Средства и системы параллельного программирования

Семинар #7. Основы MPI, средство профилирования MPI-P

MPI

Существует множество реализаций MPI:

MPICH

OpenMPI

Microsoft MPI

mpi4py

……

Использование на локальной машине

Пакеты (Debian & Ubuntu): `libopenmpi-dev libopenmpi3 openmpi-bin openmpi-doc`

Компиляция программы: `mpicc prog.c -o prog`

`mpicxx` или `mpic++`

Запуск программы: `mpirun -np 4 ./prog arg1 arg2 arg3`

Использование на Polus

`module load SpectrumMPI` (предпочтительнее)

`module load OpenMPI`

Компиляция программы: `mpixlc prog.c -o prog`

`mpixlC. . .`

Запуск программы: через специальный планировщик mpisubmit.pl

`mpisubmit.pl -p 30 -w 00:05 a.out -- 3.14 2.72`

Основные функции MPI

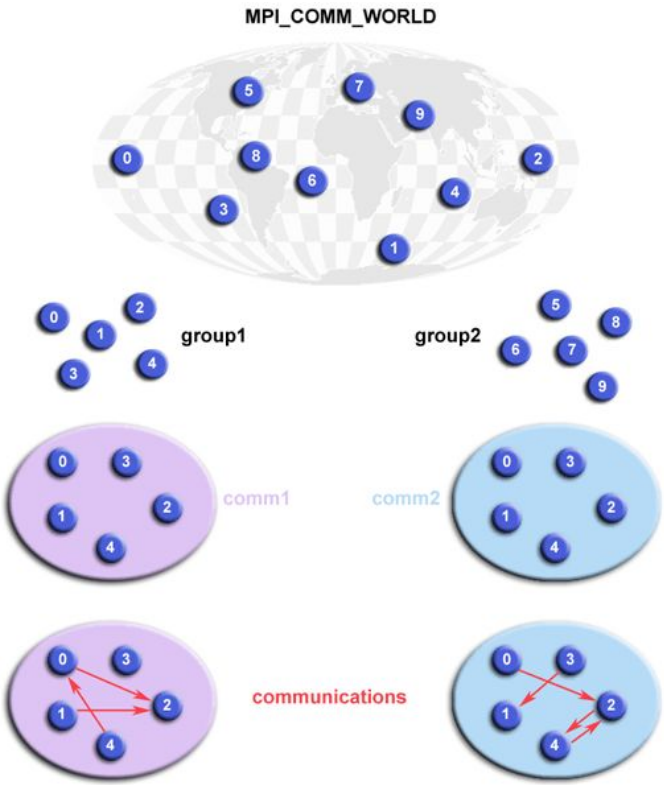
```
int MPI_Init(int *argc, char ***argv)

int MPI_Finalize()

int MPI_Comm_size(MPI_Comm comm, int *size)

int MPI_Comm_rank(MPI_Comm comm, int *rank)
```

Пока работаем только с MPI_COMM_WORLD!



MPI Point-to-point

```
int MPI_Send(const void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)
```

Отправка сообщения процессу dest. Отправляем count элементов типа datatype из буфера по адресу buf

Input Parameters

buf initial address of send buffer (choice)

count number of elements in send buffer (nonnegative integer)

datatype datatype of each send buffer element (handle)

dest rank of destination (integer)

tag message tag (integer)

comm communicator (handle)

MPI Point-to-point

```
int MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status * status)
```

Приём сообщения от процесса source. Принимаем не больше чем count элементов типа datatype. Если больше - сообщение обрезается, меньше либо равно - кладётся полностью (но лучше так не писать)

Output Parameters

buf initial address of receive buffer (choice)

status status object (Status)

Input Parameters

count maximum number of elements in receive buffer (integer)
datatype datatype of each receive buffer element (handle)

source rank of source (integer)

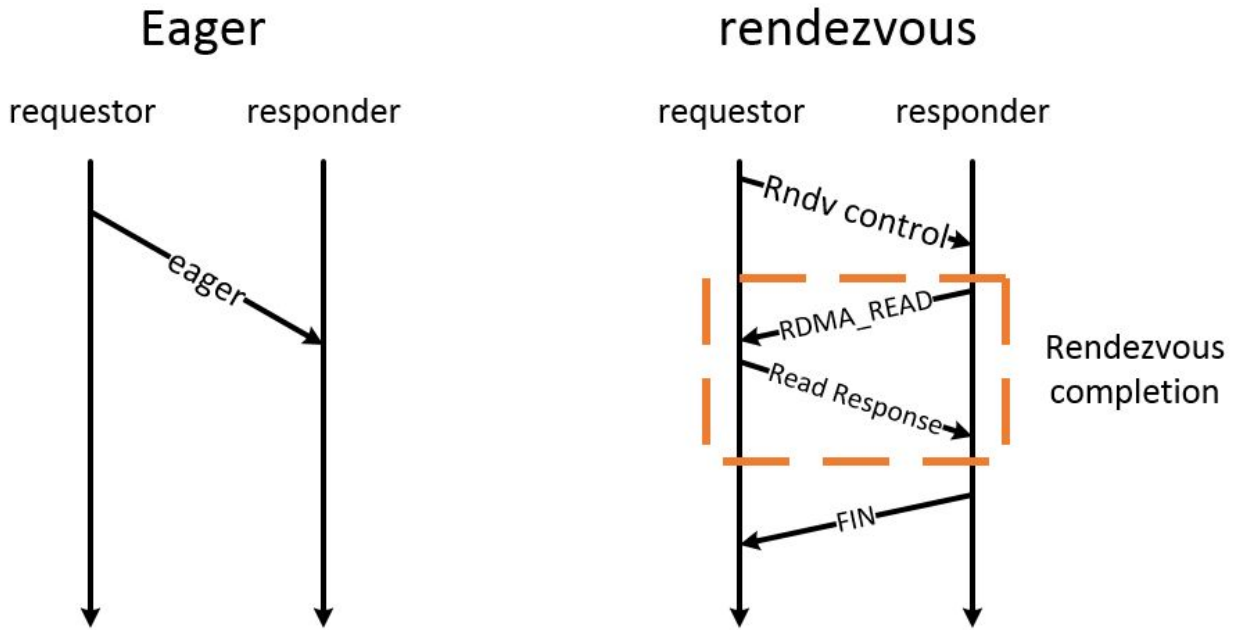
tag message tag (integer)

comm communicator (handle)

Типы данных в MPI

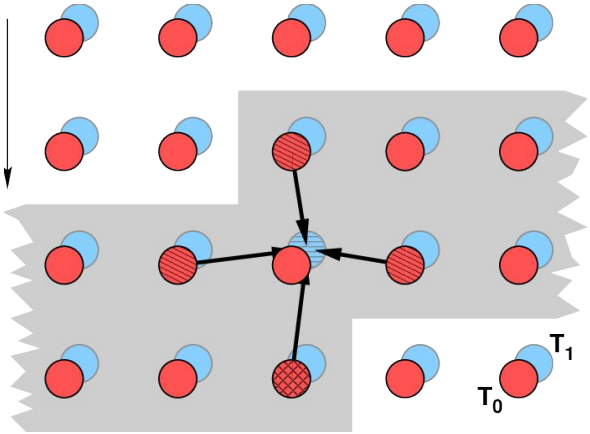
```
MPI_SHORT
MPI_INT
MPI_LONG
MPI_LONG_LONG
MPI_UNSIGNED_CHAR
MPI_UNSIGNED_SHORT
MPI_UNSIGNED
MPI_UNSIGNED_LONG
MPI_UNSIGNED_LONG_LONG
MPI_FLOAT
MPI_DOUBLE
MPI_LONG_DOUBLE
MPI_BYTE
```

Eager vs Rendesvouz



Задание

$$\Delta f = g$$



Задание (2)

Произвести итерации метода Якоби для уравнения Лапласа ($g = 0$) в 2D (двумерном) случае

Метод решения:

Построить сетку (на каждом процессе выделить массив размера, равного числу элементов в подобласти). Будем использовать **ленточные** подобласти (каждый процесс работает над своей горизонтальной частью сетки, ширина подобластей на процессах совпадает с шириной большой сетки)

Инициализировать начальное значение f случайным значением в каждой области сетки

До предустановленного числа итераций n_iter выполнять вычисления согласно методу Якоби

На последней итерации посчитать норму разности между решениями на двух соседних шагах времени на каждом процессе

Задание (3)

Требования к решению:

Запрещается хранить массив, соответствующий полной сетке, на одном процессе (за исключением запуска на 1 процессе)

Для коммуникации использовать только блокирующие point-2-point методы MPI, рассмотренные ранее на слайдах (или производные от них)

Можно предполагать, что размер сетки N - степень двойки. Сетка квадратная.

Произвести запуски на **Polus** (**через mpisubmit.pl !**)

Для фиксированного большого размера сетки произвести запуски при числе процессов P = {1, 2, 4, 8, 16, 32}, нарисовать графики T(P), S(P), E(P)

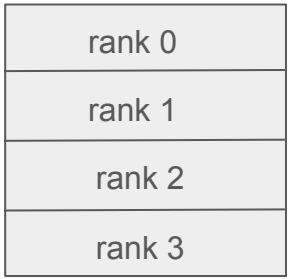
Дедлайн: 10.11, 17.11

Задание (доп. пояснения)

https://ece.uwaterloo.ca/~dwharder/nm/Lecture_materials/pdfs/7.2.3%20Laplace's%20equation.pdf - про дискретизацию уравнения Лапласа

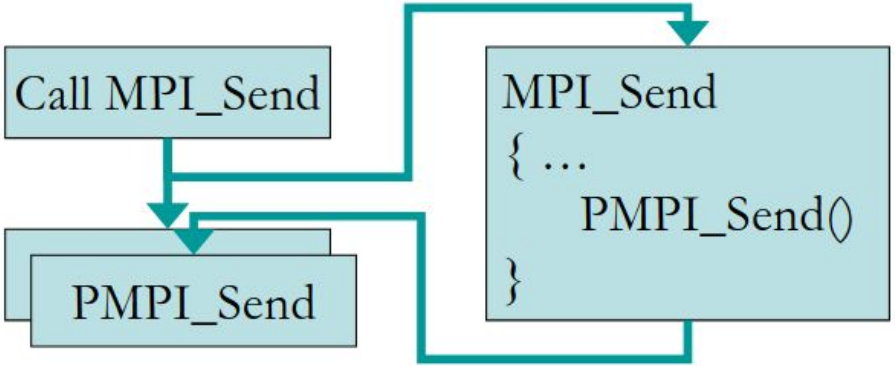
https://byjus.com/maths/jacobian-method/ - напоминалка про метод Якоби

$$f_{i,j}^{(t+1)} = \frac{1}{4}(f_{i+1,j}^{(t)} + f_{i-1,j}^{(t)} + f_{i,j+1}^{(t)} + f_{i,j-1}^{(t)})$$



mpiP (ДАЛЬШЕ ПОКА МОЖНО НЕ СМОТРЕТЬ)

Легковесный профилировщик для MPI-приложений, позволяет получить статистику по вызовам и времени работы операций MPI в программе



Профилирование обеспечивается засчёт обращений к средствам профилировки PMPI

Небольшой туториал

mpiP на Polus

Сборка на Polus:

```
wget https://github.com/LLNL/mpiP/archive/refs/tags/3.5.tar.gz
tar -xzf 3.5.tar.gz
cd mpiP-3.5
./configure
make
```

mpiP на Polus (2)

подключаем линковку библиотеки mpiP-3.5:

```
mpicc -g -Wl,-rpath=/home/lichmanov.d/mpiP-3.5 -L
/home/lichmanov.d/mpiP-3.5/ -lmpiP 1-hot-potato.c
```

определяем переменную окружения, например так:

```
export MPIP="-t 10.0 -k 2"
```

Запускаем приложение через mpisubmit.pl