Параллельные алгоритмы: MPI. Введение

Н. И. Хохлов

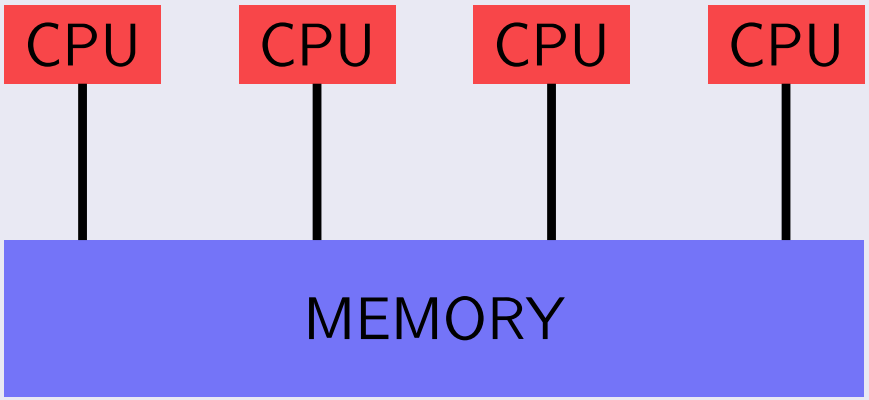
МФТИ, Долгопрудный

3 февраля 2021 г.

Системы с распределенной и общей памятью

Системы с распределенной и общей памятью

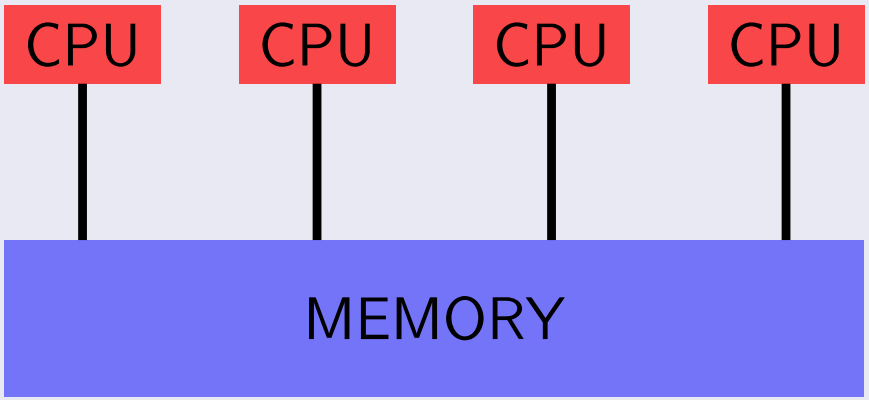
Общая память (shared memory)



Пример – современные много-ядерные/многопроцессорные машины.

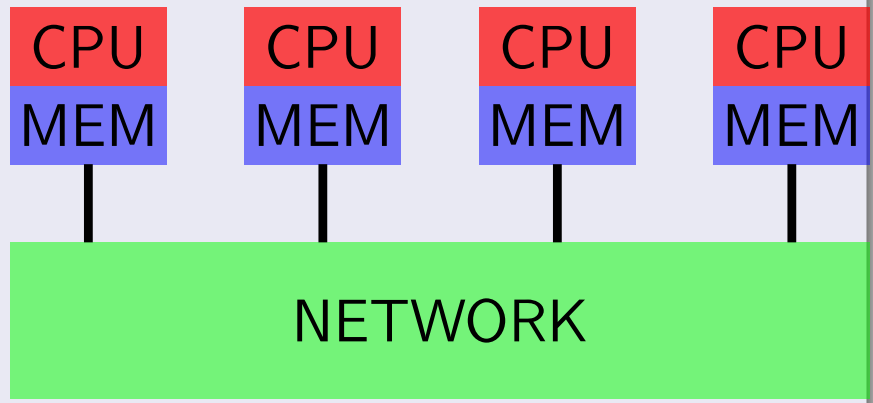
Системы с распределенной и общей памятью

Общая память (shared memory)



Пример – современные много-ядерные/многопроцессорные машины.

Распределенная память (distributed memory)



Пример – несколько одноядерных машин объеденeнных в сеть.

Большинство современных архитектур – гибридные, т. е. многоядерные машины, объеденeнные в одну общую сеть.

Что такое MPI?

- MPI = Message Passing Interface – Интерфейс для передачи сообщений.
- MPI – не библиотека, а спецификация (стандарт) для программистов и пользователей. На основе спецификации может быть написана библиотека.
- Основная цель MPI – предоставить широко используемый стандарт для написания параллельных приложений построенных на передаче сообщений.
- Интерфейс есть для языков C и Fortran. Некоторые версии стандарта также поддерживают C++.

История появления стандарта MPI

- 1980-1990 гг. – появление суперкомпьютеров с разделяемой памятью.
- 1992-1994 гг. – множество технологий для написания приложений в системах с разделяемой памятью, начало зарождения MPI.
- Апрель 1992 г. – начата работа над спецификацией MPI, были обсуждены основные идеи и функциональность. Далее шла работа над спецификацией (Center for Research on Parallel Computing, Williamsburg, Virginia).
- Ноябрь 1992 г. – встреча в Минеаполе (Minneapolis). Черновой вариант MPI. Создание MPI Forum (MPIF) – туда входит около 175 членов из 40 организация занимающихся параллельными вычислениями, программный обеспечением а также академические и научные организации.

История появления стандарта MPI

- Ноябрь 1993 г. – на конференции Supercomputing 93 были доложены стандарты MPI.
- Май 1994 г. – финальная версия стандарта MPI – http://www-unix.mcs.anl.gov/mpi.
- 1996 г. – появление спецификации MPI-2, прежняя спецификация получила название MPI-1.
- Сентябрь 2012 г. – стандарт MPI-3.

Современные реализации MPI включают в себя MPI-1, MPI-2, MPI-3, в зависимости от реализации возможна поддержка либо только части стандартов или все вместе.

Почему надо использовать MPI?

- **Стандарт** – единственный стандарт HPC на текущий момент.
- **Переносимость кода** – нет необходимости менять код при использовании различных платформ.
- **Производительность** – производители железа и софта сами заботятся о скорости работы библиотек.
- **Функциональность** – только стандарт MPI-1.1 предоставляет более 115 функций.
- **Доступность** – множество свободных реализаций.
- **Простота отладки** – в отличии от приложений на системах с общей памятью, каждый MPI процесс работает однопоточно (в рамках MPI).

Основной функционал MPI-1

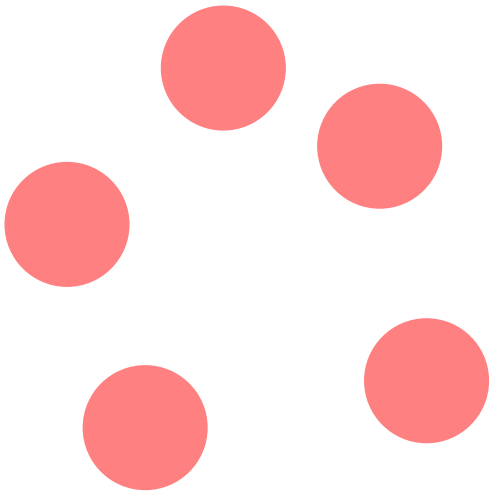
- Набор утильных функций, инициализация/завершение работы MPI.
- Взаимодействия типа точка-точка (p-t-p).
- Коллективные взаимодействия (collective).
- Типы данных MPI (datatypes).
- Функции для работы с группами процессов и создания виртуальных топологий взаимодействия процессов.

Модель программирования MPI

- Дает виртуальный интерфейс ко всем моделям программирования с распределенной памятью.
- Железо:
  - Компьютеры с распределенной памятью – изначально разрабатывалась для них.
  - Общая память – дает виртуальную распределенную память.
  - Гибридные – современные версии MPI дают большие возможности для работы на гибридных архитектурах, в том числе с наличием GPU процессоров.
- Явный параллелизм.
- Число процессов статично. Нельзя породить новый процесс во время работы программы (MPI-2 и далее обходит это ограничение).

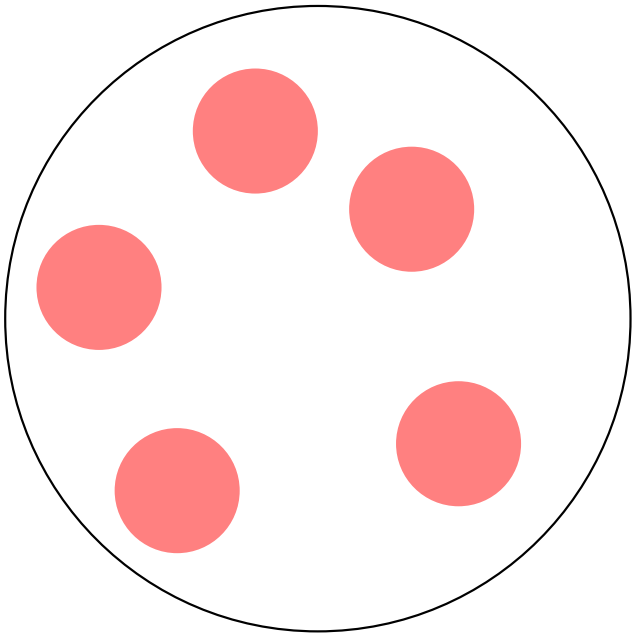
Коммуникаторы и группы

Коммуникаторы и группы



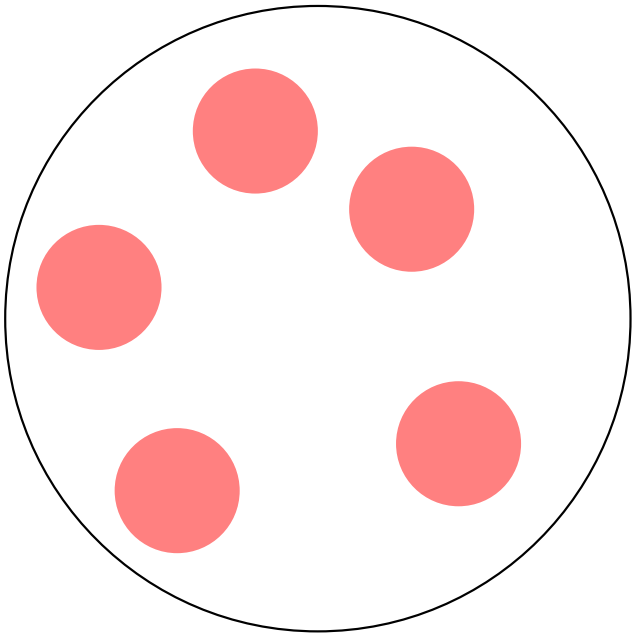
- Рассмотрим несколько процессов.

Коммуникаторы и группы



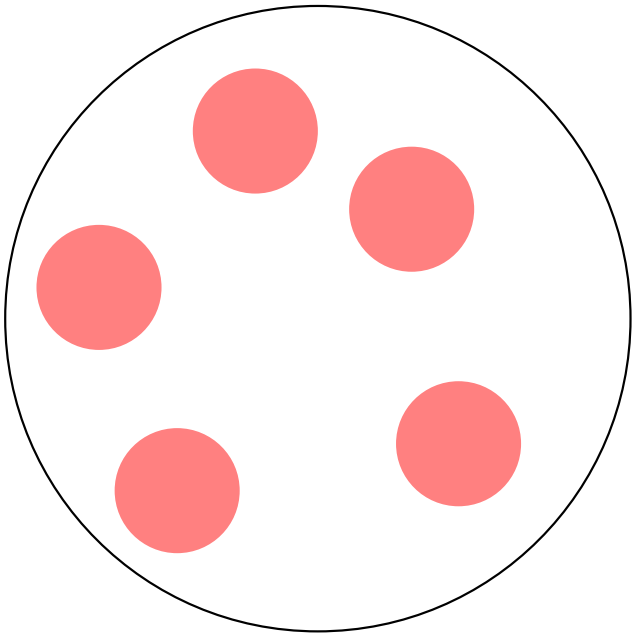
- Рассмотрим несколько процессов.
- Множество некоторых процессов называется группой.

Коммуникаторы и группы



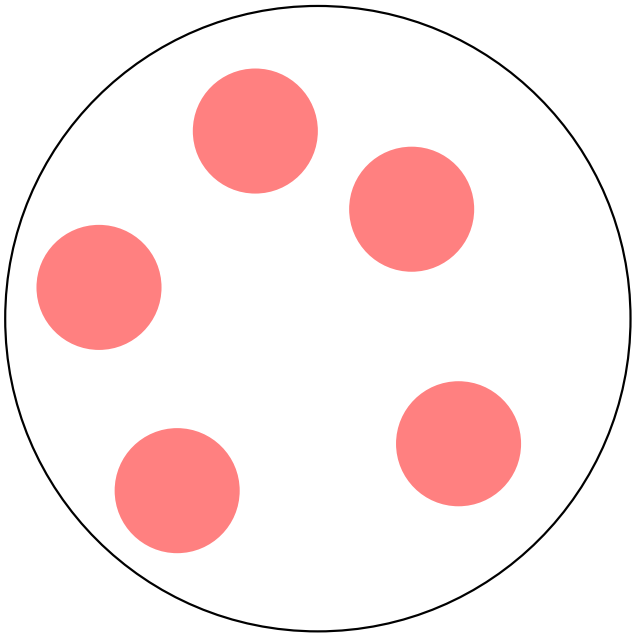
- Рассмотрим несколько процессов.
- Множество некоторых процессов называется группой.
- С группой может быть связан специальный объект – коммуникатор. Коммуникатор определяет какие процессы могут взаимодействовать между собой.

Коммуникаторы и группы



- Рассмотрим несколько процессов.
- Множество некоторых процессов называется группой.
- С группой может быть связан специальный объект – коммуникатор. Коммуникатор определяет какие процессы могут взаимодействовать между собой.
- Большинство функций MPI требуют в качестве параметра коммуникатор.

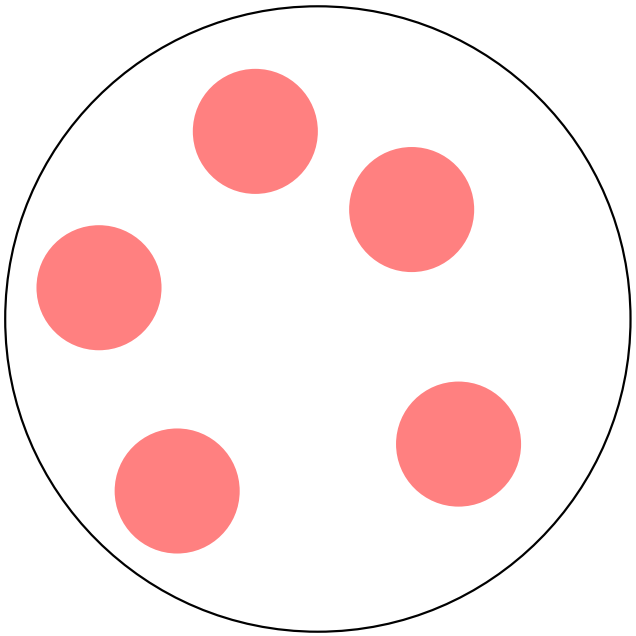
Коммуникаторы и группы



- Рассмотрим несколько процессов.
- Множество некоторых процессов называется группой.
- С группой может быть связан специальный объект – коммуникатор. Коммуникатор определяет какие процессы могут взаимодействовать между собой.
- Большинство функций MPI требуют в качестве параметра коммуникатор.
- MPI_COMM_WORLD – глобальный коммуникатор, включающий в себя все процессы.

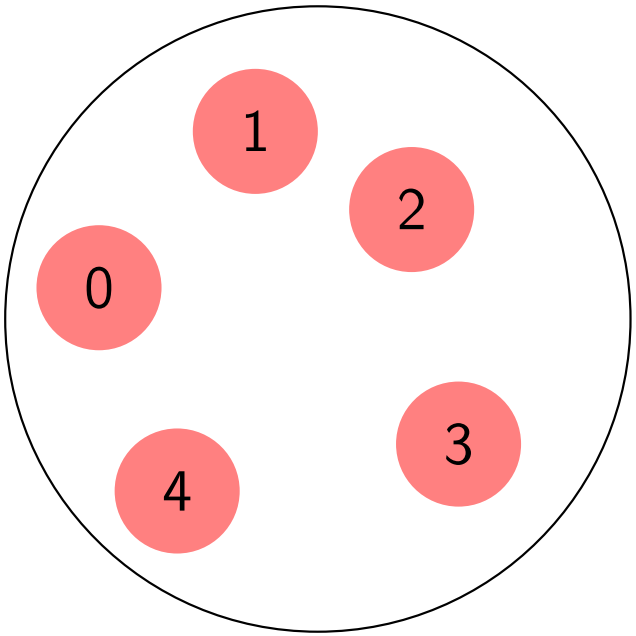
Нумерация процессов

Нумерация процессов



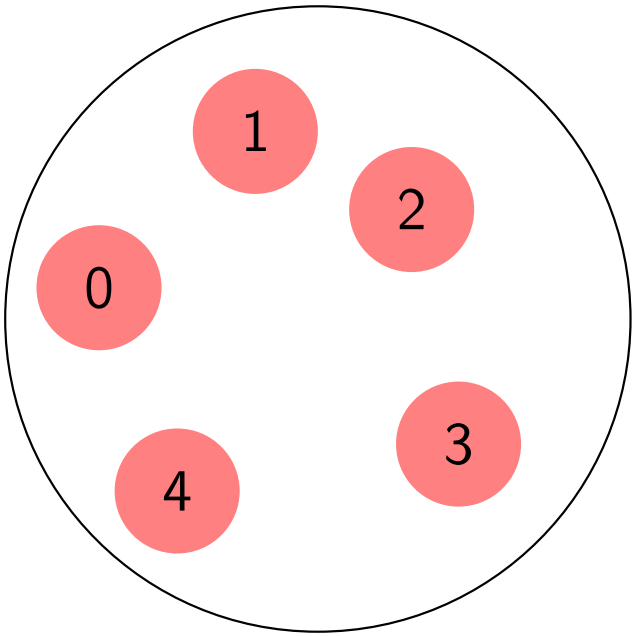
- Рассмотрим несколько процессов в коммуникаторе.

Нумерация процессов



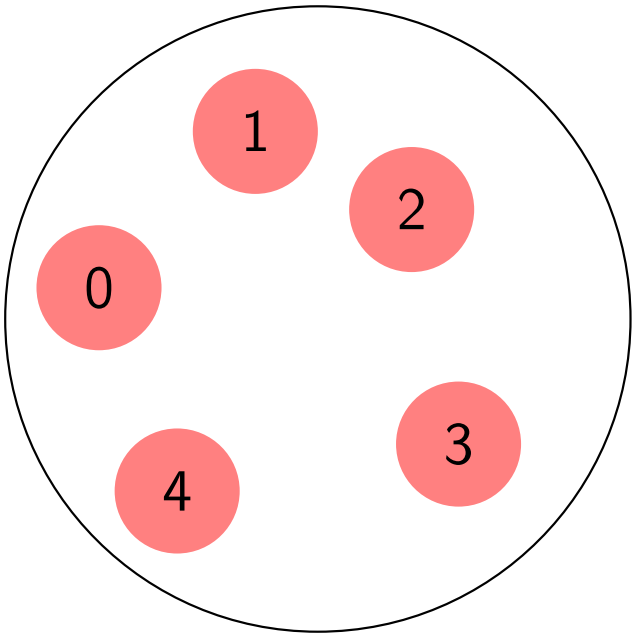
- Рассмотрим несколько процессов в коммуникаторе.
- Внутри коммуникатора каждый процесс имеет уникальный нормер (rank, task id).

Нумерация процессов



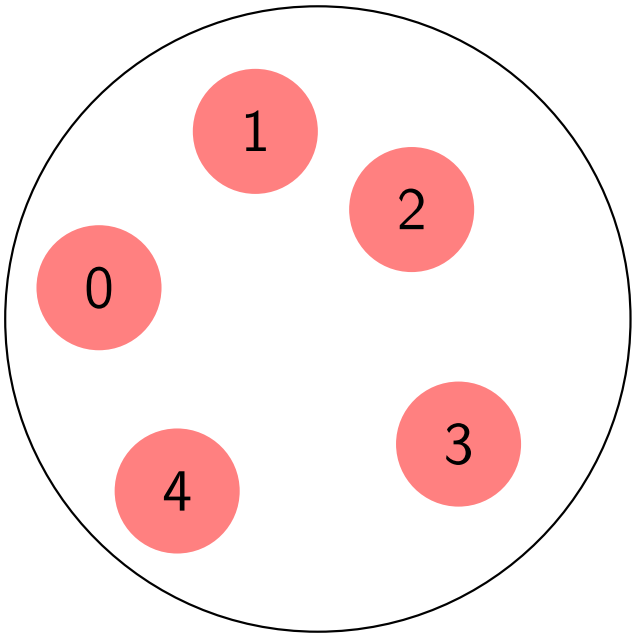
- Рассмотрим несколько процессов в коммуникаторе.
- Внутри коммуникатора каждый процесс имеет уникальный нормер (rank, task id).
- Номер представляет собой целое, неотрицательное число. Номера идут последовательно от 0 с шагом 1.

Нумерация процессов



- Рассмотрим несколько процессов в коммуникаторе.
- Внутри коммуникатора каждый процесс имеет уникальный нормер (rank, task id).
- Номер представляет собой целое, неотрицательное число. Номера идут последовательно от 0 с шагом 1.
- Используются в качестве адресатов при передаче сообщений.

Нумерация процессов



- Рассмотрим несколько процессов в коммуникаторе.
- Внутри коммуникатора каждый процесс имеет уникальный нормер (rank, task id).
- Номер представляет собой целое, неотрицательное число. Номера идут последовательно от 0 с шагом 1.
- Используются в качестве адресатов при передаче сообщений.
- Удобно использовать для контроля выполнения программы на разных процессах (if (rank == 0)).

Пример "Hello, world!"

```
#include <stdio.h>


int main(int argc, char *argv[])
{




    printf("Hello, world!\n");

    return 0;
}
```

Пример "Hello, world!"

```
#include <stdio.h>
#include <mpi.h> // Заголовочный файл MPI.

int main(int argc, char *argv[])
{




    printf("Hello, world!\n");

    return 0;
}
```

Пример "Hello, world!"

```
#include <stdio.h>
#include <mpi.h> // Заголовочный файл MPI.

int main(int argc, char *argv[])
{

    MPI_Init(&argc, &argv); // Инициализация MPI.


    printf("Hello, world!\n");

    return 0;
}
```

Пример "Hello, world!"

```
#include <stdio.h>
#include <mpi.h> // Заголовочный файл MPI.

int main(int argc, char *argv[])
{

    MPI_Init(&argc, &argv); // Инициализация MPI.


    printf("Hello, world!\n");
    MPI_Finalize(); // Завершение работы с MPI.
    return 0;
}
```

Пример "Hello, world!"

```
#include <stdio.h>
#include <mpi.h> // Заголовочный файл MPI.

int main(int argc, char *argv[])
{
    int  numtasks, rank; // Номер и число процессов.
    MPI_Init(&argc, &argv); // Инициализация MPI.


    printf("Hello, world!\n");
    MPI_Finalize(); // Завершение работы с MPI.
    return 0;
}
```

Пример "Hello, world!"

```
#include <stdio.h>
#include <mpi.h> // Заголовочный файл MPI.

int main(int argc, char *argv[])
{
    int  numtasks, rank; // Номер и число процессов.
    MPI_Init(&argc, &argv); // Инициализация MPI.
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks); // Число потоков.

    printf("Hello, world!\n");
    MPI_Finalize(); // Завершение работы с MPI.
    return 0;
}
```

Пример "Hello, world!"

```
#include <stdio.h>
#include <mpi.h> // Заголовочный файл MPI.

int main(int argc, char *argv[])
{
    int  numtasks, rank; // Номер и число процессов.
    MPI_Init(&argc, &argv); // Инициализация MPI.
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks); // Число потоков.
    MPI_Comm_rank(MPI_COMM_WORLD, &rank); // Номер текущего пот
    printf("Hello, world!\n");
    MPI_Finalize(); // Завершение работы с MPI.
    return 0;
}
```

Пример "Hello, world!"

```
#include <stdio.h>
#include <mpi.h> // Заголовочный файл MPI.

int main(int argc, char *argv[])
{
    int  numtasks, rank; // Номер и число процессов.
    MPI_Init(&argc, &argv); // Инициализация MPI.
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks); // Число потоков.
    MPI_Comm_rank(MPI_COMM_WORLD, &rank); // Номер текущего пот
    printf("Number of tasks= %d My rank= %d\n  numtasks, rank);
    MPI_Finalize(); // Завершение работы с MPI.
    return 0;
}
```

- MPI_Init(int *argc, char ***argv) – инициализация MPI-окружения. Все функции взаимодействия должны вызываться только после данной функции.
  - argc – указатель на параметр argc функции main.
  - argv – указатель на параметр argv функции main.
- MPI_Finalize() – завершение работы с MPI.
- MPI_Comm_size(MPI_Comm comm, int *size) – узнать число процессов в коммуникаторе (размер коммуникатора).
  - comm – коммуникатор MPI.
  - size – указатель на переменную, куда будет записано число процессов.
- MPI_Comm_rank(MPI_Comm comm, int *rank) – узнать номер данного процесса в коммуникаторе (rank).
  - comm – коммуникатор MPI.
  - rank – указатель на переменную, куда будет записан номер процесса.

Компиляция

- Предоставляет свою обертку для стандартного компилятора в системе.
- Названия компиляторов в среде Linux: mpicc, mpiCC, mpicxx, mpic++.
- Вызывает сторонний компилятор в системе (gcc, icc и т. д.) с набором опций.
- Для работы библиотеки необходимо подключить заголовочный файл mpi.h.

Example (Компиляция)

```
mpicc -o hello hello.c
```

Запуск

- Для инициализации окружения MPI необходим запуск через ее программы.
- Параметры запуска сильно зависят от версии библиотеки и используемого окружения.
- Для запуска на нескольких узлах использует rsh/ssh протокол.
- Основная команда для запуска – mpirun. Число процессов задается опцией -np.

Example (запуск)

```
mpirun -np 5 ./hello
```

Запуск

```
user@host:~/$ mpicc -o hello mpi_hello.c
user@host:~/$ mpirun -np 5 ./hello
Number of tasks = 5 My rank = 0
Number of tasks = 5 My rank = 3
Number of tasks = 5 My rank = 1
Number of tasks = 5 My rank = 4
Number of tasks = 5 My rank = 2
user@host:~/$
```

Особенности запуска

- Порядок строк может быть произвольным.
- MPI окружение гарантирует, что символы в отдельных строках различных процессов не будут перемешиваться.
- Запуск без команды mpirun обычно приводит к работе приложения в один поток.

Логины

- Протокол: ssh.
- Формат логина s77YXX.
- Y – 1-8, номер группы.
- XX – 01-20, номер студента.
- Пароль: b0587YXX.
- Имя сервера: head.vdi.mipt.ru.

При вводе пароля символы не отображаются на экране!
На все вопросы при входе отвечаем yes (полностью).

Example (Пример)

ssh s37101@head.vdi.mipt.ru

Редактирование

- Можно использовать консольные редакторы: nano, vim, mcedit.
- Можно использовать графический редактор на рабочем машине, для этого открываем удаленную папку.

Спасибо за внимание! Вопросы?