Знакомство с MPI

Практическое занятие #1

ВВЕДЕНИЕ В MPI

Программы на MPI

- Программа состоит из N параллельных процессов
- Процессы порождаются один раз при запуске программы
- Каждый процесс имеет отдельное адресное пространство, общей памяти нет
- Процессы взаимодействуют путем отправки и получения сообщений
- Процессы могут образовывать группы

Общая схема MPI программы

```
#include<mpi.h>
int main(int argc,char **argv){
    int rank, size;

    //Инициализация работы с MPI
    MPI_Init(&argc, &argv);

    //Параллельная часть

    //Завершение работы с MPI
    MPI_Finalize();

    return 0;
}
```

Hello, world!

```
#include<mpi.h>
#include<stdio.h>

int main(int argc, char**argv) {
    int rank,size,len;
    char host[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Get_processor_name(host, &len);
    printf("Hello, world. I am %d of %d on %s\n",
                            rank, size, host);
    MPI_Finalize();
    return 0;
}
```

Hello, world!

```
Hello, world. I am 0 of 4 on node22
Hello, world. I am 1 of 4 on node23
Hello, world. I am 2 of 4 on node21
Hello, world. I am 3 of 4 on node24
```

СООБЩЕНИЯ В MPI

Сообщения в MPI

- Передача данных между процессами осуществляется путем отправки и приема сообщений
- Требует совместная работа отправляющего и принимающего процесса
- Изменение памяти принимающего процесса происходит при его явном участии

Сообщения в MPI

```
/* Связи между частями модели. Посылка данных */
if (rank == 0) {

    // Программа процессора #0
    int data_send[L];
    // Отправить массив data_send на процессор #7
    MPI_Send(data_senf, L, MPI_INT, 7 , 1, MPI_COMM_WORLD);

} else if (rank == 7) {

    // Программа процессора #7
    int data_resv[L];     MPI_Status status;
    // Принять массив длины L с любого процессора в data_resv
    MPI_Recv(data_resv, L, MPI_INT,
             MPI_ANY_SOURCE, 1, MPI_COMM_WORLD, &status);
}
```

Типы данных

Константа MPI	Тип данных
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED_LONG	unsigned long_int
MPI_UNSIGNED	unsigned int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	Байт
MPI_PACKED	Упакованные данные

Способы отправки данных

- **Стандартные прием/передача (MPI_Send)**
  Блокирует работу. Возврат после окончания передачи.
  Абсолютно безопасная передача данных

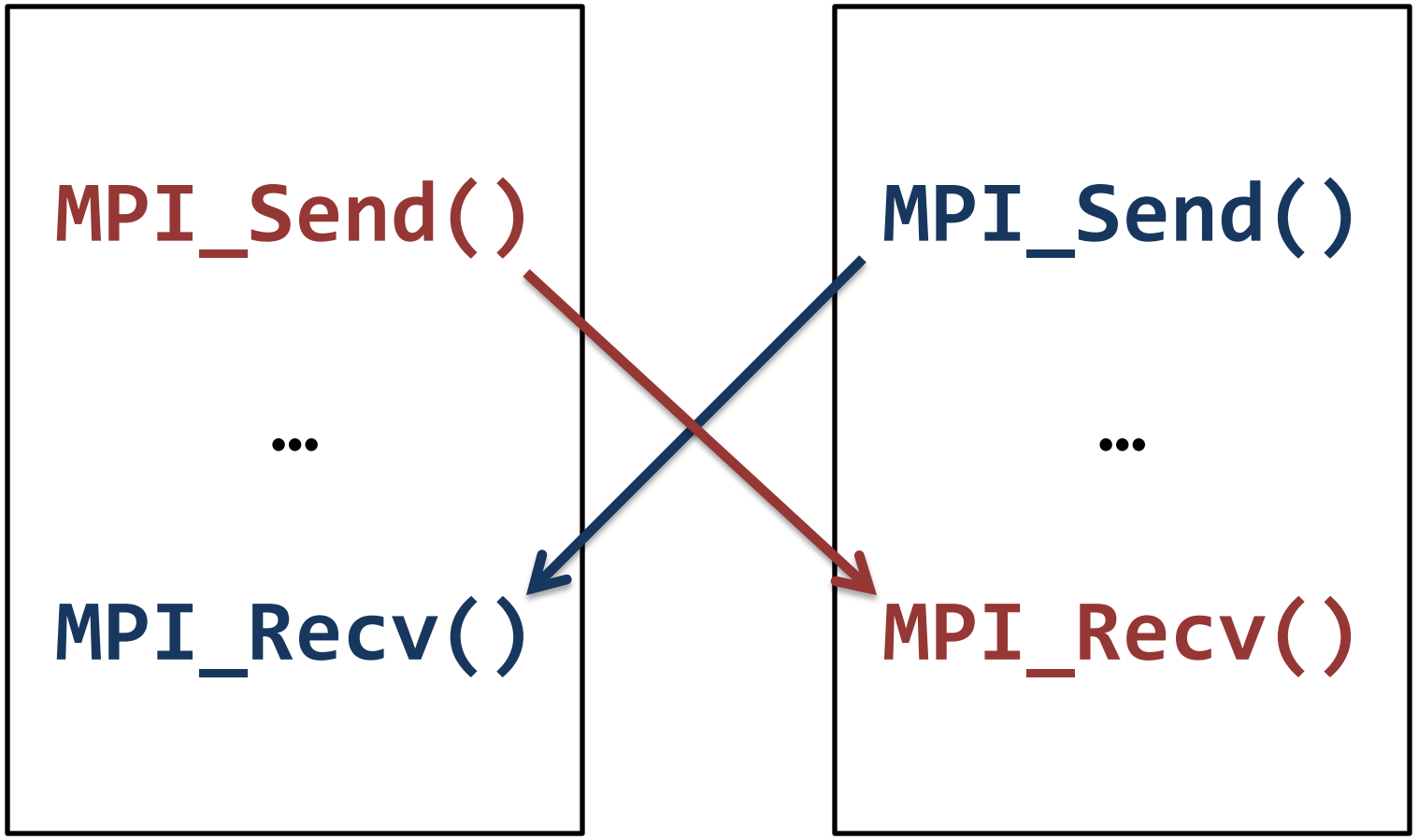
- **Передача с буферизацией (MPI_Bsend)**
  Возврат после записи сообщения в системный буфер
  Не требует вызова функции приема сообщения

- **Передача с синхронизацией (MPI_Ssend)**
  Возврат после начала приема сообщения процессом-получателем
  Гарантирует безопасное использование буфера

- **Передача по готовности (MPI_Rsend)**
  Процесс-получатель должен инициировать прием сообщения
  Уменьшает накладные расходы на организацию передачи

Зачем отправлять не стандартно?

Пример блокировки



MPI ДОМА

SOFT

- Реализация MPI: библиотека MPICH2
  http://www.mcs.anl.gov/research/projects/mpich2/
- Мануал по MPICH2
  http://iproc.ru/programming/mpich-windows/
- Компиляторы
  - Visual studio
  - MinGW
    http://www.mingw.org/

MPICH2

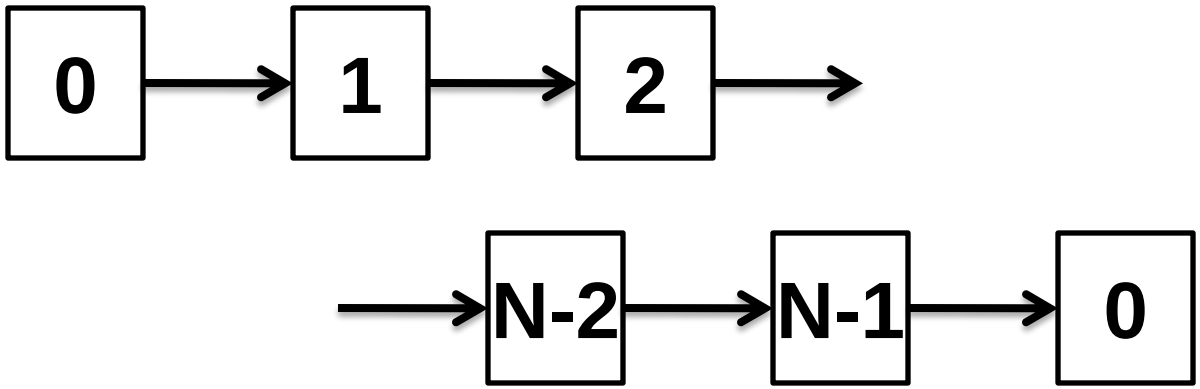
mpiexec.exe -n 4 –localroot cpi.exe

```
Enter the number of intervals: (0 quits) 1
pi is approximately 3.2000000000000002, Error is
  0.0584073464102071
wall clock time = 0.000219
Enter the number of intervals: (0 quits) 2
pi is approximately 3.1623529411764704, Error is
  0.0207602875866773
wall clock time = 0.000300
Enter the number of intervals: (0 quits) 3
pi is approximately 3.1508492098656031, Error is
  0.0092565562758100
wall clock time = 0.000133
Enter the number of intervals: (0 quits) 0
```

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Задание на освоение

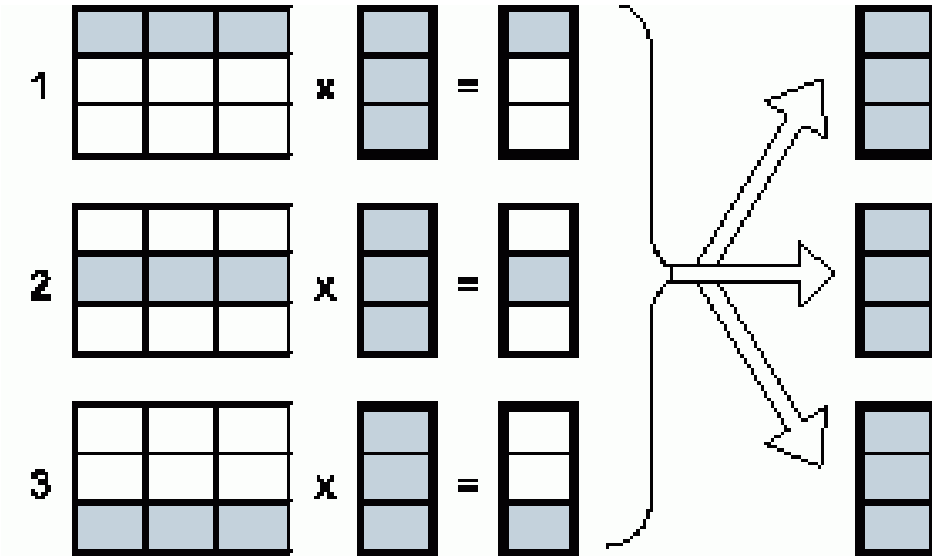
Написать программу передачи целого числа по процессорам, находящихся в топологии кольца.



Отчетное задание. Вариант 1 (до 13 ноября)

Реализовать параллельное умножение матриц.

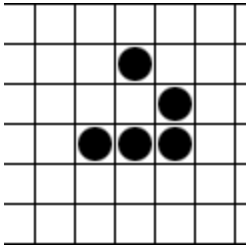
Схема умножения матрицы на вектор:



Отчетное задание. Вариант 2 (до 13 ноября)

Реализовать программу для игры Жизнь.

http://ru.wikipedia.org/wiki/Жизнь_(игра)



В программе должен использоваться обмен приграничными полосами.