netsize: 2, process: 0, system: cl73.limm, tick=1.000000e-06
Netsize: 2, process: 1, system: cl74.limm, tick=1.000000e-06
00 <-> 01
01 <-> 00
mp=0 a1=-1.000000e+00 a2=0.000000e+00 a3=3.140000e+00
mp=1 a1=0.000000e+00 a2=3.140000e+00 a3=-1.000000e+00

>mpirun -np 3 -nolocal -machinefile hosts ex06b.px (запуск в классе)
>mpirun -np 3 ex06b.px                             (запуск на сервере)
Netsize: 3, process: 0, system: cl73.limm, tick=1.000000e-06
Netsize: 3, process: 1, system: cl74.limm, tick=1.000000e-06
Netsize: 3, process: 2, system: cl78.limm, tick=1.000000e-06
00 <-> 01
01 <-> 00
01 <-> 02
02 <-> 01
mp=0 a1=-1.000000e+00 a2=0.000000e+00 a3=3.140000e+00
mp=1 a1=0.000000e+00 a2=3.140000e+00 a3=6.280000e+00
mp=2 a1=3.140000e+00 a2=6.280000e+00 a3=-1.000000e+00

>mpirun -np 4 -nolocal -machinefile hosts ex06b.px (запуск в классе)
>mpirun -np 4 ex06b.px                             (запуск на сервере)
00 <-> 01
01 <-> 00
02 <-> 03
01 <-> 02
03 <-> 02
02 <-> 01
mp=2 a1=3.140000e+00 a2=6.280000e+00 a3=9.420000e+00
mp=0 a1=-1.000000e+00 a2=0.000000e+00 a3=3.140000e+00
mp=1 a1=0.000000e+00 a2=3.140000e+00 a3=6.280000e+00
mp=3 a1=6.280000e+00 a2=9.420000e+00 a3=-1.000000e+00
```

Пример 3 – Однонаправленный синхронный обмен в топологии кольцо (ex06c.c).
```
#include <stdio.h> #include <unistd.h> #include "mpi.h"
#define MY_TAG 333
int main(int argc, char *argv[])
{
  int np, mp, nl; char pname[MPI_MAX_PROCESSOR_NAME]; MPI_Status status;
  double t0, t1, a;
  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&np);
  MPI_Comm_rank(MPI_COMM_WORLD,&mp);
  MPI_Get_processor_name(pname,&nl);
  t0 = MPI_Wtick();
  fprintf(stderr,"Netsize: %d, process: %d, system: %s, tick=%12le\n",np,mp,pname,t0);
  if (np<2) { fprintf(stderr,"Too small network\n"); MPI_Finalize(); return 0; }
  sleep(1); t1 = MPI_Wtime(); a = 0;
  if (mp==0) { a = a + 1.0;
    MPI_Send(&a, 1, MPI_DOUBLE, mp+1, MY_TAG, MPI_COMM_WORLD);
    MPI_Recv(&a, 1, MPI_DOUBLE, np-1, MY_TAG, MPI_COMM_WORLD, &status);
  }
  else {
    MPI_Recv(&a, 1, MPI_DOUBLE, mp-1, MY_TAG, MPI_COMM_WORLD, &status);
    a = a + 1.0;
    MPI_Send(&a, 1, MPI_DOUBLE, (mp+1) % np, MY_TAG, MPI_COMM_WORLD);
  }
```

```
  t1 = MPI_Wtime()-t1; sleep(1);
  fprintf(stderr,"mp=%d time=%12le res=%12le\n",mp,t1,a);
  MPI_Finalize(); return 0;
}
```

Трансляция: >mpicc -o ex06c.px -O2 ex06c.c -lm

Запуск и результаты исполнения:

```
>mpirun -np 1 -nolocal -machinefile hosts ex06c.px (запуск в классе)
>mpirun -np 1 ex06c.px                             (запуск на сервере)
Too small network

>mpirun -np 2 -nolocal -machinefile hosts ex06c.px (запуск в классе)
>mpirun -np 2 ex06c.px                             (запуск на сервере)
mp=0 time=3.210000e-04 res=2.000000e+00
mp=1 time=2.170000e-04 res=2.000000e+00

>mpirun -np 3 -nolocal -machinefile hosts ex06c.px (запуск в классе)
>mpirun -np 3 ex06c.px                             (запуск на сервере)
mp=2 time=1.213900e-02 res=3.000000e+00
mp=1 time=5.149000e-03 res=2.000000e+00
mp=0 time=4.609000e-03 res=3.000000e+00

>mpirun -np 4 -nolocal -machinefile hosts ex06c.px (запуск в классе)
>mpirun -np 4 ex06c.px                             (запуск на сервере)
mp=1 time=9.090000e-04 res=2.000000e+00
mp=2 time=9.127000e-03 res=3.000000e+00
mp=0 time=7.405000e-03 res=4.000000e+00
mp=3 time=6.542000e-03 res=4.000000e+00
```

Пример 4 – Вычисление интеграла. Сборка суммы на нулевом процессоре (ex07a.c).

```
#include <stdio.h> #include <math.h> #include "mycom.h" #include "mynet.h"
int np, mp, nl; char pname[MPI_MAX_PROCESSOR_NAME];
MPI_Status status; static double tick, t1, t2, t3;
double a = 0; static double b = 1; int ni = 1000000000; double sum = 0;
double f1(double x); double f1(double x) { return 4.0/(1.0+x*x);}
double myjob(int mp); double myjob(int mp)
{
  int n1; double a1, b1, h1, s1;
  h1 = (b - a) / np; n1 = ni / np;
  a1 = a + h1 * mp; if (mp<np-1) b1 = a1 + h1; else b1 = b;
  s1 = integrate(f1,a1,b1,n1);
  return s1;
}
int main(int argc, char *argv[])
{
  MyNetInit(&argc,&argv,&np,&mp,&nl,pname,&tick);
  if (np<2) {
    t1 = MPI_Wtime(); sum = integrate(f1,a,b,ni); t2 = MPI_Wtime(); t3 = t2;
  }
  else {
    int i; double p;
    t1 = MPI_Wtime(); sum = myjob(mp); t2 = MPI_Wtime();
    if (mp==0)
      for (i=1; i<np; i++) {
        MPI_Recv(&p, 1, MPI_DOUBLE, i, MY_TAG, MPI_COMM_WORLD, &status);
        sum = sum + p;
      }
    else
      MPI_Send(&sum, 1, MPI_DOUBLE, 0, MY_TAG, MPI_COMM_WORLD);
    MPI_Barrier(MPI_COMM_WORLD);
    t3 = MPI_Wtime();
  }
  t1 = t2 - t1; t2 = t3 - t2; t3 = t1 + t2;
  fprintf(stderr,"mp=%d t1=%lf t2=%lf t3=%lf int=%22.15le\n",mp,t1,t2,t3,sum);
  MPI_Finalize();
  return 0;
}
```

Трансляция:

```
>mpicc -o ex07a.px -O2 ex07a.c mycom.c mynet.c -lm
```

Запуск и результаты исполнения:

```
>mpirun -np 1 -nolocal -machinefile hosts ex07a.px (запуск в классе)
>mpirun -np 1 ex07a.px                             (запуск на сервере)
mp=0 t1=17.624254 t2=0.000000 t3=17.624254 int= 3.141592651591870e+00

>mpirun -np 2 -nolocal -machinefile hosts ex07a.px (запуск в классе)
>mpirun -np 2 ex07a.px                             (запуск на сервере)
mp=0 t1=8.713395 t2=5.357119 t3=14.070514 int= 3.141592648390306e+00
mp=1 t1=14.068855 t2=0.000404 t3=14.069259 int= 1.287002215586667e+00

>mpirun -np 4 -nolocal -machinefile hosts ex07a.px (запуск в классе)
>mpirun -np 4 ex07a.px                             (запуск на сервере)
mp=0 t1=4.358262 t2=0.028721 t3=4.386983 int= 3.141592642065422e+00
mp=1 t1=4.366067 t2=0.020719 t3=4.386786 int= 8.746757802958527e-01
mp=2 t1=4.360264 t2=0.025230 t3=4.385494 int= 7.194139966100185e-01
mp=3 t1=4.384680 t2=0.000514 t3=4.385194 int= 5.675882164165912e-01

>mpirun -np 8 -nolocal -machinefile hosts ex07a.px (запуск в классе)
>mpirun -np 8 ex07a.px                             (запуск на сервере)
mp=0 t1=2.179436 t2=1.423302 t3=3.602738 int= 3.141592629478097e+00
mp=2 t1=2.194378 t2=1.408253 t3=3.602631 int= 4.551680250680343e-01
mp=4 t1=3.600436 t2=0.000350 t3=3.600786 int= 3.798068224947328e-01
mp=3 t1=2.192403 t2=1.410172 t3=3.602575 int= 4.195077517209749e-01
mp=6 t1=2.190496 t2=1.410177 t3=3.600673 int= 3.013155610478502e-01
mp=7 t1=2.200823 t2=1.399654 t3=3.600477 int= 2.662726531032736e-01
mp=1 t1=2.182648 t2=1.424412 t3=3.607060 int= 4.824946705556977e-01
mp=5 t1=2.187454 t2=1.418299 t3=3.605753 int= 3.396071712388637e-01
```